

1. Bamboo Documentation Home	6
1.1 Getting started with Bamboo	6
1.1.1 Understanding the Bamboo CI Server	7
1.1.2 Getting started with Java and Bamboo	9
1.1.3 Getting started with .NET and Bamboo	13
1.1.4 Getting started with PHP and Bamboo	17
1.1.5 Getting started with Ruby and Bamboo	17
1.1.6 Using the Bamboo dashboard	17
1.1.6.1 Viewing Bamboo's agents	19
1.2 Using Bamboo	25
1.2.1 Deployment projects	26
1.2.1.1 Understanding deployment releases	28
1.2.1.2 Deployment projects workflow	29
1.2.1.3 A sample deployment project	30
1.2.1.4 Creating and configuring a deployment project	34
1.2.1.4.1 How deployment releases work	37
1.2.1.5 Creating a deployment environment	41
1.2.1.5.1 Tasks for deployment environments	44
1.2.1.5.2 Triggers for deployment environments	48
1.2.1.5.3 Permissions for deployment environments	50
1.2.1.5.4 Agents for deployment environments	50
1.2.1.5.5 Notifications for deployment environments	51
1.2.1.5.6 Variables for deployment environments	53
1.2.1.6 Managing deployment projects	54
1.2.1.7 Manually starting a deployment	56
1.2.2 Configuring plans	57
1.2.2.1 Viewing a plan's build information	59
1.2.2.2 Creating a plan	60
1.2.2.2.1 Creating a new plan	60
1.2.2.2.2 Cloning an existing plan	61
1.2.2.2.3 Importing a Maven 2 project	62
1.2.2.3 Using plan branches	63
1.2.2.4 Managing plans	70
1.2.2.4.1 Editing a plan's configuration	70
1.2.2.4.2 Configuring a plan's permissions	71
1.2.2.4.3 Disabling or deleting a plan	72
1.2.2.4.4 Labelling a plan	73
1.2.2.4.5 Modifying multiple plans in bulk	73
1.2.2.4.6 Moving plans to a different project	74
1.2.2.4.7 Configuring concurrent builds	76
1.2.2.4.8 Configuring the hanging build event	76
1.2.2.4.9 Configuring the build queue timeout event	77
1.2.2.4.10 Disabling build monitoring	78
1.2.3 Connecting to code repositories	79
1.2.3.1 Specifying the source repository	79
1.2.3.1.1 Bitbucket - Mercurial	81
1.2.3.1.2 CVS	82
1.2.3.1.3 Git	86
1.2.3.1.4 GitHub	88
1.2.3.1.5 Mercurial	89
1.2.3.1.6 Perforce	91
1.2.3.1.7 Subversion	95
1.2.3.2 Configuring a shared source repository	98
1.2.4 Triggering builds	98
1.2.4.1 Polling the repository for changes	100
1.2.4.2 Repository triggers the build when changes are committed	101
1.2.4.3 Cron-based scheduling	103
1.2.4.4 Single daily build	104
1.2.4.5 Running a plan build manually	104
1.2.4.5.1 Rerunning a failed stage	105
1.2.5 Using stages in a plan	106
1.2.5.1 Creating a stage	107
1.2.5.2 Editing a stage	108
1.2.5.3 Deleting a stage	109
1.2.6 Jobs and tasks	110
1.2.6.1 Creating a job	111
1.2.6.2 Configuring jobs	112
1.2.6.2.1 Configuring a job's requirements	112
1.2.6.2.2 Configuring a job's build artifacts	113
1.2.6.2.3 Configuring miscellaneous settings for a job	116
1.2.6.3 Disabling or deleting a job	118
1.2.6.3.1 Deleting a job's current working files	119
1.2.6.4 Configuring tasks	119
1.2.6.4.1 Checking out code	121
1.2.6.4.2 Configuring a builder task	122
1.2.6.4.3 Configuring a test task	141
1.2.6.4.4 Configuring a deployment task	151

1.2.6.4.5 Pattern matching reference	157
1.2.7 Sharing artifacts	158
1.2.8 Working with builds	161
1.2.8.1 Working with build results	162
1.2.8.1.1 Viewing a build result	162
1.2.8.1.2 Assigning responsibility for build failures	169
1.2.8.1.3 Configuring build results expiry for a plan	170
1.2.8.1.4 Deleting the results of a plan build	171
1.2.8.2 Working with comments	172
1.2.8.2.1 Commenting about a build result	172
1.2.8.2.2 Viewing comments about a build result	173
1.2.8.2.3 Viewing code check-in comments	174
1.2.8.3 Working with labels	175
1.2.8.3.1 Labelling a build result	175
1.2.8.3.2 Removing a label from a build result	176
1.2.8.3.3 Viewing labelled build results	176
1.2.8.3.4 Viewing popular labels	176
1.2.8.4 Quarantining failing tests	176
1.2.8.5 Setting up plan build dependencies	177
1.2.8.5.1 Dependency blocking strategies	178
1.2.8.6 Viewing test statistics for a job	179
1.2.8.7 Reordering jobs in the build queue	180
1.2.8.8 Stopping an active build	180
1.2.9 Getting feedback	181
1.2.9.1 Notifications	181
1.2.9.1.1 Displaying the wallboard	182
1.2.9.1.2 Configuring notifications for a plan and its jobs	183
1.2.9.1.3 Working with Instant Messenger (IM) notifications	193
1.2.9.1.4 Subscribing to RSS feeds	194
1.2.9.2 Reporting	194
1.2.9.2.1 Viewing build statistics for all users	195
1.2.9.2.2 Viewing build results for an author	195
1.2.9.2.3 Generating reports on selected authors	196
1.2.9.2.4 Generating reports across multiple plans	199
1.2.9.2.5 Viewing the Clover code-coverage for a plan	204
1.2.9.2.6 Viewing the Clover code-coverage for a build	205
1.2.10 Integrating Bamboo with Atlassian applications	207
1.2.10.1 Integrating Bamboo with JIRA	208
1.2.10.1.1 Viewing linked JIRA issues	209
1.2.10.1.2 Linking JIRA issues to a build	211
1.2.10.1.3 Creating JIRA issues from a build	212
1.2.10.2 Integrating Bamboo with Confluence	213
1.2.10.3 Integrating Bamboo with Stash	214
1.2.10.4 Integrating Bamboo with FishEye	214
1.2.10.5 Integrating Bamboo with HipChat	215
1.2.11 Managing your user profile	216
1.2.11.1 Changing your password	216
1.2.11.2 Changing your notification preferences	217
1.2.11.3 Associating your author name with your user profile	217
1.2.12 Bamboo variables	217
1.2.12.1 Defining global variables	222
1.2.12.2 Defining plan variables	223
1.2.13 Bamboo permissions	224
1.3 Administering Bamboo	225
1.3.1 System settings	226
1.3.1.1 Viewing Bamboo's system information	227
1.3.1.2 Updating your Bamboo license details	228
1.3.1.3 Specifying Bamboo's title	229
1.3.1.4 Specifying Bamboo's URL	229
1.3.1.5 Logging in Bamboo	229
1.3.1.6 Enabling GZIP compression	232
1.3.1.7 Enabling Bamboo's Remote API	232
1.3.1.8 Configuring system properties	232
1.3.1.9 Finding Your Bamboo Support Entitlement Number (SEN)	234
1.3.1.10 Configuring Gravatar support	236
1.3.2 Agents and capabilities	236
1.3.2.1 Configuring agents	237
1.3.2.1.1 Viewing a Bamboo agent's details	238
1.3.2.1.2 Creating a local agent	239
1.3.2.1.3 Editing an agent's details	240
1.3.2.1.4 Disabling or deleting an agent	241
1.3.2.1.5 Monitoring agent status	241
1.3.2.2 Configuring capabilities	243
1.3.2.2.1 About capabilities and requirements	244
1.3.2.2.2 Viewing a capability's agents and jobs	246
1.3.2.2.3 Defining a new executable capability	247
1.3.2.2.4 Defining a new JDK capability	252

1.3.2.2.5 Defining a new version control capability	255
1.3.2.2.6 Defining a new custom capability	256
1.3.2.2.7 Modifying and deleting capabilities	257
1.3.2.3 Remote agents	259
1.3.2.3.1 Disabling and enabling remote agents support	259
1.3.3 Working with Elastic Bamboo	260
1.3.3.1 About Elastic Bamboo	260
1.3.3.1.1 Elastic Bamboo Costs	262
1.3.3.2 Getting started with Elastic Bamboo	262
1.3.3.3 Configuring Elastic Bamboo	264
1.3.3.3.1 Generating your AWS Private Key File and Certificate File	266
1.3.3.3.2 Configuring elastic instances to use the EBS	268
1.3.3.3.3 Elastic IPs and Bamboo	272
1.3.3.4 Managing Elastic Bamboo	273
1.3.3.4.1 Managing your elastic images	273
1.3.3.4.2 Managing your elastic instances	291
1.3.3.4.3 Managing your elastic agents	299
1.3.3.5 Running job builds using Elastic Bamboo	304
1.3.3.6 Disabling Elastic Bamboo	305
1.3.4 Users and permissions	305
1.3.4.1 Managing users	306
1.3.4.1.1 Creating a user	306
1.3.4.1.2 Changing a user's password or details	307
1.3.4.1.3 Deleting or deactivating a user	307
1.3.4.1.4 Granting administration rights to a user	308
1.3.4.1.5 Changing usernames	309
1.3.4.2 Managing groups	309
1.3.4.2.1 Creating a group	309
1.3.4.2.2 Deleting a group	310
1.3.4.2.3 Changing members of groups	310
1.3.4.3 Connecting to external user directories	311
1.3.4.3.1 Integrating Bamboo with Crowd	311
1.3.4.3.2 Integrating Bamboo with LDAP	313
1.3.4.4 Managing permissions	322
1.3.4.4.1 Granting plan permissions in bulk	322
1.3.4.4.2 Granting global permissions to users or groups	323
1.3.4.4.3 Allowing anonymous access to Bamboo	324
1.3.4.4.4 Global security and permission properties	325
1.3.5 Connecting Bamboo to an external database	327
1.3.5.1 PostgreSQL 8.2	327
1.3.5.2 MySQL 5.1	329
1.3.5.2.1 Tomcat and External MySQL Datasource Example	332
1.3.5.3 Oracle 11g	332
1.3.5.4 Microsoft SQL Server 2005 and 2008	334
1.3.5.5 How do I connect Bamboo to an unsupported database	339
1.3.6 Add-ons	341
1.3.6.1 Add-on blacklist	341
1.3.6.2 Enabling the Clover add-on	341
1.3.7 Data and backups	345
1.3.7.1 Locating important directories and files	345
1.3.7.2 Specifying Bamboo's working directory	347
1.3.7.3 Viewing your database connection details	348
1.3.7.4 Moving your Bamboo data to a different database	348
1.3.7.5 Optimising or re-indexing data	349
1.3.7.6 Specifying a backup schedule	349
1.3.7.7 Exporting data for backup	351
1.3.7.8 Importing data from backup	352
1.3.7.9 Configuring global build results expiry	353
1.3.7.10 Importing data from Jenkins	355
1.3.8 Security	360
1.3.8.1 Agent authentication	361
1.3.8.2 Elastic Bamboo Security	362
1.3.8.3 Bamboo cookies	364
1.3.8.4 Best practices for Bamboo security	365
1.3.8.5 Securing your remote agents	366
1.4 Bamboo release notes	368
1.4.1 Bamboo release summary	371
1.4.2 Bamboo 5.0 beta 1 Release Notes	374
1.4.3 Bamboo 5.0 Release Notes	375
1.4.3.1 Bamboo 5.0.1 Release Notes	378
1.4.3.2 Bamboo 5.0 Upgrade Guide	383
1.4.4 Bamboo 5.0 beta 3 Release Notes	385
1.4.5 Bamboo 5.0 beta 2 Release Notes	386
1.4.6 Bamboo security advisories	387
1.4.6.1 Bamboo Security Advisory 2012-08-28	388
1.4.6.2 Bamboo Security Advisory 2012-05-17	388
1.4.6.3 Bamboo Security Advisory 2012-01-31	390

1.4.6.4 Bamboo Security Advisory 2011-11-22	392
1.4.6.5 Bamboo Security Advisory 2011-03-29	394
1.4.6.6 Bamboo Security Advisory 2010-05-04	395
1.4.6.7 Bamboo Security Advisory 2009-03-09	397
1.4.6.8 Bamboo Security Advisory 2008-02-08 (Bamboo 2.0 Beta)	399
1.4.6.9 Bamboo Security Advisory 2013-07-16	400
1.4.7 Bamboo 5.0-rc1 Release Notes	401
1.5 Installing and upgrading Bamboo	402
1.5.1 Supported platforms	403
1.5.1.1 End of Support Announcements for Bamboo	405
1.5.2 Bamboo system requirements	407
1.5.2.1 Hardware sizing considerations	409
1.5.3 Bamboo installation guide	410
1.5.3.1 Bamboo installation guide for Linux	411
1.5.3.2 Bamboo installation guide for Mac	412
1.5.3.3 Bamboo installation guide for Windows	414
1.5.3.4 Running the Setup Wizard	416
1.5.3.5 Bamboo EAR-WAR installation guide	419
1.5.3.5.1 Configuring Tomcat to Use HttpOnly Session ID Cookies	422
1.5.3.6 Bamboo remote agent installation guide	422
1.5.3.6.1 Configuring remote agent capabilities using bamboo-capabilities.properties	426
1.5.3.6.2 Legacy remote agent installation guide	426
1.5.4 Bamboo upgrade guides	428
1.5.4.1 Bamboo generic upgrade guide	429
1.6 Bamboo FAQ	431
1.6.1 Installation FAQ	433
1.6.1.1 Changing the Root Context Path	433
1.6.1.2 Configuring Bamboo on start-up	434
1.6.1.3 Getting Bamboo to use the jetty.xml file	436
1.6.1.4 Installation notes for Bamboo on JBoss 4.x	437
1.6.1.5 Running Bamboo as a Service on Windows	438
1.6.1.6 Running Bamboo over HTTPS	439
1.6.1.7 Running Bamboo service on Windows as the local user	444
1.6.1.8 Setting up JNDI mail on JBoss 4.2.2	446
1.6.1.9 Setting up JNDI on Jetty	447
1.6.2 Support Policies	449
1.6.2.1 Bamboo Support Policy	449
1.6.2.2 Bug Fixing Policy	450
1.6.2.3 Deploying Multiple Atlassian Applications in a Single Tomcat Container	450
1.6.2.4 How to Report a Security Issue	450
1.6.2.5 New Features Policy	451
1.6.2.6 Patch Policy	452
1.6.2.7 Security Advisory Publishing Policy	452
1.6.2.8 Security Patch Policy	452
1.6.2.9 Severity Levels for Security Issues	453
1.6.3 Usage FAQ	454
1.6.3.1 Backing up Bamboo instances over 4GB	454
1.6.3.2 Bamboo Database Schema	455
1.6.3.3 Binding Bamboo to one IP address	455
1.6.3.4 Can Bamboo build and test non-Java projects	456
1.6.3.5 Can multiple plans share a common 3rd-party directory	457
1.6.3.6 Changing Bamboo database settings	457
1.6.3.7 Finding the Support Entitlement Number (SEN)	457
1.6.3.8 How do I manually set the version of new Subversion workspaces	458
1.6.3.9 Securing your repository connection	458
1.6.3.10 Changing the remote agent heartbeat interval	459
1.6.3.11 Cloning a Bamboo instance	460
1.6.3.12 CVS Error logging in Bamboo	460
1.6.3.13 Do I have to upgrade all remote agents for Bamboo Release 2.1.2	461
1.6.3.14 Enable User Management debug logging in Bamboo	461
1.6.3.15 Hibernate errors in logs after upgrading to Bamboo 2.0	461
1.6.3.16 How do I construct a cron expression in Bamboo	461
1.6.3.17 How do I disable SSH access to my elastic instances	462
1.6.3.18 How do I shut down my elastic instances if I have restarted my Bamboo server	462
1.6.3.19 How do I stop Bamboo from shutting itself down and restarting	462
1.6.3.20 How do I stop the Bamboo server from automatically configuring my remote agent's capabilities	463
1.6.3.21 JUnit parsing in Bamboo	463
1.6.3.22 Known issues with CVS in Bamboo	468
1.6.3.23 Monitoring and Profiling Bamboo	468
1.6.3.24 Monitor Memory usage and Garbage Collection in Bamboo	469
1.6.3.25 Moving Bamboo-Home of an agent	470
1.6.3.26 Performing a thread dump.	470
1.6.3.27 Removing Coverage plug-in data from the Bamboo database	470
1.6.3.28 Restoring passwords to recover admin users	471
1.6.3.29 Send Errors to stderr - Script Builder in Visual Studio WinXP to build Solutions Files	473
1.6.3.30 Using Bamboo with Clover	474
1.6.3.30.1 Getting gcov results in Clover coverage summary	475

1.6.3.31 Working with Sun JAVA libraries	475
1.6.3.32 Bamboo indicates that my Ant or Maven builds failed, even though they were successful	476
1.6.3.33 How can I pass bamboo variables to my build script	476
1.6.4 Contributing to the Bamboo documentation	477
1.6.4.1 Tips of the Trade	478
1.6.4.2 Bamboo Documentation in Other Languages	479
1.7 Bamboo resources	479
1.8 Glossary	480
1.8.1 activity log	482
1.8.2 agent	482
1.8.3 agent-specific capability	482
1.8.4 artifact	482
1.8.5 authors in Bamboo	482
1.8.6 build	482
1.8.7 build activity	482
1.8.8 build duration	482
1.8.9 build log	482
1.8.10 build queue	483
1.8.11 build result	483
1.8.12 build telemetry	483
1.8.13 capability	483
1.8.14 child	483
1.8.15 committer	483
1.8.16 custom capability	483
1.8.17 default repository	483
1.8.18 elastic agent	484
1.8.19 elastic Bamboo	484
1.8.20 elastic block store	484
1.8.21 elastic image	484
1.8.22 elastic instance	484
1.8.23 executable	484
1.8.24 favourites	484
1.8.25 global permission	484
1.8.26 job	485
1.8.27 label	485
1.8.28 local agent	485
1.8.29 parent	485
1.8.30 permission	485
1.8.31 plan	485
1.8.32 plan permission	485
1.8.33 projects in Bamboo	485
1.8.34 queue	486
1.8.35 reason	486
1.8.36 remote agent	486
1.8.37 remote agent supervisor	486
1.8.38 requirement	486
1.8.39 shared capability	486
1.8.40 stage	486
1.8.41 stock images	487
1.8.42 task	488
1.8.43 triggering	489
1.8.44 watcher	489

Bamboo Documentation Home

Getting started

[Understanding Bamboo](#)

[Getting started with Java and Bamboo](#)

[Getting started with .NET and Bamboo](#)

[Configuring plans](#)

[Deployment projects using Bamboo](#)

[Evaluator](#)

[Developer](#)

[Getting support](#)

Administering

[All administration topics](#)

[Release notes](#)

[Installing and upgrading](#)

[System settings](#)

[Users and permissions](#)

[Data and backups](#)

[Security](#)

Popular topics

[Automatic plan branches](#)

[Quarantining intermittent tests](#)

[Integrating Bamboo with Atlassian applications](#)

[Bamboo remote agent installation guide](#)

[Getting feedback](#)

[Importing data from Jenkins](#)

What's new in Bamboo »

Getting started with Bamboo

Bamboo is a

continuous integration (CI) server. Bamboo



assists software development teams by providing:

- automated building and testing of software source-code status.
- updates on successful/failed builds.
- reporting tools for statistical analysis.

Quick start guide ➔

Please see the following pages for information about getting started with Bamboo:

- [Understanding the Bamboo CI Server](#) - a conceptual overview of using Bamboo for continuous integration (CI).
- [Getting started with Java and Bamboo](#) - a guide to setting up a simple CI workflow.
- [Getting started with .NET and Bamboo](#) - a guide to setting up a simple CI workflow on Windows.

Understanding the Bamboo CI Server

Bamboo is a continuous integration (CI) server that can be used to automate the release management for a software application, creating a continuous delivery pipeline.

What does this mean?

CI is a software development methodology in which a build, unit tests and integration tests are performed, or triggered, whenever code is committed to the repository, to ensure that new changes integrate well into the existing code base. Integration builds provide early 'fail fast' feedback on the quality of new changes.

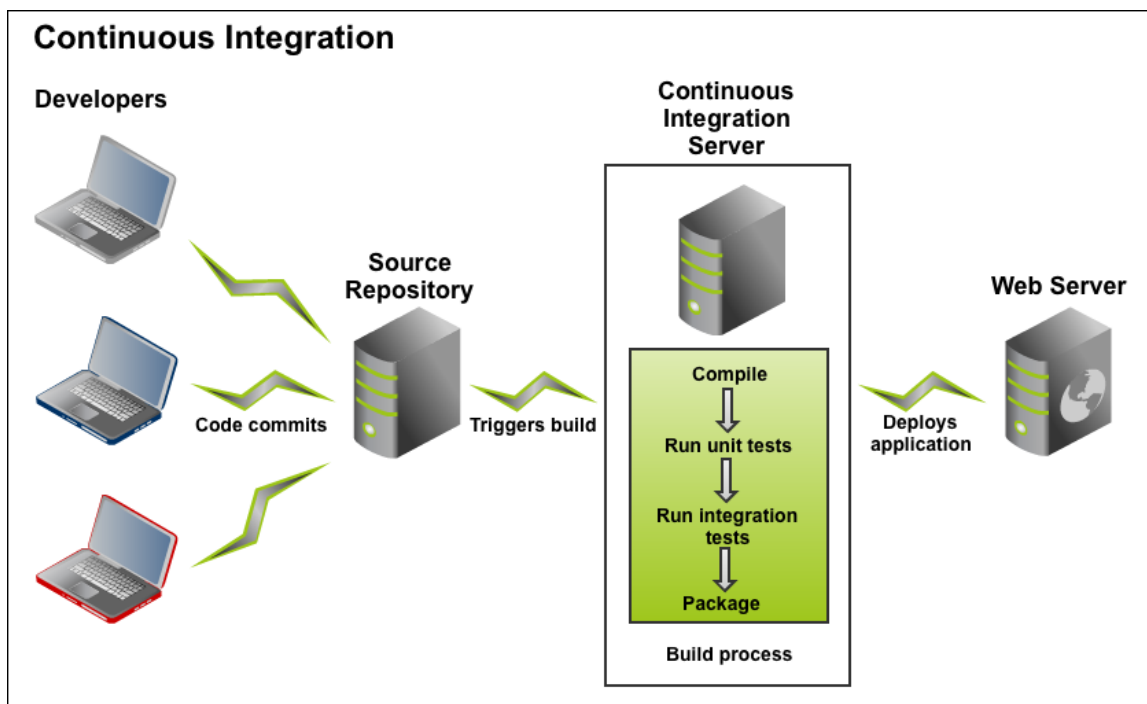
Release management describes the steps that are typically performed to release a software application, including building and functional testing, tagging releases, assigning versions, and deploying and activating the new version in production.

On this page:

- [What problem does Bamboo solve?](#)
- [How does Bamboo do this?](#)
- [What does Bamboo need?](#)
- [How is a Bamboo workflow organised?](#)

Related Pages:

- [Getting started with Java and Bamboo](#)
- [Getting started with .NET and Bamboo](#)
- [Using Bamboo](#)
- [Installing and upgrading Bamboo](#)



What problem does Bamboo solve?

If you are a solo developer, then using Bamboo gives you:

- an automated, and so reliable, build and test process, leaving you free to code more.
- a way to manage builds that have different requirements or targets.
- automatic deployment to a server, such as the App Store or the Android Market.

If you work in a team, then as well as the above advantages, using Bamboo also means that:

- your build and test process is not dependent on a specific local environment.
- builds and integration tests are triggered automatically as soon as a developer commits code (continuous integration).

If you work on a large, complex application, then, in addition to all the above advantages, using Bamboo means that:

- you can optimise build performance through parallelism.
- you can leverage elastic resources.
- you can deploy continuously, for example to user acceptance testing (UAT).
- you can implement release management.

How does Bamboo do this?

- Bamboo is the central management server which schedules and coordinates all work.
- Bamboo itself has interfaces and plugins for lots of types of work.
- Bamboo first gets your source from a source repository (lots of plugins here for a variety of systems).
- Then Bamboo starts the build - that can be done by calling something like MSBuild to build your Visual Studio solution, or Maven to call your compiler and linker - whatever you use.
- Once your solution or project is built, you have "artifacts" (build results, for example, an executable app, config files, etc.).
- You can do additional things with the build artifacts:
 - zip them up into a ZIP file and copy them somewhere.
 - run an install builder on them and create an MSI.
 - install them on a test server to make sure everything installs just fine.
- Bamboo provides a web front-end for configuration and for reporting the status of builds.

What does Bamboo need?

Bamboo schedules and coordinates the work involved in building and testing your application. Therefore, to use Bamboo, you will need to already have the following set up:

- a code repository that contains the complete source code for the project.
- build scripts
- test suites

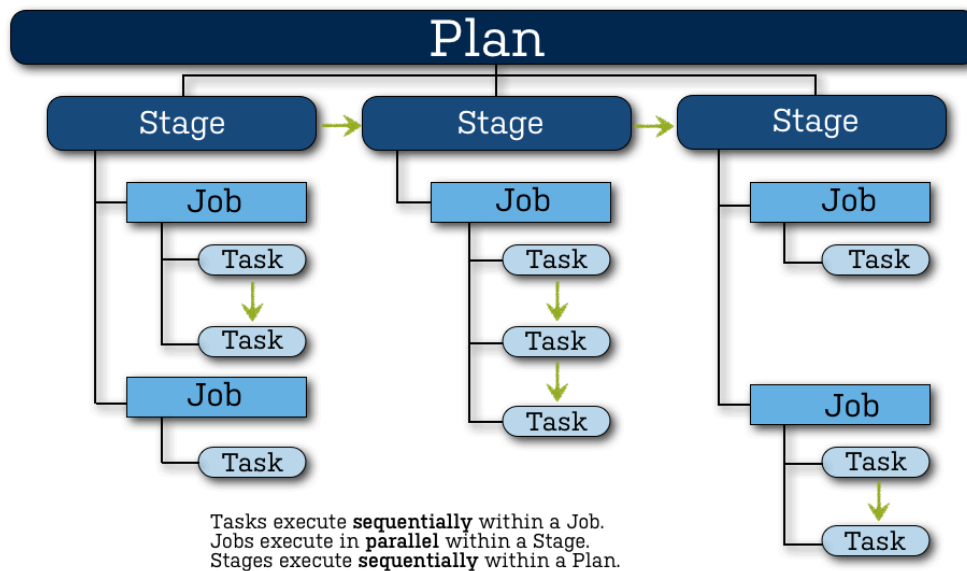
It is generally assumed that the person who commits a change to the code is responsible for fixing any resulting build errors immediately.

How is a Bamboo workflow organised?

Bamboo uses the concept of a 'plan' with 'jobs' and 'tasks' to configure and order the actions in the workflow.

Project	• Has one, or more, plans. • Provides reporting (using the wallboard, for example) across all plans in the project. • Provides links to other applications.
Plan	• Has a single stage, by default, but can be used to group jobs into multiple stages. • Processes a series of one or more stages that are run sequentially using the same repository. • Specifies the default repository. • Specifies how the build is triggered, and the triggering dependencies between the plan and other plans in the project. • Specifies notifications of build results. • Specifies who has permission to view and configure the plan and its jobs. • Provides for the definition of plan variables.
Stage	• Has a single job, by default, but can be used to group multiple jobs. • Processes its jobs in parallel, on multiple agents (where available). • Must successfully complete all its jobs before the next stage in the plan can be processed. • May produce artifacts that can be made available for use by a subsequent stage.

Job	- Processes a series of one or more tasks that are run sequentially on the same agent. - Controls the order in which tasks are performed. - Collects the requirements of individual tasks in the job, so that these requirements can be matched with agent capabilities. - Defines the artifacts that the build will produce. - Can only use artifacts produced in a previous stage. - Specifies any labels with which the build result or build artifacts will be tagged.
Task	- Is a small discrete unit of work, such as source code checkout, executing a Maven goal, running a script, or parsing test results. - Is run sequentially within a job on a Bamboo working directory.



Getting started with Java and Bamboo

This page describes how your development team can start using the Bamboo continuous integration server to get rapid feedback on your Java project.

i You may want to read [Understanding the Bamboo CI Server](#) first.

We assume that you already have:

- Bamboo installed and running. See the [Bamboo installation guide](#) for details. You'll want user accounts in Bamboo for each member of your team.
- Source code under version control. Each team member will have access to the repository.
- Tests, as part of the source code for the project.
- A command (e.g. a Maven goal) that builds the code and executes the tests.

The continuous integration workflow we want is:

1. A developer commits code.
2. Bamboo builds the project:
 - a. Connects to the repository and checks out the source code.
 - b. Compiles the code.
 - c. Runs the unit and integration tests.
3. Bamboo provides feedback on the test results.

How do we achieve this with Bamboo?

Well, we'll create a new Bamboo plan that knows how to check out and build our source code, and then report on our test results.

On this page:

Create a Bamboo plan

1. Plan details
2. Choose a source repository
3. Triggering the build
4. Configure tasks
 - The source code checkout task*
 - The builder task*
 - Getting the test results*
5. Go!

Get feedback

Related pages:

- [Getting started with .NET and Bamboo](#)
- [Getting started with PHP and Bamboo](#)
- [Getting started with Ruby and Bamboo](#)

Create a Bamboo plan

A Bamboo plan is where you define the details of your continuous integration workflow.

A plan allows us to specify a source code repository, when Bamboo gets triggered to run the build, and how Bamboo should provide feedback on the test results.

1. Plan details

Click **Create Plan** in the menu bar, and then **Create a New Plan**.

Every plan belongs to a project. We don't have a project yet, so choose **Project > New Project**, and enter details for both the project and plan.

See [Configuring plans](#) for details.

Plan Details

Project	Project 1
	Select or add a Project that the new Plan will be created in.
Plan Name	
	How do you want to identify the new Plan?
Plan Key	
	This is the key for the plan which must be unique within a project. In conjunction with the project key, it is used to identify a build in URLs, trigger scripts and API calls. The key must contain only uppercase alphanumeric characters. e.g. "CORE"
Plan Description	
	Choose a meaningful description for the new Plan. For example, "JIRA Release Plan".

2. Choose a source repository

Bamboo needs to know where the source code repository is located, and needs access to the repo so that it can check out the code when it runs a build.

Choose the repository type from **Source Repository**, and provide access details such as username and password.

See [Connecting to code repositories](#) for details.

Source Repositories

Source Repository **Git**
Git support works best if the Git executable [capability](#) is defined for agents. If not defined, Bamboo will use JGit, which currently does not support submodules.

Repository URL *
The URL of Git repository.

Branch
The name of the branch (or tag) containing source code.

Authentication Type **None**
☒ Use shallow clones
Fetches the shallowest commit history possible. Do not use if your build depends on full repository history.

3. Triggering the build

We can choose how Bamboo gets triggered to run the plan build.

We want Bamboo to build the project whenever code is checked into the repository.

Choose **Trigger type** > **Repository triggers the build...**, and optionally, specify an IP address for the repository server.

See [Triggering builds](#) for details.

Trigger

Trigger type * **Repository triggers the build when checked in**
How should Bamboo trigger Builds for this Plan? (Dependent Builds are automatically triggered)

Trigger IP Addresses
(Optional) Bamboo ensures that triggers originate from IP addresses of the repository server(s). You can authorise additional IP addresses here, separated by a comma.

4. Configure tasks

Each plan needs to have one or more tasks specified. Tasks do the real work of the plan.

The source code checkout task

A newly created plan has a default Source Code Checkout task that gets the source code from the source repository specified earlier.

See [Checking out code](#) for details.

The builder task

We also want to compile the code, and run the unit and integration tests. We'll add a builder task to the Bamboo plan to do that. We assume that your project already has a build process set up that Bamboo can call upon.

Click **Add Task**, then **Builder** and choose the task that matches the build tool for your project. Expand one of the following sections to see configuration details specific to that builder task:

▼ [Ant...](#)

Ant Configuration

Task Description

Executable

Ant  [Add New Executable](#)

Build File

Target

The target you want to execute. You can also define system properties such as `-Djava.awt.headless=true`.

See <http://ant.apache.org/manual/index.html> for information about Ant.

▼ [Maven 3.0...](#)

Maven 3.x Configuration

Task Description

Executable

Maven 3  [Add New Executable](#)

Goal

The goal you want to execute. You can also define system properties such as `-Djava.awt.headless=true`.

☐ Use Maven Return Code

When determining build success, Bamboo checks Maven return code and searches the log for "BUILD SUCCESS". By checking this option, you will configure Bamboo to skip log parsing. This may fail on some Maven versions/operating systems.

Bamboo also supports Maven 1.0 and Maven 2.0.

See <http://maven.apache.org/guides/index.html> for information about Maven.

▼ [Grails...](#)

Grails Configuration

Task Description

Executable

 [Add New Executable](#)

Grails Commands

Use a new line to separate Grails commands. Bamboo will automatically append `"-non-interactive"` to each command

See <http://grails.org/doc/latest/guide/index.html> for information about Grails.

Note that:

- A build tool needs to be installed on the Bamboo server machine before you can use the Bamboo task.

- There are plugins available for Bamboo that add build tasks for other tools, such as Gant and Gradle. See <https://plugins.atlassian.com/plugin/details/27818?versionId=851052> for details.

Getting the test results

Your tests will be run when the builder task compiles the code. Each of the builder tasks above has a section to tell Bamboo to expect test results and where to look for them. You can specify a custom results location if your project directory doesn't use the conventional structure.

Where should Bamboo look for the test result files?

- ☒ The build will produce test results.
If checked, the build will fail if no tests are found. Test output must be in **JUnit** XML format.

Test Results Directory

- ☒ Look in the standard test results directory.
- ☐ Specify custom results directories

Where should Bamboo look for the test result files?

See [Jobs and tasks](#) for details.

5. Go!

Enable the plan, and click **Create**.

You should see the plan run. The 'Plan Summary' tab will report whether the build succeeded or not.

Tests in the appropriate directory in the source code repository will be run automatically as part of the build, and the test results will be displayed in Bamboo.

Now, whenever you commit a change to the repository, Bamboo will build your source code and report on your test results.

Get feedback

Bamboo displays a summary of the results of the build on the dashboard.

You can get further information about the build in the following ways:

- Build results for one or more plans can be displayed on a wallboard.
- You can get notifications about build results sent to you by email, IM and RSS feed.
- You can get build statistics about plans, and about developers contributing code to the build.
- You can drill down into the results to see the code changes that triggered the build, and the tests that were run for that build.

See [Getting feedback](#) for details.

Getting started with .NET and Bamboo

This page describes how your development team can start using the Bamboo continuous integration server to get rapid feedback on your .NET project.

 You may want to read [Understanding the Bamboo CI Server](#) first.

We assume that you already have:

- Bamboo installed and running. See the [Bamboo installation guide](#) for details. You'll want user accounts in Bamboo for each member of your team.
- Source code under version control. Each team member will have access to the repository.
- Tests, as part of the source code for the project.
- A command that builds the code and executes the tests.

The continuous integration workflow we want is:

1. A developer commits code.
2. Bamboo builds the project:
 - a. Connects to the repository and checks out the source code.
 - b. Compiles the code.
 - c. Runs the unit and integration tests.
3. Bamboo provides feedback on the test results.

How do we achieve this with Bamboo?

Well, we'll create a new Bamboo plan that knows how to check out and build our source code, and then report on our test results.

On this page:

Create a Bamboo plan

1. Plan details
2. Choose a source repository
3. Triggering the build
4. Configure tasks
 - The source code checkout task*
 - The builder task*
 - Getting the test results*
5. Go!

Get feedback

Related pages:

- [Getting started with Java and Bamboo](#)
- [Getting started with PHP and Bamboo](#)
- [Getting started with Ruby and Bamboo](#)

Create a Bamboo plan

A Bamboo plan is where you define the details of your continuous integration workflow.

A plan allows us to specify a source code repository, when Bamboo gets triggered to run the build, and how Bamboo should provide feedback on the test results.

1. Plan details

Click **Create Plan** in the menu bar, and then **Create a New Plan**.

Every plan belongs to a project. We don't have a project yet, so choose **Project > New Project**, and enter details for both the project and plan.

See [Configuring plans](#) for details.

Plan Details

Project	Project 1 Select or add a Project that the new Plan will be created in.
Plan Name	How do you want to identify the new Plan?
Plan Key	This is the key for the plan which must be unique within a project. In conjunction with the project key, it is used to identify a build in URLs, trigger scripts and API calls. The key must contain only uppercase alphanumeric characters. e.g. "CORE"
Plan Description	Choose a meaningful description for the new Plan. For example, "JIRA Release Plan".

2. Choose a source repository

Bamboo needs to know where the source code repository is located, and needs access to the repo so that it can check out the code when it runs a build.

Choose the repository type from **Source Repository**, and provide access details such as username and password.

See [Connecting to code repositories](#) for details.

Source Repositories

Source Repository **Git**

Git support works best if the Git executable [capability](#) is defined for agents. If not defined, Bamboo will use JGit, which currently does not support submodules.

Repository URL *



The URL of Git repository.

Branch

The name of the branch (or tag) containing source code.

Authentication Type **None**

☒ Use shallow clones

Fetches the shallowest commit history possible. Do not use if your build depends on full repository history.

3. Triggering the build

We can choose how Bamboo gets triggered to run the plan build.

We want Bamboo to build the project whenever code is checked into the repository.

Choose **Trigger type** > **Repository triggers the build...**, and optionally, specify an IP address for the repository server.

See [Triggering builds](#) for details.

Trigger

Trigger type* **Repository triggers the build when checked in**

How should Bamboo trigger Builds for this Plan? (Dependent Builds are automatically triggered)

Trigger IP Addresses

(Optional) Bamboo ensures that triggers originate from IP addresses of the repository server(s). You can authorise additional IP addresses here, separated by a comma.

4. Configure tasks

Each plan needs to have one or more tasks specified. Tasks do the real work of the plan.

The source code checkout task

A newly created plan has a default Source Code Checkout task that gets the source code from the source repository specified earlier.

See [Checking out code](#) for details.

The builder task

We also want to compile the code. We'll add a builder task to the Bamboo plan to do that. We assume that your project already has a build process set up that Bamboo can call upon.

Click **Add Task**, then **Builder** and choose the task that matches the build tool for your project. Expand one of the following sections to see configuration details specific to that builder task:

▼ [MSBuild...](#)

MSBuild Configuration

Task Description

Executable*

MSBuild v2.0 (32bit)



[Add New Executable](#)

Project File*

YourSolution.sln

The Solution, Project File or MSBuild project to execute when this Job Builds

Options

The MSBuild.exe command line switches you wish to include.

See <http://msdn.microsoft.com/en-us/library/ms171452%28v=vs.90%29.aspx> for information about MSBuild.

▼ [NAnt...](#)

NAnt Configuration

Task Description

Executable*

NAnt



[Add New Executable](#)

Build File

default.build

The name of the NAnt build file that you want to execute when this Job builds

Targets*

run

The NAnt targets you want Bamboo to execute when this Job builds

Options

The NAnt command line options you wish to include.

See <http://nant.sourceforge.net/> for information about NAnt.

▼ [Visual Studio...](#)

Visual Studio Configuration

Task Description

Executable*

Visual Studio 2010



[Add New Executable](#)

Solution*

The Visual Studio solution file you want Bamboo to execute when this Job builds

Options*

The devenv command line options you wish to include.

Platform*

x86



The platform toolset required to compile your Solution.

See <http://www.microsoft.com/visualstudio> for information about Visual Studio.

Note that a build tool needs to be installed on the Bamboo server machine before you can use the Bamboo task.

See [Configuring a builder task](#) for details.

Getting the test results

Now we want to run the unit and integration tests, and display the results from those. You need to set up one of the MSTest, NUnit or MBUnit tasks so Bamboo can get and display the test results. You can specify a custom results location if your project directory doesn't use the conventional structure.

See [Configuring a test task](#) for details.

5. Go!

Enable the plan, and click **Create**.

You should see the plan run. The 'Plan Summary' tab will report whether the build succeeded or not.

Tests in the appropriate directory in the source code repository will be run automatically as part of the build, and the test results will be displayed in Bamboo.

Now, whenever you commit a change to the repository, Bamboo will build your source code and report on your test results.

Get feedback

Bamboo displays a summary of the results of the build on the dashboard.

You can get further information about the build in the following ways:

- Build results for one or more plans can be displayed on a wallboard.
- You can get notifications about build results sent to you by email, IM and RSS feed.
- You can get build statistics about plans, and about developers contributing code to the build.
- You can drill down into the results to see the code changes that triggered the build, and the tests that were run for that build.

See [Getting feedback](#) for details.

Getting started with PHP and Bamboo

This page is a stub.

If you would like to see this page improved, please vote for this issue: [BAM-10947 - Create documentation for Getting Started with PHP and Bamboo](#)

Getting started with Ruby and Bamboo

This page is a stub.

If you would like to see this page improved, please vote for this issue: [BAM-10948 - Create documentation for Getting Started with Ruby and Bamboo](#)

Using the Bamboo dashboard

The dashboard is your Bamboo 'home' page. The dashboard has three tabs:

- **My Bamboo**— a convenient summary of information that is relevant to you (only appears if you have logged in to Bamboo):
 - plans that you have nominated as your favourites.
 - your latest build results (i.e. builds that were triggered by your latest [code changes](#)).
 - a summary of your [build statistics](#) (only appear if your Bamboo User Profile has been [associated with your Author Name](#)).
- **All Plans** — a list of plans and each plan's latest [build result](#).
- **Current Activity** — Bamboo's agents and build queue, showing which plans Bamboo is currently building and which plans are waiting to be built.

You can return to the dashboard from anywhere in Bamboo by clicking **Dashboard** in the top navigation menu.

On this page:

- [Viewing the dashboard](#)
- [Filtering the plans](#)
- [Working with favourites](#)

Related pages:

- [Configuring plans](#)
- [Working with builds](#)
- [Getting feedback](#)

Viewing the dashboard

You can:

- click the **project name** (e.g. 'Bamboo Testing') to view the plans in the project.
- click the **plan name** (e.g. 'Acceptance Test JDK 1.6') to [view the plan details](#).
- click the **build number** (e.g. '7823') to [view the build result](#).
- click the **author's name** to [view the author's details](#) (the author is the person who [triggered](#) the build by checking-in code).

The icon next to a build number indicates the plan's current status:



This plan's latest build was successful.



This plan's latest build failed.



Bamboo is currently checking-out the source-code for this plan, in preparation for starting a build.



Bamboo is currently queuing a build for this plan in the Build Queue.



Bamboo is currently executing a build for this plan.



The plan is stopped at a [manual stage](#).



The plan was not built, perhaps because the build was manually stopped.



This plan has been disabled.

Screenshot: Bamboo dashboard - 'All Plans' tab

Project	Plan	Build	Completed	Tests	Reason
+core+ Bamboo	CI Tests	#8565	11 hours ago	2 of 1740 failed	Changes by Marcin Gardias
	Deploy	#965	1 hour ago	No tests found	Changes by 3 people
	Foo	#14	14 hours ago	No tests found	Scheduled
	JDK 7 Acceptance Test	#569	2 days ago	2452 passed	Scheduled
	OnDemand CI Tests	#224	2 months ago	1991 passed	Changes by Przemek Bruski
	OnDemand CI Tests - against OnDemand	#1	3 months ago	176 passed	First build for this plan
	OpenJDK7 CI Tests	#1721	2 hours ago	1 of 1250 failed	Scheduled
	REST Docs	#210	3 weeks ago	No tests found	Changes by Krystian Brazulewicz
	Sanity Checks	#8	5 days ago	No tests found	Manual build by Przemek Bruski
	Stable CI Tests	#1130	11 hours ago	8 of 1821 failed	Changes by Marcin Gardias
+core+ Bamboo Databases Tests	WebDriver Tests	#6463	11 hours ago	1 of 1 failed	Changes by Marcin Gardias
	MSSQL 2005	#155		No tests found	Scheduled
	MySQL	#771	7 hours ago	282 passed	Scheduled with changes by 3 people

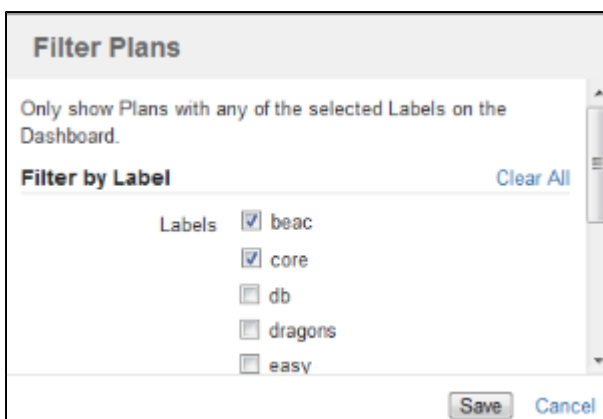
Filtering the plans

You can filter the plans on your dashboard according to plan labels. For instructions on how to add a label to a plan, see [Labelling a plan](#).

To filter the dashboard plans by label:

1. Navigate to the dashboard.
2. Click the  button. If the plan already has labels, they will be displayed next to the button, otherwise the button will read **Filter Plans**.
3. In the 'Filter Plans' dialog, select the labels to filter by.
4. Click **Save**. The dashboard will refresh, showing only the plans with the selected labels.

Screenshot: Filtering plans on a dashboard



Working with favourites

The **My Bamboo** tab lists your *favourite* plans — that is, the plans you work with the most. You can easily add and remove plans from your favourites.

When you add a plan to your favourites, you become a 'watcher' of the plan. This means that you will receive notifications about the build results for your favourite plans, depending on how your administrator has configured each plan's [notifications](#). You can receive notifications by email, Instant Messaging (IM) and RSS feed.

To add a plan to your favourites:

1. Click **Dashboard** in the top navigation bar, to display the dashboard.
2. Click the **All Plans** tab. This will display a list of all plans in your Bamboo system.
3. Locate the plan and click the grey star icon at the right.

Viewing Bamboo's agents

A Bamboo *agent* is a service that can run [job builds](#). There are two types of Bamboo agents:

- *local agents* run as part of the Bamboo server.
- *remote agents* run on computers, other than the Bamboo server, that run the [remote agent](#) tool. An [elastic agent](#) is a remote agent that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#).

 Local agents run in the Bamboo server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.

To view agents which are currently active, see [Using the Bamboo dashboard](#).

On this page:

- [Viewing all available Bamboo agents](#)
- [Viewing a specific Bamboo agent as a Bamboo user](#)
- [Viewing a specific Bamboo agent as a Bamboo administrator](#)

Related pages:

- [Configuring agents](#)
- [Bamboo remote agent installation guide](#)

Viewing all available Bamboo agents

To view all of Bamboo's available agents:

1. Click **Dashboard** in the top navigation bar to display the [Dashboard](#).
2. Click the **Current Activity** tab of the Dashboard.
3. Click **X of Y online agents are building** in the 'Building' section of the page (where X is the number of online agents that are currently building Bamboo plans and Y is the total number of available online agents). A list of all agents in your Bamboo system will be displayed:

Agents

An agent is a service that runs Bamboo builds. Below are all the agents available to this installation.

Local Agents

Local agents run on the Bamboo server.

Agent	Status
Core 1	 Building - JIRAFUNC-JDK16-4572
Core 2	 Building - JIRAFUNC-JDK16BATCH09-2295
Core 3	 Building - JIRAFUNC-JDK16BATCH07-2148
Unit Test Core	 Building - BASIC-JDK15-7634

Remote Agents

Remote agents run on computers other than the Bamboo server.

Online Remote Agents

Offline Remote Agents

There are currently 35 remote agents online (20 non-elastic and 15 elastic). A maximum of 100 agents are [supported](#) by your license.

Agent	Status
atlassian15 Agent 1	 Building - JRAMVN-COMPINST-220
atlassian15 Agent 2	 Building - JRAMVN-COMPSTD-209
atlassian15 Agent 3	 Building - SEL-ENTFFOX35BATCH01-968

Viewing a specific Bamboo agent as a Bamboo user

To view a specific agent as a standard Bamboo user:

1. Click **Dashboard** in the top navigation bar to display the [Dashboard](#).
2. Click the **Current Activity** tab of the Dashboard.
3. Click the name of the agent you wish to view in the 'Building' section of the page. To access the name of the agent, you can either:
 - Click its name in the list of builds as shown in the following screenshot:

My Bamboo All Plans **Current Activity**

Building [28 of 32 online agents are building](#)

JIRA Functional Tests - HEAD - Enterprise JDK1.6 building on Core 2	Building for 42 mins. 161 mins remaining
JIRA - MAVEN 2 - Standalone Func Tests building on Elastic Agent on i-614f6d0a	Building for 33 mins. 230 mins remaining

- Click **X of Y online agents are building** in the 'Building' section of the page, as described above, then click the agent's name on the page that is displayed:

Agents

An agent is a service that runs Bamboo builds. Below are all the agents available to this installation.

Local Agents

Local agents run on the Bamboo server.

Agent
Core 1
Core 2
Core 3
Unit Test Core

Remote Agents

Remote agents run on computers other than the Bamboo server.

Online Remote Agents Offline Remote Agents

There are currently 29 remote agents online (14 non-elastic and 15 elastic). A maximum of 100 agents are allowed.

Agent	Status
atlassian15 Agent 1	Building - SEL-
atlassian15 Agent 2	Building - JIRA
atlassian15 Agent 3	Building - JIRA
CentOS5X-x8632 (4367)	Idle

The details of the selected agent will be displayed:

Agents > Core 2 (local)

Information

Current status:  [Building - JIRAFUNC-JDK16-4573](#)

86%
5000 / 5819 Successful

Recent Builds

Executable Plans

Capabilities

▼ Showing Last 25 builds

Recent Builds

Status	Build Number	Reason	Completed	Duration	Test Results
✓	PLUGINS-SUBPLUGHEAD-3178	Dependant of BASIC-JDK16-9236	53 minutes ago	2 minutes	279 passed
✓	SEL-ENTFFOXTRIGGER-1413	Dependant of BASIC-JDK16-9236	56 minutes ago	10 seconds	No tests found
✓	DEPLOY-TRUNK-15	Manual build by Andreas Knecht	58 minutes ago	7 minutes	No tests found
✓	BASIC-JDK16-9235	Updated by Scott Harwood	1 hour ago	6 minutes	12111 passed
✓	JIRAPROFTEST-WEBSPPHEREJSPBRANCH-492	Scheduled build	4 hours ago	6 minutes	No tests found
✓	JIRAPROFTEST-WEBLOGICJSPHEAD-531	Scheduled build	5 hours ago	8 minutes	No tests found
✓	GH-DEF-782	Updated by Jean-Christophe Huet	7 hours ago	1 minute	42 passed

Click on the **Executable Plans** tab to view the [plans](#) that this agent can build:

Recent Builds

Executable Plans

Capabilities

Executable Plans

Plans whose requirements match the capabilities.

Plan
1. Deploy JIRA to Maven - Crowd Embedded JDK1.6
2. Deploy JIRA to Maven - JDK1.6 Branch
3. Deploy JIRA to Maven - JDK1.6 Trunk
4. Failing Build Debug - Unit Test Clone 2
5. GreenHopper - Default JDK1.5
6. JIRA Code Coverage - Functional + Unit Test Coverage JDK1.6
7. JIRA Code Coverage - Unit Test Coverage JDK1.6
8. JIRA Crowd Embedded - JDK1.6 Func - bundled plugins disabled
9. JIRA Crowd Embedded - JDK1.6 Func Batch 01
10. JIRA Crowd Embedded - JDK1.6 Func Batch 02
11. JIRA Crowd Embedded - JDK1.6 Func Batch 03
12. JIRA Crowd Embedded - JDK1.6 Func Batch 04

Click on the **Capabilities** tab to view the [capabilities](#) of this agent:

Recent Builds
Executable Plans
Capabilities

Custom

'Custom' capabilities are key-value pairs that define particular characteristics of an agent (e.g. 'operating.system=WindowsXP', 'fast.builds=true'). For an agent to be able to build a plan, both the 'Key' and 'Value' must match the plan's requirements.

Key	Value	Operations
container	false	View
elastic	false	View
func.test.runner	true	View
local	true	View
maven1.clover2.license	/import/tools/clover/clover.license	View
maven1.clover2.plugin	true	View
resin3x.installation	true	View
unit.test.runner	true	View
weblogic92.installation	true	View
websphere61.installation	true	View

Builder

'Builder' capabilities define the builders which are available to your build plans.

Builder Label	Path	Operations
Bash (Command)	/bin/bash	View
Maven 1 (Maven)	/import/tools/maven/maven-1.0.2	View
Maven 2 (Maven 2.x)	/opt/java/tools/maven2/current	View
Maven 2.1.0 (Maven 2.x)	/import/tools/maven/apache-maven-2.1.0	View
Script (Script)		View

JDK

'JDK' capabilities define the JDKs which are available to your build plans.

JDK Label	Java Home	Operations
JDK	/opt/java/sdk/1.6	View
JDK 1.4	/opt/java/sdk/1.4	View
JDK 1.5	/opt/java/sdk/1.5	View
JDK 1.6	/opt/java/sdk/1.6	View
Special JDK 1.6	/home/jira2-bamboo-j2ee/testjdk/jdk-6	View

Viewing a specific Bamboo agent as a Bamboo administrator

To view a specific agent as a Bamboo administrator:

1. Click **Dashboard** in the top navigation bar. This opens the [Dashboard](#).
2. Click the **Current Activity** tab of the Dashboard.
3. Click the name of the agent you wish to view in the 'Building' section of the page. To access the name of the agent, you can either:
 - Click its name in the list of builds:

My Bamboo
All Plans
Current Activity

Building

28 of 32 online agents are building

JIRA Functional Tests - HEAD - Enterprise JDK1.6 building on Core 2	Building for 42 mins. 161 mins remaining	
JIRA - MAVEN 2 - Standalone Func Tests building on Elastic Agent on i-614f6d0a	Building for 33 mins. 230 mins remaining	

- Hover over the **X of Y online agents are building** link and in the popup, click on the agent's name:

My BambooAll PlansCurrent Activity

Building

29 of 32 online agents are building

Recently Built

JIRA Functional Tests - HEAD - Enterprise JDK1.6 building on Core 2

JIRA - MAVEN 2 - Standalone Func Tests building on Elastic Agent on i-614f6d0a

JIRA Selenium Tests - HEAD - Firefox 3.5 building on Elastic Agent on i-654f6d0e

JIRA Crowd Embedded - JDK1.6 Func Tests building on Core 1

JIRA TPM - HEAD - Windows Vista, Intel 32bit x86, JDK16, J Standalone - Batch 4 building on wwisux8632msie7 (165)

JIRA TPM - HEAD - Windows Vista, Intel 32bit x86, JDK16, J Standalone - Batch 5 building on wwisux8632msie7 (166)

JIRA TPM - HEAD - Windows Vista, Intel 32bit x86, JDK16, J Standalone - Batch 2 building on wwisux8632msie7 (168)

JIRA TPM - HEAD - Windows Vista, Intel 32bit x86, JDK16, J Standalone - Batch 3 building on wwisux8632msie7 (167)

JIRA TPM - HEAD - Windows Vista, Intel 32bit x86, JDK16, J Standalone - Batch 1 building on wwisux8632msie7 (169)

JIRA Functional Tests - HEAD - JDK1.6 Batch 05 building on Elastic Agent on i-b74f6ddc

Online Agents

There are currently 36 remote agents online (21 non-elastic and 15 elastic). A maximum of 100 agents are supported by your license.

Core 2

Building - JIRAFUNC-JDK16-4573

Disable

Core 3

Idle

Disable

Daidalos

Idle

Disable

Elastic Agent on i-614f6d0a

Building - JRAMVN-TESTFUNC-121

Disable

Elastic Agent on i-634f6d08

Building - SEL-ENTFFOX358BATCH04-1156

Disable

Elastic Agent on i-654f6d0e

Building - SEL-ENTFFOX358BATCH05-96

Disable

Enable all

Disable all

Linux CentOS 5.X, JDK16, Jira Standalone, Batch 5 build

JDK16PSQ83BATCH05-96

Build took 83 minutes.

Tests - HEAD - JDK1.6 AFUNC-2218 was successful

IC-JDK16-9237, Build

Tests - HEAD - JDK1.6 AFUNC-2331 was successful

IC-JDK16-9237, Build

Tests - HEAD - Firefox 3.5

BATCH11-733 was

- The details of the selected agent will be displayed in the Bamboo administration console, where you can further configure this agent and its capabilities.

Agents > Core 2 (local)

Information

Current status:

Building - JIRAFUNC-JDK16-4573

86%

5060 / 5819 Successful

[View Recent Activity](#) | [Edit Details](#) | [Disable](#) | [Request Agent Stop](#) | [Delete](#)

Capabilities

Executable Plans

System Properties

Agent-Specific Capabilities

Add Capability

A capability is a feature of an agent. There are 3 types of capabilities: builders, JDKs and custom.

The following capabilities are specific to this agent:

Custom

Add Custom

'Custom' capabilities are key-value pairs that define particular characteristics of an agent (e.g. 'operating.system=WindowsXP', 'fast.builds=true'). For an agent to be able to build a plan, both the 'Key' and 'Value' must match the plan's requirements.

Key	Value	Operations
func.test.runner	true	View Edit Delete

Shared Capabilities

Edit

The following 'shared' capabilities are inherited by this agent. Note that the value of a shared capability is overridden by the value of an agent-specific capability with the same key/label (if one exists).

Custom

Key	Value	Operations
container	false	View Edit
elastic	false	View Edit
local	true	View Edit
maven1.clover2.license	/import/tools/clover/clover.license	View Edit
maven1.clover2.plugin	true	View Edit
resin3x.installation	true	View Edit
unit.test.runner	true	View Edit
weblogic92.installation	true	View Edit
websphere61.installation	true	View Edit

Click on the **Executable Plans** tab to view the plans that this agent can build:

Executable Plans

Plans whose requirements match the capabilities.

Plan	Last built
1. Deploy JIRA to Maven - Crowd Embedded JDK1.6	✓ DEPLOY-CROWDEMBEDDED-37 2 weeks ago
2. Deploy JIRA to Maven - JDK1.6 Branch	✓ DEPLOY-BRANCH-24 1 month ago
3. Deploy JIRA to Maven - JDK1.6 Trunk	✓ DEPLOY-TRUNK-15 1 hour ago
4. Failing Build Debug - Unit Test Clone 2	Never built
5. GreenHopper - Default JDK1.5	✓ GH-DEF-782 7 hours ago
6. JIRA Code Coverage - Functional + Unit Test Coverage JDK1.6	Never built
7. JIRA Code Coverage - Unit Test Coverage JDK1.6	✓ COV-UNIT-792 9 hours ago
8. JIRA Crowd Embedded - JDK1.6 Func - bundled plugins disabled	❗ JIRACROWD-FUNC-150 19 hours ago (Last successful build JIRACROWD-FUNC-149 , 3 days ago)
9. JIRA Crowd Embedded - JDK1.6 Func Batch 01	Never built
10. JIRA Crowd Embedded - JDK1.6 Func Batch 02	Never built
11. JIRA Crowd Embedded - JDK1.6 Func Batch 03	Never built
12. JIRA Crowd Embedded - JDK1.6 Func Batch 04	❗ JIRACROWD-FB04-35 3 weeks ago (Never built successfully)

Click on the **System Properties** tab to view the system properties of this agent:

Capabilities	Executable Plans	System Properties																				
<table><tr><td>System Date:</td><td>Tuesday, 25 May 2010</td></tr><tr><td>System Time:</td><td>22:37:13</td></tr><tr><td>Up Time:</td><td>5 days, 2 hours, 4 minutes, 10 seconds (since Thu May 20 20:33:08 CDT 2010)</td></tr><tr><td>Username:</td><td>bamboo-agent</td></tr><tr><td>User Timezone:</td><td>America/Chicago</td></tr><tr><td>User Locale:</td><td>English (United States)</td></tr><tr><td>System Encoding:</td><td>ANSI_X3.4-1968</td></tr><tr><td>Operating System:</td><td>Linux 2.6.18-164.15.1.el5.028stab068.9</td></tr><tr><td>Operating System Architecture:</td><td>i386</td></tr><tr><td>Available Processors:</td><td>4</td></tr></table>			System Date:	Tuesday, 25 May 2010	System Time:	22:37:13	Up Time:	5 days, 2 hours, 4 minutes, 10 seconds (since Thu May 20 20:33:08 CDT 2010)	Username:	bamboo-agent	User Timezone:	America/Chicago	User Locale:	English (United States)	System Encoding:	ANSI_X3.4-1968	Operating System:	Linux 2.6.18-164.15.1.el5.028stab068.9	Operating System Architecture:	i386	Available Processors:	4
System Date:	Tuesday, 25 May 2010																					
System Time:	22:37:13																					
Up Time:	5 days, 2 hours, 4 minutes, 10 seconds (since Thu May 20 20:33:08 CDT 2010)																					
Username:	bamboo-agent																					
User Timezone:	America/Chicago																					
User Locale:	English (United States)																					
System Encoding:	ANSI_X3.4-1968																					
Operating System:	Linux 2.6.18-164.15.1.el5.028stab068.9																					
Operating System Architecture:	i386																					
Available Processors:	4																					
Network																						
<table><tr><td>Host Name:</td><td>bamboo-agent-90.private.atlassian.com</td></tr><tr><td>IP Address:</td><td>172.16.3.90</td></tr></table>			Host Name:	bamboo-agent-90.private.atlassian.com	IP Address:	172.16.3.90																
Host Name:	bamboo-agent-90.private.atlassian.com																					
IP Address:	172.16.3.90																					
Memory Statistics																						
<table><tr><td>Total Memory:</td><td>62 MB</td></tr><tr><td>Free Memory:</td><td>35 MB</td></tr><tr><td>Used Memory:</td><td>27 MB</td></tr></table>			Total Memory:	62 MB	Free Memory:	35 MB	Used Memory:	27 MB														
Total Memory:	62 MB																					
Free Memory:	35 MB																					
Used Memory:	27 MB																					
Bamboo Paths																						
<table><tr><td>Current running directory:</td><td>/opt/bamboo-agent/bin</td></tr><tr><td>Build Working Directory:</td><td>/opt/bamboo-agent/xml-data/build-dir</td></tr><tr><td>Bamboo Home:</td><td>/opt/bamboo-agent</td></tr><tr><td>Temporary directory:</td><td>/tmp</td></tr></table>			Current running directory:	/opt/bamboo-agent/bin	Build Working Directory:	/opt/bamboo-agent/xml-data/build-dir	Bamboo Home:	/opt/bamboo-agent	Temporary directory:	/tmp												
Current running directory:	/opt/bamboo-agent/bin																					
Build Working Directory:	/opt/bamboo-agent/xml-data/build-dir																					
Bamboo Home:	/opt/bamboo-agent																					
Temporary directory:	/tmp																					
File System Statistics																						
<table><tr><td>Disk space free (for bamboo.home):</td><td>11 GB</td></tr></table>			Disk space free (for bamboo.home):	11 GB																		
Disk space free (for bamboo.home):	11 GB																					

Using Bamboo

Bamboo is a [continuous integration](#) (CI) server. Bamboo assists software development teams by providing:

- automated building and testing of software source-code status.
- updates on successful/failed builds.
- reporting tools for statistical analysis.

This user guide has information about using Bamboo. Please see [Administering Bamboo](#) for information about managing the Bamboo server itself.

Using

[Configuring plans](#)

[Connecting to code repositories](#)

[Jobs and tasks](#)

[Working with builds](#)

[Getting feedback](#)

[Integrating Bamboo with Atlassian applications](#)

[Managing your user profile](#)

Installing

[Bamboo installation guide](#)

[Bamboo EAR-WAR installation guide](#)

[Bamboo installation guide for Linux](#)

[Bamboo installation guide for Mac](#)

[Bamboo installation guide for Windows](#)

[Connecting Bamboo to an external database](#)

[Bamboo remote agent installation guide](#)

[Supported platforms](#)

See also

[Getting started](#)

[Administering Bamboo](#)

[Release notes](#)

[Bamboo security advisories](#)

[Bamboo documentation downloads](#)

Deployment projects

What are deployment projects?

A deployment project is a container for holding the software project you are deploying: releases that have been built and tested, and the environments to which releases are deployed. Teams typically have QA, staging and product environments.

Why use deployment projects?

Continuous Integration was not designed for Continuous Delivery. Continuous Integration is designed to keep developers informed about the state of the latest code changes.

In Continuous Integration, historical build results (along with information such as issue and commits) are de-emphasised as more changes are made, since only the latest build is important to the developer.

Using a traditional Continuous Integration server for Continuous Delivery is less than ideal because:

- **Deployed builds are difficult to find** – Builds that were deployed only a few days ago are de-emphasised by the system. While this is good for a Continuous Integration workflow, the behaviour makes it difficult for team members to identify which builds have been deployed and when, versus which have not. Teams can work around this with a system that uses labelling but it's not immediately obvious how it should work unless team members are trained to use it correctly.
- **Difficult to find what changes were made between deployments** – Build results report commit and issue data between a specific build result and the one immediately before it. There can be many build results between two different deployments. Often the entire history has to be navigated between the two deployments to build a complete view of the changes between them. Sometimes, even other tools have to be used, such as version control systems.
- **Difficult to know what was deployed, and when and where it was deployed** – Builds do not have visibility of where code is deployed or what was previously deployed to an environment.
- **Lack of insight into the QA process** – Given a list of deployment candidates, builds offer no clear way (other than commenting or labelling) for QA to 'sign off' on a tested release or mark a release as broken or un-releasable.
- **Poor control over who can deploy** – While it can be controlled by permissions who can run, view or edit a build, they do not give enough fine grained control over which people in the team can deploy to production or other sensitive environments. In essence, if

someone has permission to run the build they can deploy the software any time they wish.

To solve these issues Bamboo provides the following concepts:

- **Deployment project** – Represents the software you are deploying (such as a web application), the releases of the software deployed and the environments that they will be deployed to throughout the lifecycle.
- **Environment** – Represents the servers or groups of servers where the software release has been deployed to, and the tasks that are needed for the deployment to work smoothly. Example environments could be named Development, QA, Staging or Production. Environments have permissions that allow fine grained control of who can deploy, edit or view an environment and record the full history of releases deployed to it.
- **Release** – Identifies a snapshot of artifacts and its associated data such as commits, JIRA issues and the builds that were used to test it. As a release contains the information of the difference between itself and the release beforehand, it's very easy to see the changes between releases or to show the difference between the software deployed on two different environments. Releases also track what environments they have been deployed to.

How do deployment projects work?

Consider the following diagram:

On this page:

- [What are deployment projects?](#)
- [Why use deployment projects?](#)
- [How do deployment projects work?](#)

Related pages:

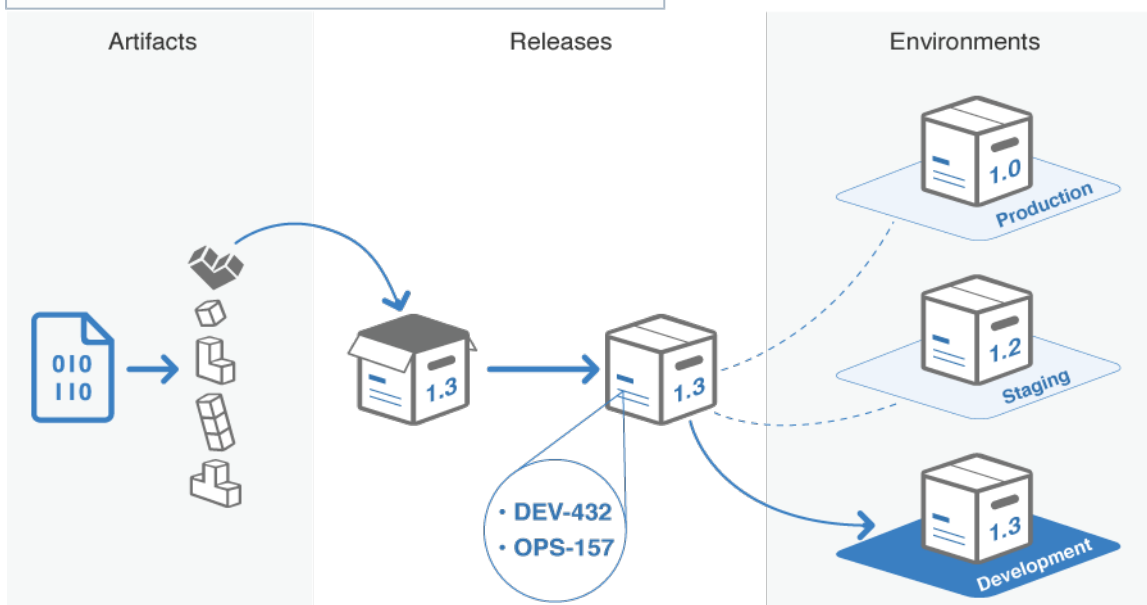
- [Understanding deployment releases](#)
- [Deployment projects workflow](#)
- [A sample deployment project](#)
- [Creating and configuring a deployment project](#)
- [Creating a deployment environment](#)
- [Managing deployment projects](#)
- [Manually starting a deployment](#)



What is Continuous Delivery?

Continuous Delivery is the practice where all changes made to a software project are automatically built, tested and made ready for deployment to users. In practice, once the project has been built and tested it is 'staged' somewhere where it can be manually verified and then made available to users.

Unlike Continuous Deployment (the process where code changes are automatically built, tested and deployed without human intervention), typically there is a decision made by a human being to whether or not the software is of sufficient quality or if it is the correct time for the business to make the software available to its users.



Artifacts

Create and test deployable artifacts with build plans. Ensure any artifacts you wish to deploy with Bamboo are flagged as "shared" to make them available to the deployment instructions of the environment.

Releases

Any artifact that has been successfully tested can be used to create a release; you can create as many releases as you like. Bamboo will add other metadata such as related commits and JIRA issues to each release which enable reporting and tracking as it moves through your environments.

Environments

Environments in Bamboo reflect the development, testing and production environments in your IT infrastructure – hostnames and authentication credentials for each environment reside at the task level inside your deployment jobs. At any point in time, you will be able to see which release is running in each environment, which release it replaced, when it was deployed and who deployed it. You will also be able to see any associated JIRA issues.

Understanding deployment releases

Key to getting the most out of deployment projects is understanding what releases are, and how you should be using them.

It is also important to understand the difference and relationship between 'artifacts' - the results of a build plan - and 'releases' - a snapshot of artifacts at a specific time that can be deployed somewhere.

On this page:

- [What are artifacts?](#)
- [What are deployment releases?](#)
- [Why use releases?](#)
- [How artifacts and deployment releases work together](#)
- [The next step](#)

What are artifacts?

When the continuous integration process is triggered by a developer committing code, the first stage of the process compiles the code, runs tests and then assembles the code into binaries. These assembled binaries are known as 'artifacts'. The build process can produce build artifacts at any stage of the build that can then be shared with other builds or deployment projects.

Since Bamboo manages artifacts, any artifacts that are needed by builds or deployments are automatically transferred by Bamboo to a remote server as needed, so long as that build or deployment project declares that it needs the artifact to complete its work.

For more information, see [Configuring artifact sharing between jobs](#).

What are deployment releases?

Releases are used to track exactly what software was deployed to an environment. In essence, a release is a snapshot of any number of artifacts that will be used in the deployment process and their associated metadata, such as JIRA issues, code changes and any test metadata that might be relevant to what is being deployed.

A release is created from the result of a single build. When you view a release, you can see all the code changes, JIRA issues and other metadata that were used when making the artifact for that build. This information can be used for purposes such as release notes, quality control and infrastructure planning, and allows you to compare any two releases to see the changes between them.

Why use releases?

In Bamboo, releases are tracked against environments, which represent a server or group of servers that you wish to deploy your software to. Because each environment can only host a single active release at any one time, Bamboo gives a unique release name to the software being deployed. By checking the environments for our project, we can quickly identify:

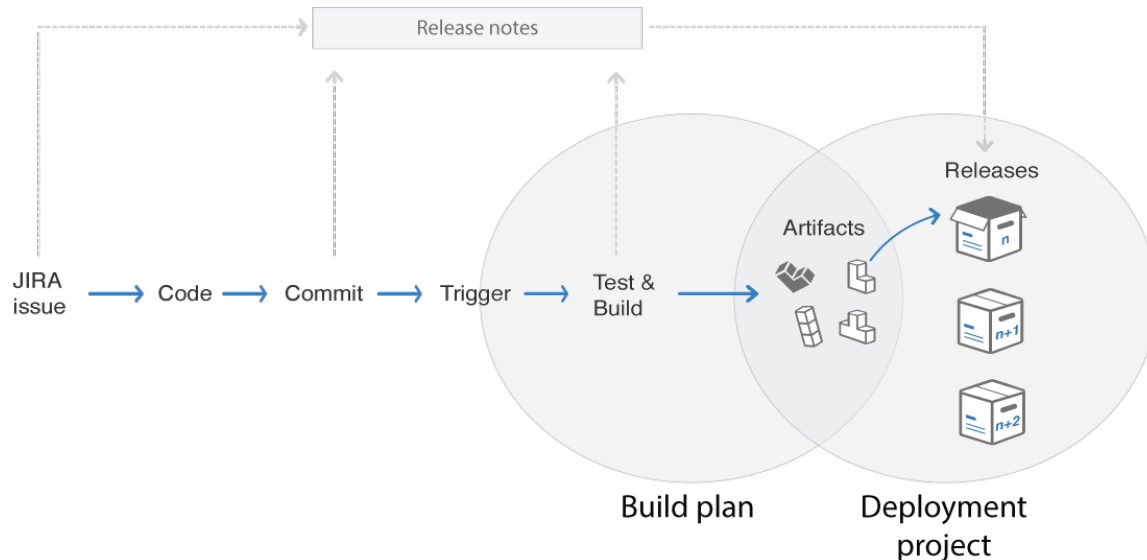
- Where releases have been deployed
- Which release is currently deployed
- The release deployment history

- The release deployment status.

Another key feature of releases is that as well as providing a deployable snapshot of your artifacts, they also collate the JIRA issues, commit record and test & build metadata for the specific series of changes associated with the release. This enables much smoother reporting and tracking as the release moves through your environments, and allows you to easily track changes between releases.

How artifacts and deployment releases work together

The relationship between artifacts and releases shows the 'hand-over point' between Bamboo builds and Bamboo deployments.



As the diagram shows, a developer who is responding to JIRA issues, commits a code change and triggers a build. This build produces a number of artifacts. In deployment, these artifacts are assembled into a release, and the JIRA issue, commits and test/build metadata are added. This release then gets a unique identification name which serves as an identifier throughout the system. You can define the unique identifier according to your needs via the [release naming system](#).

Once a release has been created, it is now ready to be deployed to an environment.

The next step

The next step is to examine and understand the deployment project workflow. Learn more about the [deployment project workflow](#).

Deployment projects workflow

Deployment projects are an important feature of the continuous deployment philosophy. Identifying and understanding the key configuration steps for a deployment project will help you to gain a better insight into how a deployment project functions.

On this page:

- [Deployment project prerequisites](#)
- [Step 1: Create a new deployment project](#)
- [Step 2: Decide on a release naming scheme](#)
- [Step 3: Decide who can view and edit the project](#)
- [Step 4: Create a deployment environment](#)
- [Step 5: Customize your deployment environment](#)
- [Step 6: Start deploying!](#)

Deployment project prerequisites

There are a number of prerequisites that must be in place before you can start using deployment projects. The prerequisites are:

1. A build plan
2. Artifacts to deploy (these are produced by the build plan and shared)

Step 1: Create a new deployment project

Creating the deployment project is the first step. Here we will give the project a name and a description, but most importantly we associate the deployment project with an existing Bamboo build plan. This is why we must have a build plan available to associate with our new deployment project.

Learn more about creating a deployment project [here](#).

Step 2: Decide on a release naming scheme

The next step is to configure the release naming scheme for the deployment project. The release naming scheme will define how Bamboo names the releases that you create from your build artifacts for deployment. You can use either a simple release naming scheme, or a scheme that uses global or plan variables already defined in Bamboo.

Learn more about release naming schemes [here](#).

Step 3: Decide who can view and edit the project

You need to decide who can view and edit the deployment project: This is done using the permission scheme. You can add or remove individuals or groups from the scheme, and give them access to either view and/or edit the project.

Learn more about the permissions scheme [here](#).

Step 4: Create a deployment environment

The next step is to create a deployment environment. A deployment environment represents the servers or groups of servers where the software has been deployed, and any tasks needed for the deployment to go smoothly. You can call the deployment environment anything you like, though typical names are QA, Staging and Production.

Learn more about creating a deployment environment [here](#).

Step 5: Customize your deployment environment

Once you have created your deployment environment, you need to set it up to reflect the needs of your project. You can control most aspects of the deployment environment, including:

- **Tasks** - Run executable tasks during the deployment process, for example downloading a needed artifact from a different plan
- **Triggers** - Decide which events or schedule points will trigger off deployment of your project to an environment
- **Permissions** - Decide who can view and edit your deployment environment
- **Agents** - Control which agents you will use to support your deployment process
- **Notifications** - Create a notification scheme to keep you informed about your deployment progress
- **Variables** - Assign variables for your deployment projects

Step 6: Start deploying!

Once you have set up your deployment project, you're ready to start the deployment process.

A sample deployment project

On this page we will examine a sample deployment project, and work through the steps required to get a deployment project up and running.

On this page:

- [Step 1: Create a deployment project](#)
- [Step 2: Define the release naming scheme](#)
- [Step 3: Create a deployment environment](#)
- [Step 4: Add some environment tasks](#)
- [Step 5: Let's deploy!](#)
- [Step 7: Additional deployment environment options](#)

Step 1: Create a deployment project

The first step in creating a deployment project is to associate the project with an existing build plan. This is done at the same time as creating the deployment project. To create a new deployment project, and associate an existing build plan with it:

1. Click **Create > Create deployment project** in the drop down menu from the header bar. The 'Setup deployment project' screen will appear.
2. Using the Search for a plan drop down menu, select an existing build plan. Bamboo will identify any relevant build plans for you in the menu:

Set up deployment project [How deployments work](#)

Source build plan*
for artifacts

Name*

Description

Search for a plan

- Imported from Jenkins > Atlassian HttpClient
- Imported from Jenkins > Build Pipeline ABC
- Imported from Jenkins > Build Pipeline XYZ
- JGit Flow > Checkstyle
- JGit Flow > Coverage (Clover)
- JGit Flow > Unit Tests
- JIRA Workflow Transitions > AMKT Auto-close
- JIRA Workflow Transitions > Resolve Issue - Merge and Close Issue Branch
- JIRA Workflow Transitions > Start Progress - Create Issue Branch
- Project Bonsai > Atlassian IRKD Continuous Integration**

used as the source for artifacts that will be deployed.

15 - 27 Jun 13

administrators

Atlassian

In this example we can see that the associated build plan is that for project Bonsai

3. Complete the Name and Description fields as required
4. Click **Create deployment project**. Your deployment project will be created, and will automatically be associated with the build plan you selected above.

Step 2: Define the release naming scheme

The next step is to provide a version naming strategy for the deployment project. This will define how the deployment project will ascribe names to current and subsequent artifact bundles that it generates. See [How deployment releases work](#) for more information. To configure your version naming scheme:

1. From the deployment project configuration screen, click **Release versioning** to display the deployment project release versioning screen:

Release versioning: Deployment for Project Bonsai [How release versioning works](#)

Specify what version Bamboo should assign to automatically created releases. You can override this manually whenever you create a new release, but you'll have to come here for any lasting changes to take effect.

Next release

Version* 1.0 [Generate preview](#)

What version should Bamboo use for the next release? e.g. 1.0-m1 or 1.0-\${bamboo.buildNumber}

[Add variable to version](#)

Automatically increment with each new release

If you can't find variables in the list below that you added to the version, press 'Generate preview'. Bamboo will automatically add a suffix to non-unique versions.

Numbers ☒ Last number in version

Variables No incrementable variables found in your version

Only plan variables and global variables are incrementable

Preview

Version of next release 1.0

Version of subsequent release 1.1

Why do some variables still show in the preview?

Bamboo can only substitute variables in the preview if they do not depend on other data. If a variable depends on other workflows (e.g. \${bamboo.buildNumber}), it will be replaced at runtime but not in the preview.

[Save](#) [Cancel](#)

2. Complete the required fields according to your naming scheme. In this example we can see that a simple naming scheme has been adopted - the next name will be 1.0, and the subsequent 1.1 etc
3. Click **Save** to save your naming scheme.

Step 3: Create a deployment environment

Once we have defined our naming scheme, we need to create a deployment environment for the artifact(s) to be deployed into. Typically, deployment environments include Test, Staging, QA and Production, however there's no limit to creating useful deployment environments. Lets look at how it's done:

1. From the deployment project configuration screen, click **Add environment** to open the Set up environment screen:

Set up environment for Deployment for Project Bonsai

How environments work

Environment name*

Dev Sandbox

e.g. Staging, QA, or Production

Description

Sandboxed development environmen

Continue to task setup

Create and back

Cancel

2. Enter the name of the deployment environment, and a brief description. In this example we will call our environment Dev Sandbox, and give it a suitable description
3. Click either **Create and back** or **Continue to task setup** to create the environment. The next stage will be to add some tasks, so clicking **Continue to task setup** will take us straight to the next step.

Step 4: Add some environment tasks

Tasks are activities that the deployment project will perform in order to run. These could be checking out some code from a repository, downloading an artifact from a server or running a script. Let's have a look at how to add a couple of tasks to the deployment environment:

1. If you continued to task setup from the environment creation process, then you will already be at the Set up tasks screen. alternatively, From the deployment project configuration screen, click **Set up tasks**:

Set up tasks: Dev Sandbox

How deployment tasks work

What are the tasks that need to happen to make this deployment a success

3 agents and 5 elastic images have the capabilities to run this deployment

No tasks configured

Add task

Finish deployment project

2. Click **Add task** to display the list of tasks that are available to you:

Task types

Search

All

Builder

Deployment

Source Control

Reload Tomcat Application

Reloads a Web Application running on a Tomcat server

Script

Execute a script as part of your build by referencing a file or writing your script inline

SSH Task

Run a remote command over SSH

Stop Tomcat Application

Stops a Web Application running on a Tomcat server

SCP Task

Copy files to a remote server using SCP

Source Code Checkout

Checkout from a repository

Start Tomcat Application

Starts a Web Application running on a Tomcat server

Undeploy Tomcat Application

Undeploys a Web Application running on a Tomcat server

Get more tasks on the [Atlassian marketplace](#) or [write your own](#)

Cancel

In this example, we will add a simple script task to run as part of our build. Clicking on the task we wish to add adds it to the set up tasks screen, and allows us to configure the individual task:

Update tasks: Dev Sandbox

How deployment tasks work

What are the tasks that need to happen to make this deployment a success

3 agents and 5 elastic images have the capabilities to run this deployment

Script

Final tasks Are always executed even if a previous task fails

Drag tasks here to make them final

Add task

Script configuration

Task description

Echo task

☐ Disable this task

Script location

Inline

Script body*

```
1 echo "staging complete"
```

(Windows: .bat file; Unix: /bin/sh compatible script)

- Click **Save** to save the individual task configuration, and then on **Finish deployment project** to complete configuration of the script task for the deployment environment. In reality, we would require a number of tasks, not least one to obtain an artifact for use in the deployment. The following task configuration for a production environment includes an artifact download, DB change script, a Tomcat deployment, source code checkout and a Maven 3.x task:

Update tasks: Production

How deployment tasks work

What are the tasks that need to happen to make this deployment a success

3 agents have the capabilities to run this deployment

Artifact download

Download version contents

Script

Apply DB changes

Deploy Tomcat Application

Ship it!

Source Code Checkout

Maven 3.x

Run smoke tests against new deploy

Final tasks Are always executed even if a previous task fails

Drag tasks here to make them final

Add task

No task selected

Select a task from the list on the left to configure it.

Back to deployment project

Step 5: Let's deploy!

Our sample deployment project now has all of the elements required to run. We can trigger the deployment project manually by clicking on the appropriate deploy icon on the projects page:

All Deployment Projects

Atlassian IRKD Microsite 

Environment	Status	Release	Completed	Actions
Dev Sandbox	DEPLOYED	1.3_RC4	05 June 2013 10:06 PM	 
Production	DEPLOYED	1.2_RC3 APPROVED	03 June 2013 10:56 PM	 
QA Env.	DEPLOYED	1.3_RC2 BROKEN	04 June 2013 11:11 PM	 

Deployment for Project Bonsai 

A test deployment project for Bonsai

Environment	Status	Release	Completed	Actions
Dev Sandbox	Never deployed			 

Step 7: Additional deployment environment options

But deployments don't end here. This simple example is just a snapshot of how a deployment project is configured and works. Bamboo deployment projects feature a host of additional features to help you manage your development and deployment processes. These include:

- **Automated triggering** - choose to automatically deploy after a successful build plan completes, or at a scheduled time
- **Agents** - Assign specific agents, elastic agents or image configurations to execute the deployment for the environment
- **Variables** - Incorporate variables for use when deploying versions to environments
- **Permissions** - Define what users are allowed to view, edit and deploy in the environment
- **Notifications** - Define who and how notifications about events for the environment are made.

Creating and configuring a deployment project

Creating a deployment project from a plan is easy with Bamboo.

A deployment is a container that holds:

- Environments that represent the physical environments, such as QA, Staging and Production
- Releases which represent the actual software artifacts being deployed - these include the issues and commits which make up the release.

To create a new deployment project you will need to:

1. Provide a name and a description that represents your project
2. Associate the project with a Plan that produces the artifacts you will snapshot in a release and deploy to the environment.

On this page:

- [Creating a new deployment project](#)
- [Editing the details of an existing deployment project](#)
- [Configuring release naming](#)
- [Configuring deployment project permissions](#)

Related pages:

- [How deployment releases work](#)

Creating a new deployment project

To create a new deployment project:

1. Click on **Create > Create deployment project** in the drop down menu from the header bar. The 'Setup deployment project' screen will appear:

Set up deployment project
[How deployments work](#)

Source build plan*
for artifacts
Project Bonsai > Atlassian IRKD Continuous Integration

Name*
Project Bonsai Deployment Test

Description
A test deployment project for Bonsai

Create deployment project
Cancel

- Complete the 'Setup deployment project' screen using the following fields:

Field	Description	Optional?
Related Build Plan	The name of the plan you wish to associate with the deployment project Hint: This field identifies the source of your deployment artifacts	
Name	The name of your deployment project	
Description	A brief description of your deployment project	

- Click on the **Create deployment project** button. The deployment project configuration screen will display:

Deployment projects / Deployment for Project Bonsai

Configuration: Deployment for Project Bonsai

A test deployment project for Bonsai

Edit details
Add environment
Actions

What you want to deploy

Source build plan Project Bonsai > Atlassian IRKD Continuous Integration

Available artifacts IRKD microsite WAR, Clover Report (System), clover

Edit build plan
Release versioning
Project permissions

Add environment

Your deployment project has been created with the build plan relation, name and description you specified. It is now ready for configuration.

Editing the details of an existing deployment project

Bamboo allows you to edit the details of an existing deployment project.

To edit the details of an existing deployment project:

- From the deployment project configuration screen, click the **Edit details** button. The **Update deployment project** screen will display:

Update deployment project
[How deployments work](#)

Source build plan*
for artifacts
Project Bonsai - Atlassian IRKD Continuous Integration

Name*
Deployment for Project Bonsai

Description
A test deployment project for Bonsai

Save deployment project
Cancel

- Complete the 'Update deployment project' screen using the following fields:

Field	Description	Optional?

Related Build Plan	The name of the plan you wish to associate with the deployment project Hint: This field identifies the source of your deployment artifacts	
Name	The name of your deployment project	
Description	A brief description of your deployment project	

- Click on **Save deployment project** to save your changes.

Configuring release naming

Bamboo's release naming configuration allows you to control:

- What Bamboo will call the next release the deployment project generates
- Automatic incrementing of the release number each time a new release is created
- Automatic incrementing of the release number as specified by a global variable each time a new release is created.

To configure your release naming strategy:

- Click the **Release naming** button on the 'Deployment project configuration' screen. The 'Release naming' screen will display:

Release versioning: Deployment for Project Bonsai
[How release versioning works](#)

Specify what version Bamboo should assign to automatically created releases. You can override this manually whenever you create a new release, but you'll have to come here for any lasting changes to take effect.

Next release

Version ^{*} Generate preview

What version should Bamboo use for the next release? e.g. 1.0-m1 or 1.0-\${bamboo.buildNumber}

[Add variable to version](#)

Automatically increment with each new release

If you can't find variables in the list below that you added to the version, press 'Generate preview'. Bamboo will automatically add a suffix to non-unique versions.

Numbers ☒ Last number in version

Variables ☐ No incrementable variables found in your version

Only plan variables and global variables are incrementable

Preview

Version of next release 1.0

Version of subsequent release 1.1

Why do some variables still show in the preview?

Bamboo can only substitute variables in the preview if they do not depend on other data. If a variable depends on other workflows (e.g. \${bamboo.buildNumber}), it will be replaced at runtime but not in the preview.

- Complete the 'Release naming' screen using the following fields:

Field	Description	Optional?
Next release name	The identification for the next release version that Bamboo will create by replacing the <code>\${bamboo.buildNumber}</code> term with the numbering strategy of your choice. For example 1.0-m1.	

Automatically increment with each new release	Numbers: Check this box to automatically increment the last number in the release string. Variables: You may set Bamboo to increment release numbers according to a global variable. The global variables available will appear as part of the drop down menu in the 'Next release name field', however this field must be checked to activate global variable functionality. Available global variables will appear in the drop down menu when the 'Auto increment global variables' box is checked.	
Preview	A preview of the release naming strategy. Note: A preview may not always be available	

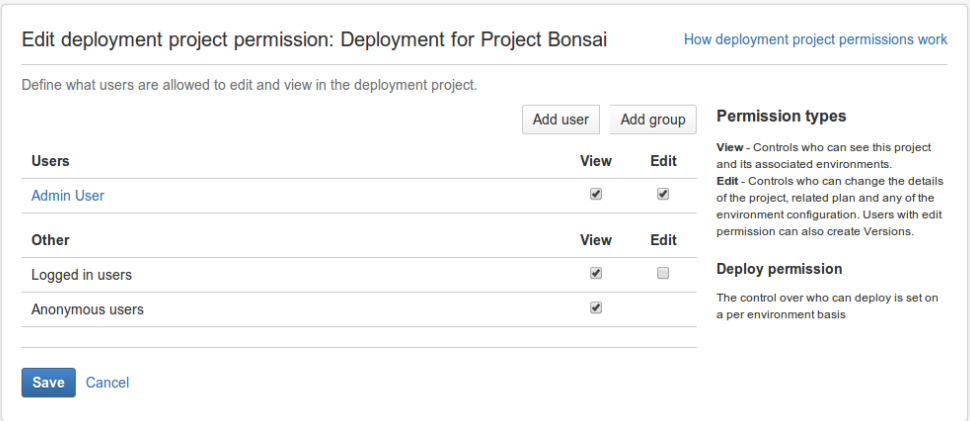
- Click **Save** to save your changes.

Configuring deployment project permissions

Bamboo gives you control over who has permission to **View** and **Edit** aspects of your deployment project.

To configure your permission strategy:

- Click the **Permissions** button on the 'Deployment project configuration' screen to display the 'Edit permissions' screen:



- Click the **Add User** or **Add Group** button to search for and add, users or groups
- Check the relevant 'View' and 'Edit' permission boxes to assign your desired permission scheme
- Click **Save** to save your permission scheme.

How deployment releases work

Bamboo provides a range of options that allow you to control your release naming scheme. You can specify how Bamboo handles release versioning, and control automatic incrementing between releases.

Bamboo allows you to use:

- Simple incremental numbering
- Advanced numbering based upon Bamboo variables

Bamboo also allows you to manually override automatic release settings when you create a new release.

On this page:

- [Simple release versioning](#)
- [Release versioning using variables](#)

Simple release versioning

Simple release versioning allows you to specify a starting release number, for example 1.0, which Bamboo will automatically increment. When using simple release versioning, Bamboo will increment the final number in the release name. For example:

Release name	Incremented release name
1	2
1.1	1.2

1.11	1.12
1.0.1	1.0.2

To configure simple release naming:

1. Click the **Release versioning** button on the 'Deployment project configuration' screen to display the 'Release versioning' screen:

Release versioning: Deployment for Project Bonsai

[How release versioning works](#)

Specify what version Bamboo should assign to automatically created releases. You can override this manually whenever you create a new release, but you'll have to come here for any lasting changes to take effect.

Next release

Version*

1.0

Generate preview

What version should Bamboo use for the next release? e.g. 1.0-m1 or 1.0-\${bamboo.buildNumber}
 [Add variable to version](#)

Automatically increment with each new release

If you can't find variables in the list below that you added to the version, press 'Generate preview'. Bamboo will automatically add a suffix to non-unique versions.

Numbers

☒ Last number in version

Variables

No incrementable variables found in your version

Only plan variables and global variables are incrementable

Preview

Version of next release

1.0

Version of subsequent release

1.1

Why do some variables still show in the preview?

Bamboo can only substitute variables in the preview if they do not depend on other data. If a variable depends on other workflows (e.g. \${bamboo.buildNumber}), it will be replaced at runtime but not in the preview.

Save

Cancel

2. Complete the 'Release versioning' screen using the following fields:

Field	Description	Optional?
Create release name	The identification for the next release name that Bamboo will create. In simple release naming, you should use something straightforward like 1.0.	
Automatically increment with each new release	Check the 'Numbers' check box to automatically increment the release number according to the 'Next release name' field, as defined above. If you leave this box unchecked, then no release number incrementing will occur.	
Preview	This field allows you to preview what the next release name will look like. To view the preview, click on the Generate preview button next to the 'Create release name' field. Note: In some cases a preview may not be available.	-

3. Click **Save** to save your changes.

Release versioning using variables

Release versioning using variables allows you to develop more complex naming schemes, based upon variables set up within Bamboo. You can use global, plan and build variables in your releasing scheme.

Example

You may have a plan variable called '*planvar*' with a value of 'm6'. By including this variable key within the 'Next release name' field, Bamboo will automatically add the variable value to the next release name, and increment it accordingly:

Variable Key	Variable Value	Next Release Name	Next Release	Subsequent Release
planvar	m6	1.0-\${bamboo.planvar}	1.0-m6	1.1-m7

To configure release naming using variables:

1. Click the **Release versioning** button on the 'Deployment project configuration' screen to display the 'Release naming' screen:

Release versioning: Deployment for Project Bonsai
[How release versioning works](#)

Specify what version Bamboo should assign to automatically created releases. You can override this manually whenever you create a new release, but you'll have to come here for any lasting changes to take effect.

Next release

Version*

What version should Bamboo use for the next release? e.g. 1.0-m1 or 1.0-\${bamboo.buildNumber}

[Add variable to version](#)

Automatically increment with each new release

If you can't find variables in the list below that you added to the version, press 'Generate preview'. Bamboo will automatically add a suffix to non-unique versions.

Numbers
☒ Last number in version

Variables
☒ planvar

Only plan variables and global variables are incrementable

Preview

Version of next release 1.0-m6

Version of subsequent release 1.1-m7

Why do some variables still show in the preview?

Bamboo can only substitute variables in the preview if they do not depend on other data. If a variable depends on other workflows (e.g. \${bamboo.buildNumber}), it will be replaced at runtime but not in the preview.

2. Complete the 'Create release name' field using the following data:

Field	Description	Optional?
Version	The identification for the next release name that Bamboo will create. In simple release versioning, you should use something straightforward like 1.0.	

3. Click on the **Add variable to release name** link to display the Variables selection screen:

Variables

Incrementable Variables

Key	Value	Type	Actions
myvar	WHATEVER2	GLOBAL	Add variable
zip	5.0.0	GLOBAL	Add variable
planvar	m6	PLAN	Add variable

Build Variables

The below variables are generated when a plan runs. There may be other variables generated by your plan not listed here.

Key	Value	Actions
planKey	BUILD-SOME	Add variable
planName	Build - Something	Add variable
buildNumber	e.g. 1234	Add variable
shortPlanKey	SOME	Add variable
buildResultKey	e.g. BUILD-SOME-1234	Add variable

Close

- Click **Add variable** to include the variable in your release naming scheme. Click on **Close** to return to the 'Release naming' screen
- Complete the 'Release name' field using the following fields:

Field	Description	Optional?
Automatically increment with each new release	1.Check the 'Numbers' check box to automatically increment the release number according to the 'Next release name' field, as defined above. If you leave this box unchecked, then no release number incrementing will occur. 2. Check the 'Variables' check box to include the associated variable in your automated release number incrementing plan.	
Preview	This field allows you to preview what the next release name will look like. To view the preview, click on the Generate preview button next to the 'Create release name' field. Note: In some cases a preview may not be available.	-

- Click **Save** to save your changes.

Example release versioning schemes

Bamboo also allows you to use combinations of simple and variable release naming. The following table provides examples of combined naming schemes, and demonstrates how careful control of the Numbers and Variables checkboxes can be used to customise your scheme.

Naming scheme	Next release field	Numbers checkbox	Variables checkbox	Variable value	Next release name	Subsequent release name
Static naming	1.0			-	1.0	1.0
Naturally unique variable	1.0- <code>{bamboo.buildNumber}</code>			13	1.0-13	1.0-13
Number incrementing	1.0			-	1.0	1.1
Number incrementing + static variable	1.0- <code>{bamboo.appName}</code>			<code>{bamboo.appName}</code> Plan or Global variable: Awesome	1.0-Awesome	1.1-Awesome

Variable incrementing	1.0-\${bamboo.milestone}			\${bamboo.milestone} Plan or Global variable: m6	1.0-m6	1.0-m7
Number and variable incrementing	1.0-b\${bamboo.appNumber}			\${bamboo.appNumber} User defined variable: 1567	1.0-b1567	1.1-b1568

Creating a deployment environment

Once you have created and configured your new deployment project, you can create environments for it to deploy to. Bamboo allows you to create multiple deployment environments and also allows you to manage:

- Environment details such as name and description
- Tasks
- Triggers
- Permissions
- Agents
- Notifications
- Variables

To create a new deployment environment you will need to:

1. Provide a name that represents your environment e.g. Test or Production
2. Provide a description that describes the function of your environment.

On this page:

- [Creating a new deployment environment](#)
- [Using the deployment environment panel](#)
- [Editing the environment details](#)

Related pages:

- [Tasks for deployment environments](#)
- [Triggers for deployment environments](#)
- [Permissions for deployment environments](#)
- [Agents for deployment environments](#)
- [Notifications for deployment environments](#)
- [Variables for deployment environments](#)

Deployment environments are added from the Deployment project configuration screen:

Deployment projects / Deployment for Project Bonsai

Edit details
 Add environment
 Actions

Configuration: Deployment for Project Bonsai

A test deployment project for Bonsai

What you want to deploy

Source build plan [Project Bonsai](#) > [Atlassian IRKD Continuous Integration](#)

Available artifacts [IRKD microsite WAR](#), [Clover Report \(System\)](#), [clover](#)

Edit build plan
 Release versioning
 Project permissions



Creating a new deployment environment

To create a new deployment environment:

1. Click the blue **Add environment** button from the 'Deployment project configuration' screen. The 'Set up environment' screen will display:

2. Complete the 'Set up deployment project' screen using the following fields:

3. Click on **Create and back** to create the environment and return to the 'Deployment project configuration' screen.
Note: Because no tasks have been defined for this environment, it will appear with a red exclamation mark:

Alternatively, click on **Continue to task setup** to continue to task setup for your new environment:

Set up tasks: Dev Sandbox

How deployment tasks work

What are the tasks that need to happen to make this deployment a success

3 agents and 5 elastic images have the capabilities to run this deployment

Clean working directory task

Artifact download

Download version contents

Final tasks Are always executed even if a previous task fails

Drag tasks here to make them final

Add task

No task selected

Select a task from the list on the left to configure it.

Finish deployment project

See [Tasks for deployment environments](#) for more information on task configuration.

Using the deployment environment panel

All deployment environments are managed from the Deployment project configuration screen. By default, when the screen loads, each environment panel is displayed in its collapsed state. Click **Edit** to expand the deployment environment panel:

Environment: Dev Sandbox

Sandboxed development environment

Deploy
 Edit...

When expanded, the environment panel shows three separate sub-panels:

Environment: Dev Sandbox

Sandboxed development environment

Deploy
 Actions
 Minimise

How you want to deploy

Tasks define the steps involved in deploying a version to the Environment.

Edit tasks

Other environment settings

These settings are not strictly necessary for your deployment to run, but they can be very helpful and allow you to make Bamboo Deployments go just right.

Triggers

Environment permissions

Agents

Notifications

Variables

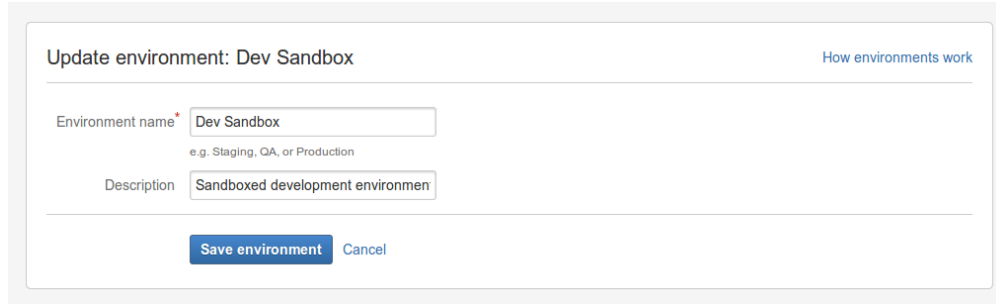
The three sub-panels provide the following functionality:

Sub-panel	Functionality	Description
Environment	- Deploy - Actions - Minimise	- Manually deploys to the environment - Allows the user to View, Delete or Move down the environment - Minimise the environment panel back to its collapsed state
How you want to deploy	Edit tasks	Allows the user to edit the tasks associated with the environment
Other settings	- Triggers - Environment permissions - Agents - Notifications - Variables	A set of optional settings that make your Bamboo deployments run more smoothly

Editing the environment details

Bamboo allows you to change both the environment name and the description. To edit these details:

1. Expand the environment panel and click the pencil icon next to the environment name. The 'Update environment' screen will display:



2. Complete the 'Update deployment project' screen using the following fields:

Field	Description	Optional?
Environment name	The name of the environment	
Description	A brief description of your deployment project	

3. Click **Save environment** to save your changes.

Tasks for deployment environments

Once you have created and configured your new deployment project and deployment environments, you can set up associated tasks for the deployment process. Bamboo allows you to execute a range of different tasks upon deployment including:

- Bash and other shell commands
- Bespoke written scripts
- SCP, SSH and Artifact handler tasks
- Ant executables
- Maven 1.x, 2.x & 3.x executables
- Tomcat executables
- Heroku deployments

On this page:

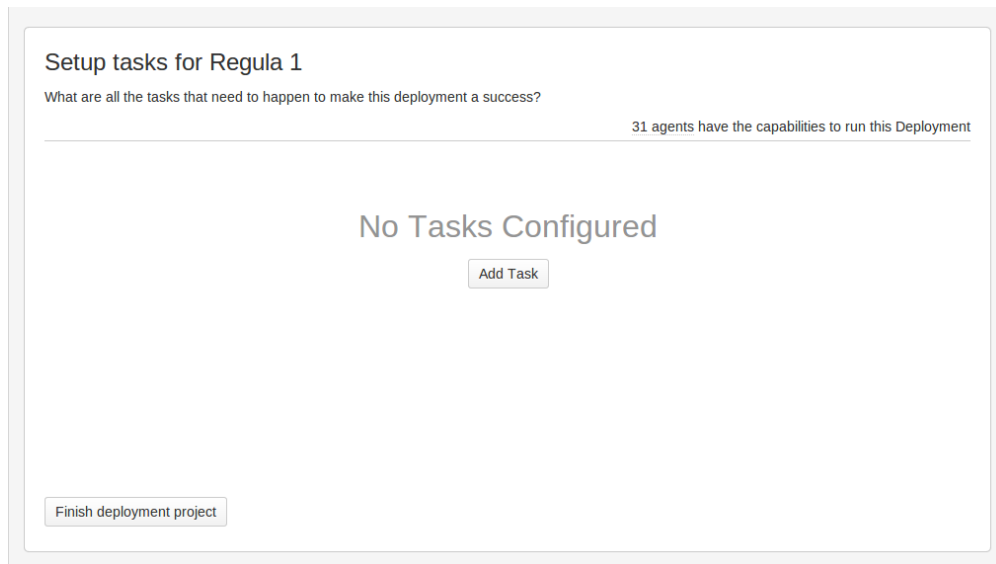
- [Adding an environment task](#)
- [Some useful deployment tasks](#)
 - [Deploying with Tomcat](#)
 - [Copying and moving files with SCP](#)
 - [Integrating with Heroku](#)
 - [Deploying ASP.NET applications with MSDeploy](#)
- [Assigning a final task](#)
- [Editing an environment task](#)

Adding an environment task

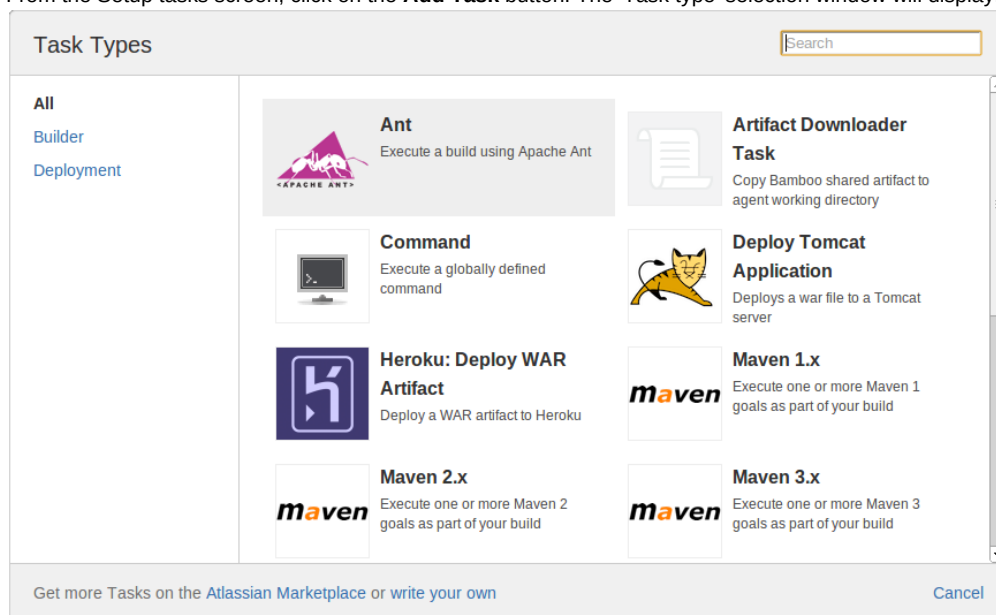
Tasks may be added to a deployment environment during or after the environment creation process; tasks may also be edited and changed afterwards as well.

To add a task to a deployment environment:

1. Open your deployment project and expand the relevant environment panel. In the Other settings section, click on the **Tasks** button. The 'Set up tasks' screen will display:



- From the Setup tasks screen, click on the **Add Task** button. The 'Task type' selection window will display:

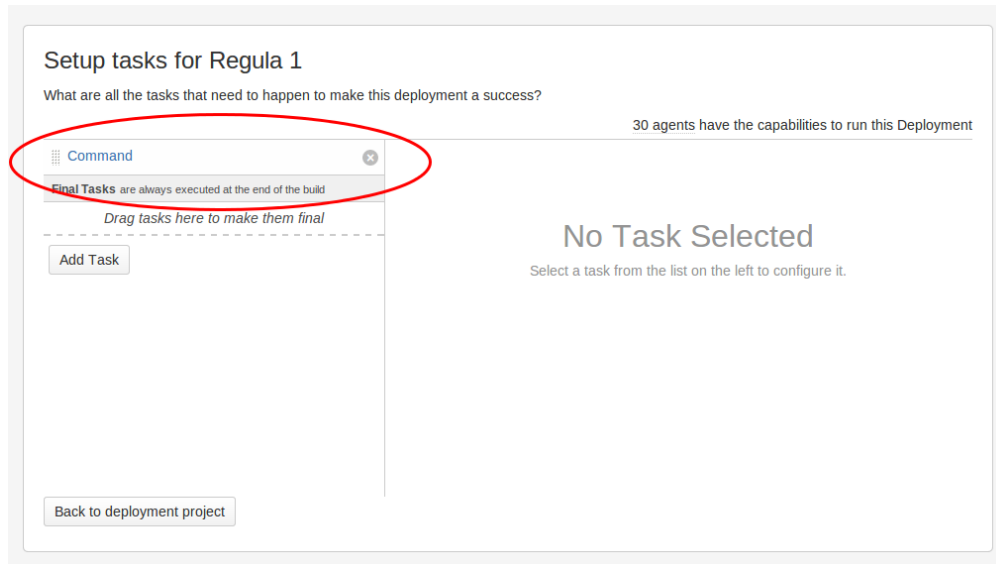


- Click on the task type that you want to use. Only tasks applicable to the deployment environment will be available for selection. The task configuration window will open, and you can configure the task according to the needs of your deployment project. Different tasks will have different requirements



Remember that capability and requirement matching is still in effect for deployment environments. If your task does not have the right capabilities it will not be executed, even if the relationship has been defined.

- Click on the **Save** button to save your task configuration. The Setup tasks screen will display again. Your newly configured task will display in the left side of the pane:



5. Click on the **Add Task** button to add additional tasks to your environment and complete the above process again
6. When all of the tasks have been set up, click on **Back to deployment project** to return to the deployment project page.

Some useful deployment tasks

Deploying with Tomcat

You can use Bamboo to deploy and manage your Java web application with Tomcat 6 or 7, without having to directly interact with Maven, Ant or write special scripts.

See [Using Tomcat with Bamboo for continuous deployment](#).

Copying and moving files with SCP

You can use the Bamboo SCP task to upload files from Bamboo directly to a remote server as part of a Bamboo job. The SCP task is able to copy multiple files and preserves the directory structure for the copied files.

See [Using the SCP task in Bamboo](#).

Integrating with Heroku

You can use Bamboo to deploy your Java web application to the [Heroku](#) cloud platform.

See [Using the Heroku task in Bamboo](#).

Deploying ASP.NET applications with MSDeploy

You can use Bamboo to deploy your ASP.NET web application by using a [Script](#) task to run `msdeploy.exe`.

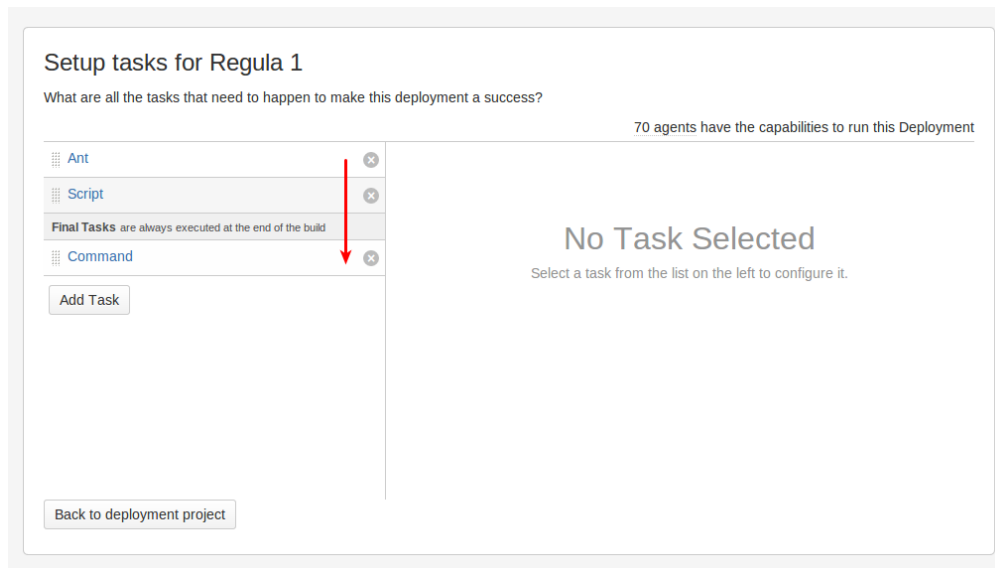
The MSDeploy command-line syntax is available at: [http://technet.microsoft.com/en-us/library/dd569106\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/dd569106(v=ws.10).aspx)

Assigning a final task

Once all of your tasks have been configured, you may assign some or all of them to be Final Tasks. Final Tasks are always executed at the end of the build.

To assign a final task:

1. Open your deployment project and expand the relevant environment panel. In the Other settings section, click on the **Tasks** button to display the 'Set up tasks' screen.
2. To make a task a Final Task, simply drag the task beneath the Final Task bar in the Setup tasks window:

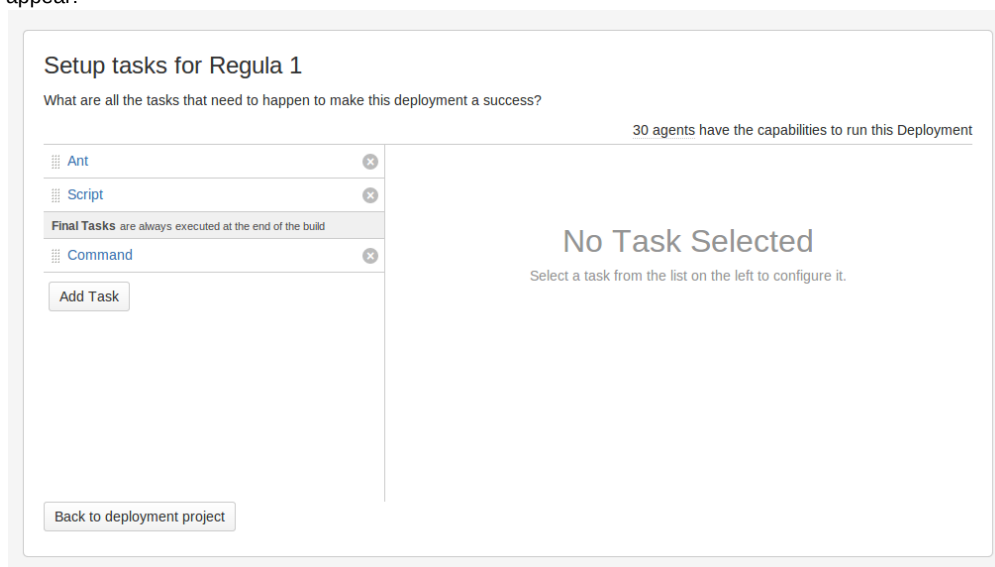


3. Click on **Back to deployment project** to return to the deployment project page.

Editing an environment task

If you have already added some tasks to an environment, then they can be easily changed at a later stage. To edit environment tasks:

1. Open your deployment project and expand the relevant environment panel. Click on **Edit tasks**. The 'Setup tasks' screen will appear:



2. Existing tasks can be edited by clicking on the task in the left hand pane of the setup window
3. New tasks can be added by clicking on the **Add Task** button and following the process described in the above section
4. Unwanted tasks can be deleted from the environment by clicking on the cross icon belonging to the task.

Using the Heroku task in Bamboo

You can use Bamboo to deploy your Java web application to the [Heroku](#) cloud platform.

The plan that does this:

- Should have access to the previously created WAR artifact. For example, in Maven-based projects, this would typically be created using the package goal. See [Configuring artifact sharing between jobs](#).
- Should use the **Heroku: Deploy WAR Artifact** task. To configure this task, specify the **API Key**, **App Name**, and **WAR File** to use. Your API key can be found on the [Heroku account page](#). If the app name specified does not exist, it will be created for you. The WAR file value should be the relative path to the WAR artifact.

To find out more, go to the Heroku documentation:

- [Getting started with the Heroku plugin for Atlassian Bamboo](#)
- [Getting started with Java on Heroku](#)



Note that the Heroku task is bundled with Bamboo 4.3, but is also available from the [Atlassian Marketplace](#).

Triggers for deployment environments

Bamboo offers a range of optional settings to make your deployment project function more smoothly. Deployment triggers allow you to configure how and when Bamboo will trigger a deployment project. You can set up triggers to start your deployment automatically. Triggering strategies include:

- Cron based scheduling
- Timed daily build
- Successful plan conclusion

On this page:

- [Configuring a deployment trigger](#)
- [Editing a deployment trigger](#)

Configuring a deployment trigger

Deployment triggers are configured as part of the Other settings section of the environment panel. To configure a trigger:

1. Open your deployment project and expand the relevant environment panel. In the Other settings section, click on the **Triggers** button. The Triggers screen will display:

A screenshot of the 'Edit triggers: Dev Sandbox' screen in Bamboo. The screen has a header 'Edit triggers: Dev Sandbox' and a link 'How deployment triggers work'. Below the header, there is a paragraph: 'Set triggers to specify how and when the deployment will be triggered automatically. When a deployment is automatically triggered, a new release is created from the latest successful build result of the linked plan. If you want to deploy to this environment manually, use the "Deploy" action at any time.' The main content area displays 'No automatic triggers defined' with an 'Add trigger' button below it. At the bottom, there is a 'Back to deployment project' button.

2. Click on the **Add Trigger** button to display the Trigger configuration screen:

Edit triggers: Dev Sandbox
[How deployment triggers work](#)

Set triggers to specify how and when the deployment will be triggered automatically. When a deployment is automatically triggered, a new release is created from the latest successful build result of the linked plan.

If you want to deploy to this environment manually, use the "Deploy" action at any time.

New Trigger
Add trigger

Trigger configuration

Trigger description

Trigger type*

Scheduled

How should Bamboo trigger this Deployment?

Schedule*
Daily at 12:00 am

Save trigger
Cancel

Back to deployment project

3. Complete the 'Trigger configuration' screen using the following fields:

Field	Description	Optional?
Trigger description	A brief description of your trigger	
Trigger type	The type of trigger you wish to employ. Available triggers are: - Cron based scheduling - Timed daily build - Successful plan conclusion	

Each trigger type will require individual configuration as described:

Trigger type	Description	Data entry	Entry tool
Cron based scheduling	Schedule entry	Click on the edit pencil next to the time entry and enter via the Schedule editor	Schedule* Daily at 12:00 am
Timed daily build	Nominated build time	Enter a time in the Build Time field E.g. 17:30	Build Time At what time should Bamboo build? Use hh:
Successful plan conclusion	Valid plan name	Enter a valid plan name using the drop-down field	Run this Deployment after success Z Personal Branches - PW test

Only triggers applicable to the deployment environment will be available for selection

4. Click on the **Save Trigger** button to save your configuration.

Editing a deployment trigger

Existing triggers can be edited at any point by clicking on the **Scheduled builds** link in the left hand pane of the Triggers window:

Triggers

If you want Bamboo to start this deployment automatically, you will need to set Triggers to specify how and when the build will be triggered. If you want to start it manually at any time, use the "Run" menu or [trigger a release from JIRA](#).

[Scheduled builds](#)

Add Trigger

No Trigger selected

Select a trigger from the list on the left to configure it.

Back to deployment project

Permissions for deployment environments

Bamboo offers a range of optional settings to make your deployment project function more smoothly. Deployment environment permissions allow you to configure which groups or individuals can view, edit or deploy a project.

i Note that the global Bamboo permissions still take precedence. Where a user has environment permissions enabled but project permissions disabled, they will still be unable to access a deployment environment. Please see [Bamboo permissions](#) and [Creating and configuring a deployment project](#) for more information on managing deployment project permissions.

On this page:

- [Configuring deployment environment permissions](#)

Configuring deployment environment permissions

Deployment environment permissions are configured as part of the Other settings section of the environment panel.

To configure your permission strategy:

1. Open your deployment project and expand the relevant environment panel. In the Other settings section, click on the **Permissions** button. The 'Edit permissions' screen will display:

Edit environment permission: Dev Sandbox

[How environment permissions work](#)

Define what users are allowed to edit and view in the environment.

Add user
Add group

Users	View	Edit	Deploy
Admin User	☒	☒	☒
Other	View	Edit	Deploy
Logged in users	☒	☐	☐
Anonymous users	☒		

Permission types
View - Controls who can see this environment and its deployment results. Users must have view permission on the deployment project.
Edit - Controls who can edit this environment
Deploy - Defines who can releases for this project and who can deploy them to this environment.

[Edit project permissions](#)

Save Cancel

2. Click the **Add User** or **Add Group** button to search for and add, users or groups
3. Check the relevant 'View' and 'Edit' permission boxes to assign your desired permission scheme
4. Click **Save** to save your permission scheme.

Agents for deployment environments

Bamboo offers a range of optional settings to make your deployment project function more smoothly. Bamboo allows you to assign specific agents, elastic agents or image configurations to execute the deployment for the environment.

Note: When an agent is assigned, no other builds or deployments can run on it unless they are also explicitly assigned to use that agent or image configuration.

On this page:

- [Configuring deployment agents](#)

Configuring deployment agents

Deployment environment agents are configured as part of the Other settings section of the environment panel.

To configure your deployment agent:

1. Open your deployment project and expand the relevant environment panel. In the Other settings section, click on the **Agents** button. The 'Assigned agents' screen will display:

Edit agents: Dev Sandbox [How agent assignment works](#)

Dedicate specific agents or image configurations to execute all deployments for this environment. If you do not assign an agent to this deployment, one will be chosen at run time according to [standard requirement/capability mappings](#).

NOTE: By adding an agent to this list, it will no longer execute any build jobs. It will only execute deployments for this environment, and other environments that are explicitly configured to use this agent. One agent can be assigned to several deployment environments.

Agent / Image

Save [Cancel](#)

2. Enter an agent name, or use the drop down menu to select an appropriate agent:

Edit agents: Dev Sandbox [How agent assignment works](#)

Dedicate specific agents or image configurations to execute all deployments for this environment. If you do not assign an agent to this deployment, one will be chosen at run time according to [standard requirement/capability mappings](#).

NOTE: By adding an agent to this list, it will no longer execute any build jobs. It will only execute deployments for this environment, and other environments that are explicitly configured to use this agent. One agent can be assigned to several deployment environments.

Agent / Image

Name	Type	
Local Agent 1	Local Agent	✕
Default Image EBS i386 (linux)	Elastic Image	✕
Default Image EBS x86_64 (windows)	Elastic Image	✕

Save [Cancel](#)

Only agents applicable to the deployment environment will be available for selection.



Remember that capability and requirement matching still applies for deployment environments. If your agent does not have the right capabilities it will not be assigned at runtime even if the relationship has been defined.

3. Click on **Save** to save your agent scheme
4. You can remove an unwanted agent by clicking the associated cross on the right hand side of the screen.

Notifications for deployment environments

Bamboo offers a range of optional settings to make your deployment project function more smoothly. Notifications allow you to assign a specific notification scheme to events triggered by the deployment environment. Notification events include start and finish of a deployment, and may be delivered by any of:

- User or group notification
- Email
- Hipchat
- Instant Messaging

To set up a notification you will need to:

1. Select a triggering event
2. Configure a mechanism for delivering notifications

On this page:

- [Configuring deployment notifications](#)

Configuring deployment notifications

Deployment environment notifications are configured as part of the Other settings section of the environment panel.

To configure your deployment notifications:

1. Open your deployment project and expand the relevant environment panel. In the Other settings section, click on the **Notifications** button. The 'Edit notifications' screen will display:

Edit environment notifications: Dev Sandbox [How environment notifications work](#)

Define who receives notifications about deployment events for this environment.

ⓘ There is currently no instant messaging server or email server configured. Bamboo will not send out any notifications.
[Add an email server](#) or [add an instant messaging server](#) now.

Current notifications
There are currently no notifications set up for this plan.

Add notification

Event: **Deployment Started and Finished**
Notifications sent when a deployment is started or finished.

Recipient type: **User**

User: **Start typing to select**
User to receive notification

Add **Cancel**

[Back to deployment project](#)

2. Complete the 'Edit notifications' screen by selecting the event to trigger the notification:

Notification	Description
Deployment started and finished	Notification is issued when a deployment is started and finished
Deployment finished	Notification is issued only when deployment is finished

And configuring the notification delivery system:

Recipient Type	Data requirements
User	Username of the user
Hipchat	Hipchat API token, Hipchat room name, Room participants notification
Group	Groupname
Email Address	Email address
IM Address	Instant messaging address

Note: If you have not done so, you may need to set up an IM server for IM notifications to work correctly.

3. Click the **Add** button to add your notification:

Edit environment notifications: Dev Sandbox
[How environment notifications work](#)

Define who receives notifications about deployment events for this environment.

There is currently no instant messaging server or email server configured. Bamboo will not send out any notifications. [Add an email server](#) or [add an instant messaging server now](#).

Current notifications

Event	Notification recipient	Actions
Deployment Finished	Brenda@work.com (email address)	✎ ✕
	Colin@work.com (email address)	✎ ✕
Deployment Started and Finished	Bill@work.com (email address)	✎ ✕
	Jane@chat.com (im address)	✎ ✕

Add notification

Event

Deployment Finished

Notification sent when a deployment is finished.

Recipient type

Email Address

Email address

Email address (eg. bob@work.com)

Add

Cancel

Back to deployment project

- You can edit or remove notifications by clicking on the associated **Edit** or **Remove** link on the right hand side of the screen.

Variables for deployment environments

Bamboo provides a number of deployment variables for use when deploying releases to environments. These are listed in the tables on this page. For Bamboo variables to do with build plans, see [Bamboo variables](#).

Configuring variables for deployment environments

Deployment environment variables are configured as part of the Other settings section of the environment panel.

On this page:

- [Configuring variables for deployment environments](#)

To configure an environment variable:

- Open your deployment project and expand the relevant environment panel. In the Other settings section, click **Variables**:

Edit variables: Dev Sandbox
[How deployment variables work](#)

Define variables to be used by your deployment tasks. You can use any of the keys below in your task configuration or scripts and it will be automatically substituted with the value. You can override global variables by using the same key.

Key	Value	
		Add

*If the key contains the phrase 'password', the value will be masked with '*****' in the build logs. e.g. if the key is 'password', 'repopassword' or 'userpassword' the value will be masked.*

Back to deployment project

- Enter a valid key and value into the relevant fields in the Variables screen.
- Click on Add to add the variable scheme:

Edit variables: Dev Sandbox
[How deployment variables work](#)

Define variables to be used by your deployment tasks. You can use any of the keys below in your task configuration or scripts and it will be automatically substituted with the value. You can override global variables by using the same key.

Key	Value
planvar	m6

💡 If the key contains the phrase 'password', the value will be masked with "*****" in the build logs. e.g. if the key is 'password', 'repopassword' or 'userpassword' the value will be masked.

[Back to deployment project](#)

- You can remove unwanted variables by clicking the relevant dustbin icon on the right of the Variables screen.
- Click **Back to deployment project** to return.

Managing deployment projects

Bamboo makes it easy to manage and monitor your deployment projects.

A single dashboard screen allows you to monitor deployment environments, deployment status, releases and time/date stamps. It also allows you to edit and deploy your projects.

On this page:

- Managing deployment projects
- Viewing individual deployment projects
- Project summary
 - Releases

Managing deployment projects

Deployment projects are viewed and managed from the 'All deployment projects' screen. Think of this as a dashboard view of all of your deployment projects.

To manage your existing deployment projects

- Click **Deploy > All deployment projects** in the drop down menu from the header bar. The 'All deployment projects' screen will appear:

All Deployment Projects				
Atlassian IRKD Microsite				
Environment	Status	Version	Completed	Actions
Dev Sandbox	DEPLOYED	1.3_RC4	05 June 2013 10:06 PM	
Production	DEPLOYED	1.2_RC3 APPROVED	03 June 2013 10:56 PM	
QA Env.	DEPLOYED	1.3_RC2 BROKEN	04 June 2013 11:11 PM	
Deployment for Project Bonsai				
A test deployment project for Bonsai				
Environment	Status	Version	Completed	Actions
Dev Sandbox	Never deployed			
Staging	Never deployed			

It provides an overview of all your development projects, including the following useful information:

Item	Description
Name	The name of the deployment project

Environment	The environment the release was deployed into
Status	The current status of the deployment. Values are <i>Never deployed</i> , <i>Deployed</i> , <i>Deploying</i> and <i>Failed</i>
Release	The release that was deployed or attempted deployment to that environment
Completed	The time and date stamp for the deployment, or the time spent deploying so far
Actions	Actions that can be performed. These are <i>Deploy</i> and <i>Edit</i>

Viewing individual deployment projects

Individual deployment projects can be viewed in greater detail. Available details include:

- Associated environments
- Release history
- Project item details (artifacts)

To view individual deployment projects

1. From the 'All deployment projects' screen, simply click on the deployment project name:
A detailed view of the deployment project will display. From this tabbed display, you can view details of your project's summary and releases.

Deployment projects
Atlassian IRKD Microsite
+ Create release
Configure project

Project summary
Releases

Deployment project summary
Source build plan Project Bonsai > Atlassian IRKD Continuous Integration
Available artifacts IRKD microsite WAR, Clover Report (System), clover

Environment	Status	Release	Completed	Actions
Dev Sandbox	DEPLOYED	1.3_RC4	05 June 2013 10:06 PM	
Production	DEPLOYED	1.2_RC3 APPROVED	03 June 2013 10:56 PM	
QA Env.	DEPLOYED	1.3_RC2 BROKEN	04 June 2013 11:11 PM	

Project summary

The Associated environments tab provides details of the environments that are associated with the deployment project:

Project summary
Releases

Deployment project summary
Source build plan Project Bonsai > Atlassian IRKD Continuous Integration
Available artifacts IRKD microsite WAR, Clover Report (System), clover

Environment	Status	Release	Completed	Actions
Dev Sandbox	DEPLOYED	1.3_RC4	05 June 2013 10:06 PM	
Production	DEPLOYED	1.2_RC3 APPROVED	03 June 2013 10:56 PM	
QA Env.	DEPLOYED	1.3_RC2 BROKEN	04 June 2013 11:11 PM	

Details available on this screen include:

Item	Description
Environment	The environment the release was deployed into
Status	The current status of the deployment. Values are <i>Never deployed</i> , <i>Deployed</i> , <i>Deploying</i> and <i>Failed</i>
Release	The release that was deployed or attempted deployment to that environment

Completed	The time and date stamp for the deployment, or the time spent deploying so far
Actions	Actions that can be performed. These are <i>Deploy</i> and <i>Edit</i>

Releases

The Releases tab provides details of the releases associated with the deployment project:

Project summary Releases			
Currently deployed			
Version	Flagged	Created	Deployed on
1.3_RC4		05 June 2013 10:05 PM	Dev Sandbox
1.3_RC2	BROKEN	04 June 2013 11:09 PM	QA Env.
1.2_RC3	APPROVED	03 June 2013 10:39 PM	Production
History			
Version	Flagged	Created	Was deployed on
1.3_RC4		05 June 2013 10:05 PM	Dev Sandbox
1.3_RC3		05 June 2013 09:45 PM	Dev Sandbox
1.3_RC2	BROKEN	04 June 2013 11:09 PM	QA Env.
1.4		04 June 2013 09:59 PM	
1.3_RC1	BROKEN	03 June 2013 11:10 PM	QA Env.

This view provides information on the currently deployed release, and also on the history of previous releases. Release details include:

Item	Description
Name	The name of the release
Flagged	Any flags that have been applied to the release. Values are <i>Broken</i> and <i>Approved</i> . Neutral flags remain blank
Created	The time and date stamp for the creation of the release
Deployed on	The environment the release was deployed into

Manually starting a deployment

Bamboo can start deployments either by configuring triggers, or by starting the process manually.

Manually executing the deployment gives you the ability to start the process at your convenience, without having to wait for a scheduled event or trigger to take place.

On this page:

- [Manually starting a deployment](#)

Manually starting a deployment

Deployment projects are viewed and managed from the 'All deployment projects' screen. Think of this as a dashboard view of all of your deployment projects. You can also start deployments from this screen.

To manually start a deployment

1. From the 'All deployment projects' screen, click on the associated **Deploy** icon:



The 'Execute deployment' screen will display:

Execute deployment: Dev Sandbox

Select a release to deploy

Release to deploy ☐ Latest version available: 1.0-m6
☒ Other:
 Want to [create a new release](#) instead?

Release details

Version 1.0-m6

Reviewed not reviewed

Difference to deployed version no issues or commits

Deployment history Never been deployed

Current release on Dev Sandbox

never been deployed to

Execution

Tasks to run

1. Clean working directory task
2. Artifact download - Download release contents
3. Script - echo beach

Estimated time unknown

Eligible agents 3 agents and 5 elastic images

[Start deployment](#) [Cancel](#)

2. Using the radio buttons, decide if you wish to use the latest release available, a different release, or create a new release
3. If using the latest available release:
 - a. Check the **Latest version available** radio button
 - b. The Release details segment of the screen will update the release version to be used
 - c. Click on **Start deployment**
4. If using a different release:
 - a. Check the **Other** radio button
 - b. Use the drop down menu to search for a version
 - c. The Release details segment of the screen will update the release version to be used
 - d. Click on **Start deployment**
5. If creating a new release:
 - a. Click **create a new release** beneath the release version text field. The Create new release screen will display:

Create new release: Deployment for Project Bonsai [How deployment releases work](#)

Define release version

Last version 1.0-m6
Release created by Admin User 21 hours ago

This version
A unique, human readable label for your release e.g. 3.0-beta1

Subsequent version 1.2-\${bamboo.planvar}
Want a different one? [Change release versioning system](#)

Choose build result

Source build plan [Project Bonsai > Atlassian IRKD Continuous Integration](#)

Choose build result ☒ #131 Manual run by Paige N. Spector [QA] 05 Jun 2013
The above are the only successful build results available to create a new release.

Changes since last release created

The build result you have selected was used in the last release created. As such you will see no issues or commits for the newly created release.

[Create release](#) [Cancel](#)

- b. Enter a release version number in the 'This version' field, or alternatively use the default setting provided
- c. Select a source build plan from the list provided under 'Choose build result'
- d. Click on **Create release**

Configuring plans

A *plan* defines everything about your continuous integration build process in Bamboo.

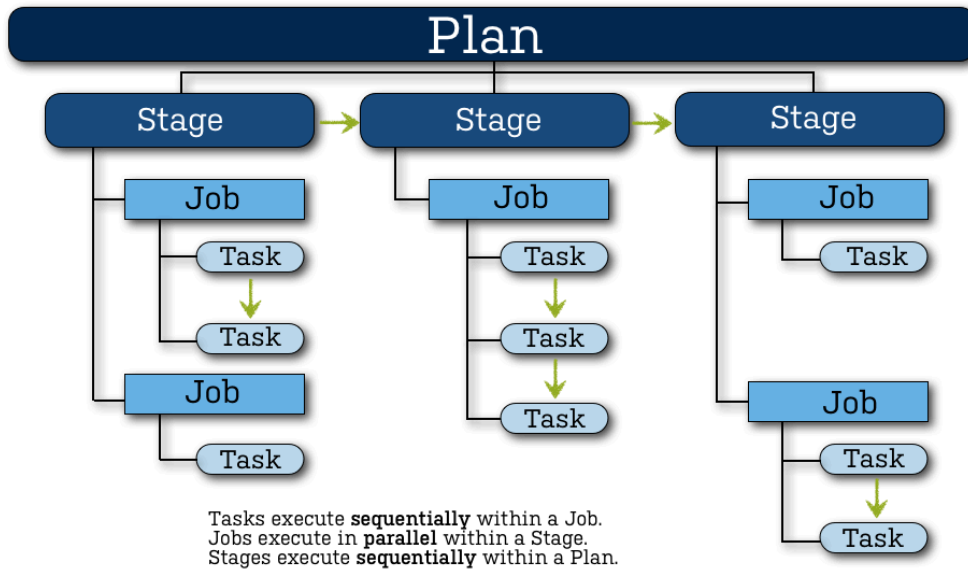
A plan:

- Has a single [stage](#), by default, but can be used to group [jobs](#) into multiple stages.
- Processes a series of one or more stages that are run **sequentially** using the **same** repository.
- Specifies the default repository.
- Specifies how the build is [triggered](#), and the triggering dependencies between the plan and other plans in the project.
- Specifies [notifications of build results](#).
- Specifies who has [permission](#) to view and configure the plan and its jobs.
- Provides for the definition of [plan variables](#).

Every plan belongs to a [project](#).

Projects and plans can only be configured by Bamboo administrators (see [Creating a plan](#)).

Diagram showing the relationship between plans, stages, jobs and tasks:



On this page:

- [Navigating to a plan](#)

Related pages:

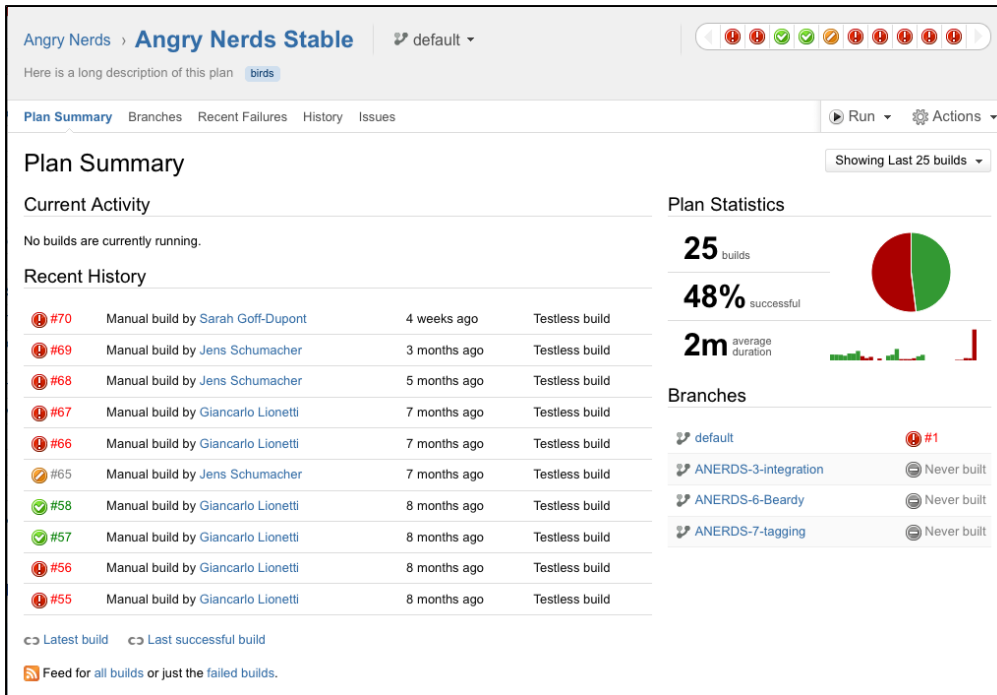
- [Viewing a plan's build information](#)
- [Creating a plan](#)
- [Using plan branches](#)
- [Managing plans](#)

Navigating to a plan

To navigate to a plan:

1. Click **Dashboard** and then the **All Plans** tab.
2. In the list of plans, click the name of the desired plan. The plan's 'Plan Summary' page will be displayed.
3. Choose **Actions** > **Configure** Plan to see the configuration pages for the plan.

Screenshot: *The Plan Summary page*



Viewing a plan's build information

A [plan](#) defines everything about your continuous integration build process in Bamboo.

To view information about a plan:

- Navigate to the desired plan, as follows:
 - If you are viewing the [Dashboard](#), locate and click the plan's name in the list, or
 - If you are viewing a job or [build result](#), click the plan name in the breadcrumb links at the top of the screen.
- Click a tab to view information about the plan:

Related pages:

- Using the Bamboo dashboard
- Viewing a build result
- Editing a plan's configuration
- Configuring a plan's permissions
- Integrating Bamboo with JIRA

Tab	Notes
Plan Summary	Information about the plan, as shown in the diagram below.
Branches	The branch plans for this plan.
Recent Failures	Information about recent failures of the plan, including the builds that failed, links to the build results, time taken to fix, etc.
History	The full history of builds of the plan.
Tests	A summary of the 10 most frequently broken tests.
Quarantined Tests	Failing test's results that have been disconnected from the build results.
Issues	View the JIRA issues linked to builds of your plan. (Only displayed if your administrator has integrated Bamboo with JIRA .)

Use the **Actions** menu to access functions for the plan, such as **Disable Plan** and **Configure Plan**. (This menu is only displayed if you are an [administrator](#) for the plan.)

Diagram: Plan Summary (annotated)

Navigation
Click the project to view a summary of its plans.

Tabs
View further details for the plan.

Current activity and recent history
See whether the plan is building.
Shows results for the ten most recent plan builds. Click a build number to view that build result.

Plan status
Icons show if the plan is building, and the status of recent builds.
Click an icon to see the [build result](#).

Actions menu
e.g. Configure Plan
Disable Plan

Run menu
Choose options for running the plan
e.g. run a parameterised build.

Statistics and charts
For the build history of the plan.

Plan branches
A summary of all the branches for the plan.

Creating a plan

Before you begin:

- You require the 'Create Plan' or 'Admin' [global permission](#) to create new plans.

To create a Plan in Bamboo:

- Click **Create Plan** in the top navigation bar.
- Choose and complete one of the following options:
 - Create a New Plan**
 - Clone an Existing Plan** — This option only appears if there is at least one existing plan, and you have the 'Clone' and/or 'Admin' plan permission for at least one plan, on the Bamboo server.
 - Import a Maven 2 Project** — This option only appears if Maven 2 has been installed.

Screenshot: Create Plan options

Create Plan

A Plan defines everything about your build process, including what gets built, how the Build is triggered and what Jobs are executed.

Create a New Plan

Create a completely new Plan, specify its default repository and configure the Executable for this Plan's Default Job.

Clone an Existing Plan

Make a copy of a Plan and its entire configuration.

Import a Maven 2 Project

You can import a Plan into Bamboo from a Maven 2 project by getting Bamboo to parse the Plan information from this project's pom.xml file.

Creating a new plan

This page describes how to create a completely new [plan](#).

For other ways to [create a plan](#) see:

- [Cloning an existing plan](#)
- [Importing a Maven 2 project](#)

On this page:

- [Step 1. Create the plan](#)
- [Step 2. Configure tasks for the plan](#)
- [Notes](#)

Step 1. Create the plan

When you create a new plan, you can define everything about your build process, including what gets built, how the plan's build is triggered and what jobs are executed.

To create a new plan, complete the following sections:

- Click **Create Plan** in the top menu bar, and then click **Create a New Plan**.

2. Complete the following sections:

Plan Details	Select either an existing project to which this new plan will belong, or New Project .
Source Repositories	See the following for details: • Bitbucket - Mercurial • CVS • Git • GitHub • Mercurial • Perforce • Subversion
Trigger	Choose how Bamboo should trigger builds. See Triggering builds .

Step 2. Configure tasks for the plan

When a new plan is created, a default job is also created as part of the plan. You need to configure one or more tasks for the default job. Tasks are the 'units of work' for a plan. You can always add more tasks to the default job after the plan is created, as well as create new jobs.

1. On the 'Configure Tasks' screen, click **Add Task** to add a new task to the new plan.
2. Click the desired task type in the 'Task Types' dialog.
3. Fill out the details for the task. The fields and options will be different depending on the executable that you chose. See the following documentation for specific instructions on each executable:
 - [Checking out code](#)
 - [Configuring a builder task](#)
 - [Configuring a test task](#)
 - [Configuring a deployment task](#)
 - [Pattern matching reference](#)
4. In the 'Enable this plan' section, choose whether or not to enable this plan. Enabling the plan instructs Bamboo to commence executing builds of the plan based on the plan's trigger configuration (defined above). To enable this plan, select the **Yes please!** checkbox.
5. Click **Create**. Bamboo will automatically run an initial build for your new plan. When you next return to the [Dashboard](#), your new plan (and new project, if applicable) will be displayed on the **All Plans** tab.

If you wish to configure more plan options, please refer to [Editing a plan's configuration](#).

Screenshots: Creating a new plan - plan configuration and job configuration (click to view gallery)

The left screenshot shows the 'Create a New Plan' form. It includes fields for Plan Name, Plan ID, Plan Key, Plan Description, Source Repositories, Build Strategy, and Build Frequency. The right screenshot shows the 'Configure Tasks' form, which includes a list of tasks (Source Code Checkout, Checkout Default Repository) and a 'Source Code Checkout Configuration' panel on the right.

Notes

- You can configure Bamboo to automatically start an initial build for a new plan, irrespective of the trigger configuration. To do this, add the `fire.initial.build.for.manual.strategy` to your `bamboo.cfg.xml` file as described in [Configuring system properties](#).

Cloning an existing plan

When you clone an existing plan, you make a copy of that plan and its entire configuration.

To clone an existing plan:

1. Click **Create Plan** in the top menu bar, and then click **Clone an Existing Plan**.
2. Use **Plan to clone** to select a plan. Only plans for which you have the 'Clone' and/or 'Admin' [plan permission](#) are shown.
3. Enter details for the new plan. You can add the new plan to an existing or new project.
4. Choose whether to enable this plan. Enabling the plan instructs Bamboo to start running builds of the plan, based on the plan's trigger configuration.
5. When you click **Create**, the 'Plan Summary' page for the new plan will be displayed. Bamboo will automatically run an initial build for your new plan.

Related pages:

- [Creating a plan](#)
- [Creating a new plan](#)
- [Importing a Maven 2 project](#)

If you wish to configure more plan options, please refer to [Editing a plan's configuration](#). You can configure Bamboo to automatically start an initial build for every new plan. To do this, add the `fire.initial.build.for.manual.strategy` to your `bamboo.cfg.xml` file as described in [Configuring system properties](#).

Screenshot: Cloning an existing plan

Clone an Existing Plan

On this page, you can make a copy of a Plan and its entire configuration.

Plan to clone

Plan to clone Bamboo Main - CI Tests
This will be used as the template for creating your new plan.

Plan Details

Project Bamboo Main
Select or add a Project that the new Plan will be created in.

Plan Name*
How do you want to identify the new Plan?

Plan Key*
This is the key for the plan which must be unique within a project. In conjunction with the project key, it is used to identify a build in URLs, trigger scripts and API calls. The key must contain only uppercase alphanumeric characters. e.g. "CORE"

Plan Description
Choose a meaningful description for the new Plan. For example, "JIRA Release Plan".

Enable this Plan?

- ☒ Yes please!
By selecting this option your Plan will be available for building and change detection straight away.
Do not select this option if you have advanced configuration changes to make after creation.

Importing a Maven 2 project

This page describes how to import a [plan](#) from a Maven 2 project.

Bamboo can check out and parse the `pom.xml` from a Maven 2 project, and create a new plan using the details from it. This feature is not available for Maven 1 projects.

Related pages:

- [Creating a plan](#)
- [Creating a new plan](#)
- [Cloning an existing plan](#)

To create a new plan from a Maven 2 pom.xml file:

1. Click **Create Plan** in the top menu bar, and then click **Import a Maven 2 Project**. Maven 2 needs to be installed on the Bamboo server machine. See [Defining a new executable capability](#).
2. Specify the details required for Bamboo to locate your Maven 2 project's `pom.xml` file.
3. Click **Import**.
4. Confirm the details from your `pom.xml` file that are shown on the 'Confirm Plan Details' page.
5. Choose whether to enable this plan. Enabling the plan instructs Bamboo to start running builds of the plan, based on the plan's trigger configuration.
6. When you click **Create**, the new plan's 'Plan Summary' page will be displayed. Bamboo will automatically run an initial build for your new plan.
7. Click **Confirm** to save your plan configuration so far. The **Tasks** tab of the plan's [default job](#) is displayed.
8. Configure the Maven 2 task options, as described in [Maven](#). You can select a different builder task but this would only be useful if your Maven 2 project's `pom.xml` depended on a non-Maven 2 builder. If you do need to specify a non-Maven 2 builder, refer to the '2. Configure tasks for the plan' section on [Create a new plan](#).

If you wish to configure more plan options, please refer to [Editing a plan's configuration](#).

Screenshot: Importing a plan from Maven 2

Import a Maven 2 Project

On this page, you can import a Plan into Bamboo from a Maven 2 project by getting Bamboo to parse the Plan information from this project's pom.xml file. A Plan with a single Default Job will be created.

Enter pom.xml Details

Source Repository

Repository URL
The location of subversion repository (e.g. http://svn.collab.net/repos/svn/trunk)

Username
(Optional) The subversion username (if any) required to access the Repository

Authentication Type

Password
(Optional) The password required by the subversion username

[Cancel](#)

Notes

- Security Manager settings too strict — Please note, during the Maven import process, the project file (pom.xml or other specified by the user) is analysed using the maven-embedder library. The parent POM files will be examined in the local repository or downloaded from network repositories. In order to reuse Maven's credentials for the network repositories, Bamboo requires internal access to the maven-embedder. If your JVM's or web container's SecurityManager settings are too strict, this process will fail and you will see an error similar to the following in your [Bamboo logs](#):

```
"Problem while initializing Maven Embedder. Probably Security Manager settings
are too strict, refer to
http://tomcat.apache.org/tomcat-6.0-doc/security-manager-howto.html"
```

If your imported Maven project file uses repositories that require credentials (i.e. not public or local), you will need to adjust the SecurityManager settings for your JVM or web container appropriately for the import to work.

- By default, all plans created by importing a Maven 2 project use the **Polling the Repository for changes trigger type**, but you can change this by [editing the plan](#). You can configure Bamboo to automatically start an initial build for every new plan. To do so, add the fire.initial.build.for.manual.strategy to your bamboo.cfg.xml file as described in [Configuring system properties](#).

Using plan branches

Plan branches are used to represent a branch in your version control repository, with the plan branch using the same build configuration as your plan.

Tools such as Git and Mercurial encourage a practice called [feature branching](#), where a developer can use a new branch to work in isolation from his or her team members before merging their changes back into main line development. Previously however, changes made on a branch may not have been built and tested by Bamboo unless the developer had specifically set up a new build plan, or had cloned an existing plan and configured it to build the new branch.

Now, with plan branches in Bamboo:

- Any new branch created in the repository can be automatically built and tested using the same build configuration as that of the parent plan.
- You have the flexibility to individually configure branch plans, by overriding the parent plan, if required.
- Optionally, changes from the feature branch can be automatically merged back to the "master" (e.g. trunk, default or mainline branch) when the build succeeds.

On this page:

- Activating plan branching
 - Auto branching
 - Manual branching
- Integrating branches with JIRA
- Branch notifications
- Branch dependencies
- Configuring branch plans
- Using automatic merging
 - Branch updater
 - Gatekeeper
- Limitations with plan branches
- Branches wallboard

Related pages:

- Disabling or deleting a plan
- Specifying the source repository
- Defining plan variables

Activating plan branching

When you activate plan branching, Bamboo automatically creates plan branches whenever the source repo is branched. You can also create a plan branch manually.

You can [override](#) the master plan's configuration in a branch plan, if required.

To see a list of branches for a plan, click on the branch icon beside a plan name on the **All Plans** tab of the dashboard. Select a branch name from the list to go directly to the summary page for that branch plan.

Screenshot: The Plan Summary page for a branch, showing the 'branches' menu.

Auto branching

You can use auto branching for Git, Mercurial and Subversion repositories. For other repository types, you can use [manual branching](#).

To have Bamboo automatically manage plan branches whenever the repo branches:

1. Go to the **Branches** tab in the configuration pages for the plan you wish to branch.
2. Select **Automatically manage branches**.
3. Enter a regular expression to specify the repo branch names for which plan branches will be created. An example is: (branch1|branch2|branch3)/. * See the [Java documentation on regular expressions](#).
4. Make the following optional settings as required. These will be applied to all branch plans created from this plan configuration, although they can be [overridden](#) in those branch plans, if required.

Remove after	Edit the value, in days, after which branches are automatically deleted, if no commits have been made to the VCS branch in that period. A value of zero prevents plans from being deleted.
Merging	Not available for Subversion. Check Branch Merging Enabled , and complete either the 'Branch updater' or 'Gatekeeper' sections, as described below .

JIRA Feature Branches	Check Create Remote Links from JIRA Issues to have the plan branch automatically linked, using an issue key in the branch name. Described below .
Notifications	Described below .
Branches Root	Only available for plans that use a Subversion source repository. Bamboo assumes that your Subversion repository structure follows the convention for branches, and automatically calculates the branch root URL. For example, for the fastBuild repo with this URL: <code>https://svn.mycompany.com/svn/fastBuild/trunk</code>, Bamboo will expect that branches will be created at this location: <code>https://svn.mycompany.com/svn/fastBuild/branches</code>. If your Subversion repository structure follows a different convention, you can specify where repository branches will be created by selecting Change branch root URL.

5. Click **Save**.

Manual branching

Use manual branching for all supported repository types. You may want to consider using [auto branching](#) for Git, Mercurial and Subversion repositories.

To manually create a branch of a plan:

1. Go to the **Branches** tab in the configuration pages for the plan you wish to branch.
2. Click **Create Branch**. Bamboo automatically checks for branches in the specified repository for the plan.
3. Select from the available VCS branches, then click **Create**.
4. You can [override](#) the default settings for the branch, such as the source repository used, if you wish.

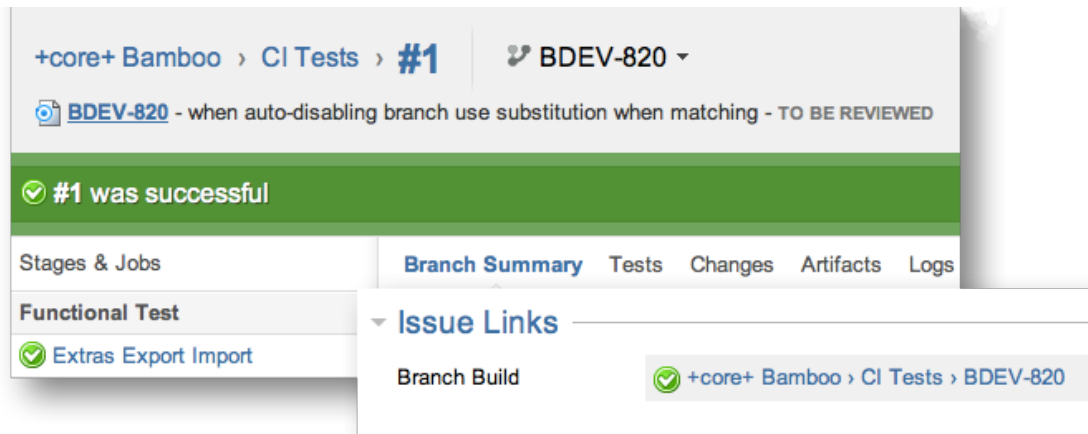
The screenshot shows the Bamboo Configuration interface for the 'Angry Nerds Master Plan'. The 'Configuration' tab is active, and the 'Branches' sub-tab is selected. On the left sidebar, 'Branches' is highlighted with 4 items. The main content area shows the 'Branches' configuration page. At the top right of this page, there is a 'Create Branch' button, which is pointed to by a red arrow. Below this button, there are sections for 'Automatically Manage Branches', 'Merging', 'JIRA Feature Branches' (with a checked option for 'Create Remote Links from JIRA Issues to Plan Branches'), and 'Notifications'.

Integrating branches with JIRA

When a developer begins working on a feature described in a JIRA issue, they use Git or Mercurial to branch the repository. If they use the issue key as part of the VCS branch name, Bamboo will detect the issue key and automatically link the new branch to the issue:

- The JIRA issue key needs to be in the name of the branch – 'jb-BDEV-790' and 'BDEV-769 1' are valid forms.
- The link shows up right under the breadcrumb on the Build Result Summary for the plan branch, and on the JIRA issue too.

To use JIRA Feature Branching, Bamboo needs an [application link](#) to the JIRA server.



Branch notifications

You can get build notifications from branch plans just as you do for master plans.

To specify how notifications are sent by all branches created from a plan, go to the **Branches** tab for the plan's configuration and choose one of the following options:

- Notify committers and people who have favoured this branch.
- Use the plan's notification settings.
- Notifications should not be sent for this branch.

You can [override](#) how notifications are sent from a particular branch plan, if necessary, by going to the **Notifications** tab on the Plan Branch configuration.

See [Configuring notifications for a plan and its jobs](#) for information about plan notifications.

Branch dependencies

You can use build dependencies for plan branches in a similar way to that for plans: a branch plan is triggered only when another branch plan has been successfully built. This can be used to ensure that breaking source code changes associated with one branch plan are detected before they can break the build of a dependent branch plan. Dependencies between master plans are maintained if their branch plans have the same name. See [Setting up plan build dependencies](#) for further information about dependencies.

Select **Trigger Dependencies for Branches**, on the **Dependencies** tab for the plan configuration, if you want plan branches to honour the build dependencies of their respective master plans.

Configuring branch plans

Whether a plan branch is created automatically or manually, the master plan maintains the structure and configuration of its branch plans. However, you can go to the configuration pages to override the following settings in a branch plan:

Branch clean-up	On the Branch Details tab of the branch's configuration, you can specify that a plan branch is <i>not</i> cleaned up automatically.  Please note that 'Automatic Branch Clean-up' is supported for Mercurial, Git (Bamboo 4.1.1 and above) and Subversion (Bamboo 4.2.0 and above).
Trigger type	On the Branch Details tab of the branch's configuration. See Triggering builds . Note that you can only configure one trigger for a plan branch, and that this overrides all triggers that may be configured for the master plan.
Merging	On the Branch Details tab of the branch's configuration. Described below.
Source repository	On the Source Repository tab of the branch's configuration. See Specifying the source repository .

Notifications	On the Notifications tab of the branch's configuration. The options are: • Notify committers and people who have favoured this branch. • Use the plan's notification settings. • Notifications should not be sent for this branch. See Configuring notifications for a plan and its jobs for information about plan notifications.
Variables	On the Variables tab of the branch's configuration. See Defining plan variables.

Using automatic merging

Bamboo provides 2 merging models if you choose to automate your branch merging:

- *Branch Updater* — a branch repo is kept up-to-date with changes to master.
- *Gatekeeper* — the default repo is only updated with changes in the branch that have built successfully.

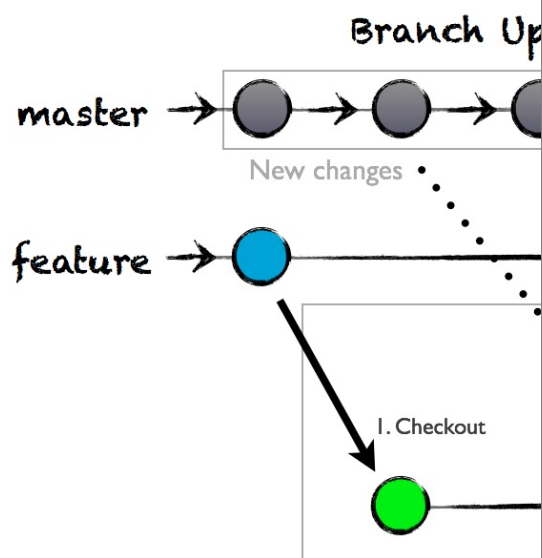
The automatic branch merge strategy for the master plan can be [overridden](#) in an individual plan branch, if required.

Branch updater

When to use

The Branch Updater should be used when you want to:

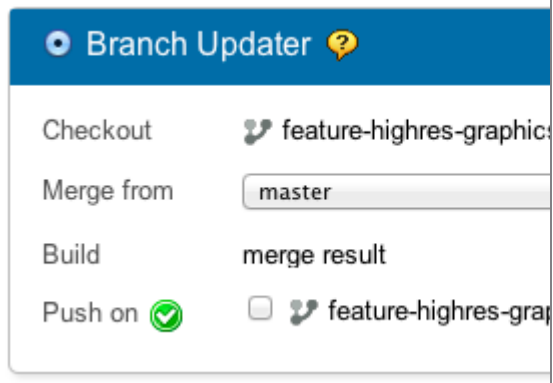
- Automatically merge changes from the team's master branch into your feature branch, after a successful build of the master branch.
- Get notified when the changes on your feature branch are no longer compatible with the team's master branch.



Configuring

To have recent changes in another repo merged into your branch repo:

1. Go to the **Branch Details** tab of the branch plan's configuration pages.
(Click on the branch icon beside a plan name on the **All Plans** tab, then choose **Actions > Configure Branch**.)
2. Under 'Merging' select **Branch Merging Enabled**, and then click **Branch Updater**.
3. Use the **Merge From** list to choose the repo from which changes should be merged with your feature branch.
4. Select **Push on** only if you want those changes merged back into your branch once the build completes successfully,
5. Click **Save**.



The screenshot shows the 'Branch Updater' configuration window. It has a blue header with the title 'Branch Updater' and a help icon. Below the header, there are four rows of configuration options:

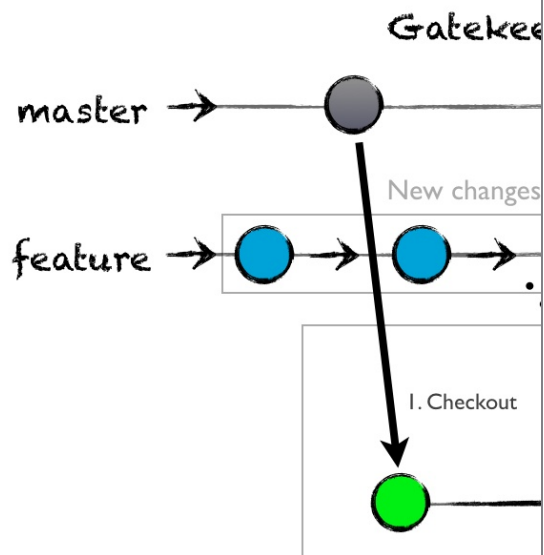
Checkout	 feature-highres-graphics
Merge from	master
Build	merge result
Push on 	☐  feature-highres-grap

Gatekeeper

When to use

The Gatekeeper should be used when you want to:

- Automatically merge your feature branch back into the team's master branch, after a successful build of the merged changes from both branches.
- Get notified when a build of combined changes from both branches fails, preventing the feature branch from being merged back into the team's master branch.



Configuring

To have your successfully built changes pushed to another repo:

1. Go to the **Branch Details** tab of the branch plan's configuration pages.
(Click on the branch icon beside a plan name on the **All Plans** tab, then choose **Actions > Configure Branch**.)
2. Under 'Merging' select **Branch Merging Enabled**, and then click **Gate Keeper**.
3. Use the **Checkout** list to choose the repo with which to merge your changes (and to which changes should be pushed).
4. Select **Push on** only if you want your changes pushed to the other repo once the build completes successfully,
5. Click **Save**.

Gatekeeper ?

Checkout

master

Merge from

🔗
feature-highres-graphics

Build
merge result

Push on ✔

☒ 🔗
master

Limitations with plan branches

The following limitations apply to using automated plan branching and merging:

Action	Limitations

Auto plan branching	• Can only be used with Git, Mercurial and Subversion repositories. For other repository types, use manual branching. • Cannot be used with the Git implementation embedded in Bamboo. (You need to have set up native Git.)
Manual plan branching	• Can be used for all repository types supported by Bamboo.
Auto branch merging	• Can only be used with Git and Mercurial repositories. • Can only be used with branches that were configured in Bamboo. • Cannot be used with the Git implementation embedded in Bamboo. (You need to have set up native Git.)

Branches wallboard

The branches wallboard displays the status of all the branches and the plan that the branches belong to. The plan's own status always appears first. Plans shown as grey are disabled.

To display the branches wallboard:

1. Go to the Plan Summary for the plan that has branches you want to display.
2. Choose **Actions** > **Branch Wallboard**.

× CI Tests  +core+ Bamboo 3 hours ago  Changes by Jason Berry	project-filters +core+ Bamboo - CI Tests  Changes by 4 people	× BAM-11636  +core+ Bamboo - CI Tests 1 week ago First build for this plan
× BAM-12221  +core+ Bamboo - CI Tests 1 day ago Manual build by Martin Henderson	× BAM-6483  +core+ Bamboo - CI Tests 2 weeks ago Changes by 11 people	✓ BDEV-1371  +core+ Bamboo - CI Tests 1 week ago Changes by Eduardo Garcia

Managing plans

A *plan* defines everything about your continuous integration build process in Bamboo. See [Configuring plans](#) for information about how to set up build plans.

You can also perform actions on one or more plans together, or make global settings that affect all plans on the Bamboo server.

See the following pages for information about managing your Bamboo plans:

- [Editing a plan's configuration](#)
- [Configuring a plan's permissions](#)
- [Disabling or deleting a plan](#)
- [Labelling a plan](#)
- [Modifying multiple plans in bulk](#)
- [Moving plans to a different project](#)
- [Configuring concurrent builds](#)
- [Configuring the hanging build event](#)
- [Configuring the build queue timeout event](#)
- [Disabling build monitoring](#)

Editing a plan's configuration

To edit an existing plan:

1. Click **Dashboard**, then the **All Plans** tab, then the name of the plan in the list, to get to the plan you want to edit.
2. Choose **Actions** > **Configure Plan**.
3. Click a tab to configure that aspect of your plan:
 - **Plan Details** — A plan's **Project Key** and **Plan Key** are not editable, but can be changed as described in [Moving plans to a different project](#).
 - **Source Repositories** — see [Specifying the source repository](#).
 - **Triggers** — see [Triggering builds](#).
 - **Branches** — see [Using plan branches](#).
 - **Stages** — see [Using stages in a plan](#).
 - **Dependencies** — see [Setting up build dependencies](#).
 - **Permissions** — see [Configuring a plan's permissions](#).
 - **Notifications** — see [Configuring notifications](#).

- **Variables** — see [Defining plan variables](#).
- **Miscellaneous** — see [Configuring expiry of a plan's build results](#).
- **Audit Log** — a record of changes to the plan's configuration. This feature is disabled by default. To enable it, please go to **Administration > System > Audit Log**.

Related pages:

- [Creating a plan](#)
- [Specifying the source repository](#)
- [Using plan branches](#)
- [Using stages in a plan](#)
- [Setting up build dependencies](#)
- [Configuring notifications for a plan and its jobs](#)
- [Configuring expiry of a plan's build results](#)
- [Configuring a plan's permissions](#)

Screenshot: A plan's configuration pages

Plan Configuration

Plan Details | Source Repositories | Triggers | Branches | Stages | Dependencies | Permissions

Stages & Jobs 6 | Branches 4

Notifications | Variables | Miscellaneous | Audit Log

Plan Details

You can update the names and description of the current project and plan.

Project Name*
How do you want to call the Project within Bamboo? e.g. "Issue Tracking Application".

Plan Name*
How do you want to identify the new Plan?

Plan Description
Choose a meaningful description for the new Plan. For example, "JIRA Release Plan".

☒ Plan Enabled

Configuring a plan's permissions

This page describes how to change the permissions for a particular plan. For ongoing ease of management, we recommend that you grant permissions to groups rather than to individual users.

You need to have 'Admin' permission on the plan to edit its permissions.

Note that a Bamboo Admin can also set [global permissions](#) for access to Bamboo.

To change plan permissions:

1. Click **Dashboard**, then the **All Plans** tab, then the name of the plan in the list, to get to the plan you want to edit.
2. Choose **Actions > Configure Plan**.
3. Click the **Permissions** tab.
4. Use the 'Grant permission to' section to add users or groups for which you wish to set permissions.
5. Select (or clear) the check box for each permission that you wish to change for a user or group. See the table below for details.
6. Click **Save**.

Related pages:

- [Editing a plan's configuration](#)
- [Granting plan permissions in bulk](#)
- [Managing permissions](#)
- [Creating a user](#)
- [Creating a group](#)

Plan permission	Actions
View	• View the plan and its builds • Add a comment or label to a build result
Edit	• Edit the configuration for a plan and its jobs (except for plan permissions and stages) • Delete a comment or label from a build result • Add and delete plan labels
Build	• Trigger a manual plan build • Pause and resume a plan build
Clone	• Clone the plan
Admin	• Edit the configuration for a plan and its jobs (including plan permissions and stages)

Permissions

You can edit your Plan permissions here. Permissions can be granted to specific users or groups.

Users	View	Edit	Build	Clone	Admin
Giancarlo Lionetti	☒	☒	☒	☒	☒

Other	View	Edit	Build	Clone	Admin
Logged in Users	☒	☐	☐	☐	☐
Anonymous Users	☒				

Grant permission to

Permission Types

- View** - User can view the Plan in Bamboo, including its builds.
- Edit** - User can view and edit the configuration of the Plan and its Jobs. This does not include the ability to change a Plan's permissions or its Stages.
- Build** - User can trigger a manual build on the Plan, as well as suspending and resuming the Plan.
- Clone** - User can clone the Plan.
- Admin** - User can administrate all components of this Plan including the Stages and the Plan's permissions.

Note:

To add new Jobs to a Plan, a user will also require the global 'Create Plan' permission. Users with the global 'Admin' permission have all of the above permissions for this Plan.

Disabling or deleting a plan

Bamboo allows you to disable or delete plans that you don't want to be built:

- **Disabling a plan** prevents it from being built. You can re-enable the plan, if you want to build it again. For example, if a plan's latest build is broken and cannot be fixed quickly, you may want to disable it temporarily to stop the plan from being built.
- **Deleting a plan** removes it completely from your Bamboo system. You will need to recreate a new plan from scratch, if you want to build it again. For example, if a plan is no longer relevant, you may want to delete it.

On this page:

- [Disabling a plan](#)
- [Deleting a plan](#)

Related pages:

- [Configuring plans](#)
- [Disabling or deleting a job](#)
- [Stopping an active build](#)
- [Exporting data for backup](#)

Disabling a plan

To disable a plan:

1. On the **All Plans** tab of the dashboard, click on the plan's name.
2. Choose **Actions > Disable Plan**.

You can also disable the plan using the **Plan Enabled** check box on the **Plan Details** tab of a plan's configuration pages.

Note that disabling a plan doesn't disable its branch plans.

Deleting a plan

Deleting a plan deletes everything related to that plan, including the plan's configuration, all of the plan's job configurations and the plan's branch plans, job build results, artifacts, labels and comments.

Before you begin:

- If you need to keep a permanent record of the job build results for your plan, see [Exporting data for backup](#).
- The **'Admin' global permission** is required to delete a plan.
- A plan that is currently being built cannot be deleted. If you need to delete such a plan, stop the plan's build first. Refer to [Stopping an active build](#) for more information.
- Deleting a plan also deletes its branch plans. Be careful!

To delete a plan:

There are two ways to delete a plan:

- From the dashboard:
 1. On the **All Plans** tab of the dashboard. click on the plan to delete.
 2. Choose **Actions > Configure Plan**.
 3. Choose **Actions > Delete Plan**.
- In the Administration Console:
 1. Click **Administration** in the top navigation bar.
 2. Click **Remove Plans** (under 'Plans') in the left navigation column.
 3. Select the plan you wish to delete.
 4. Click **Delete** at the bottom of the list. You will be prompted to confirm the deletion.

Labelling a plan

Bamboo allows you to label plans. Labelling a plan allows you to filter the plans displayed on the Dashboard or Wallboard. You may want to do this if you have set up a large number of plans in your Bamboo instance and want to highlight specific plans for attention.

For example, you may want to label all builds related to the release with a 'release' label. You can then filter your wallboard during your release, to display only these builds.

Before you begin:

- You must be logged in to Bamboo before you can label a plan.

Related pages:

- [Working with labels](#)
- [Using the Bamboo dashboard](#)
- [Displaying the wallboard](#)

Atlassian Blogs:

- [Making your Bamboo dashboard quicker and more relevant using plan labels](#)
- [Get to know Bamboo's build expiry and labels](#)

To label a build result:

1. Click **Dashboard**, then the **All Plans** tab, then the name of the plan in the list, to get to the plan you want to edit.
2. Click **Actions > Modify Plan Label**.
3. Type the relevant label (or multiple labels, separated by commas or spaces).
4. Click **Add**. Note that the label will be saved in lowercase characters.
5. Click **Close**.

Screenshot: Adding labels to a plan

The screenshot shows the 'Labels' dialog box in the Bamboo interface. The dialog has a title bar with tabs: 'Plan Summary', 'Recent Failures', 'History', 'Issues', 'Configuration', and a 'Run' button. The main content area is titled 'Labels' and contains a text input field with the text 'ci'. To the right of the input field is an 'Add' button. Below the input field, there are three tags: 'core', 'trunk', and 'test', each with a close button (X). At the bottom of the dialog, there is a 'Shortcut tip: Pressing L also opens this dialog box' and a 'Close' button.

Modifying multiple plans in bulk

Bulk actions allow you to make changes to multiple plans at once.

You need to be a Bamboo administrator to modify plans in bulk.

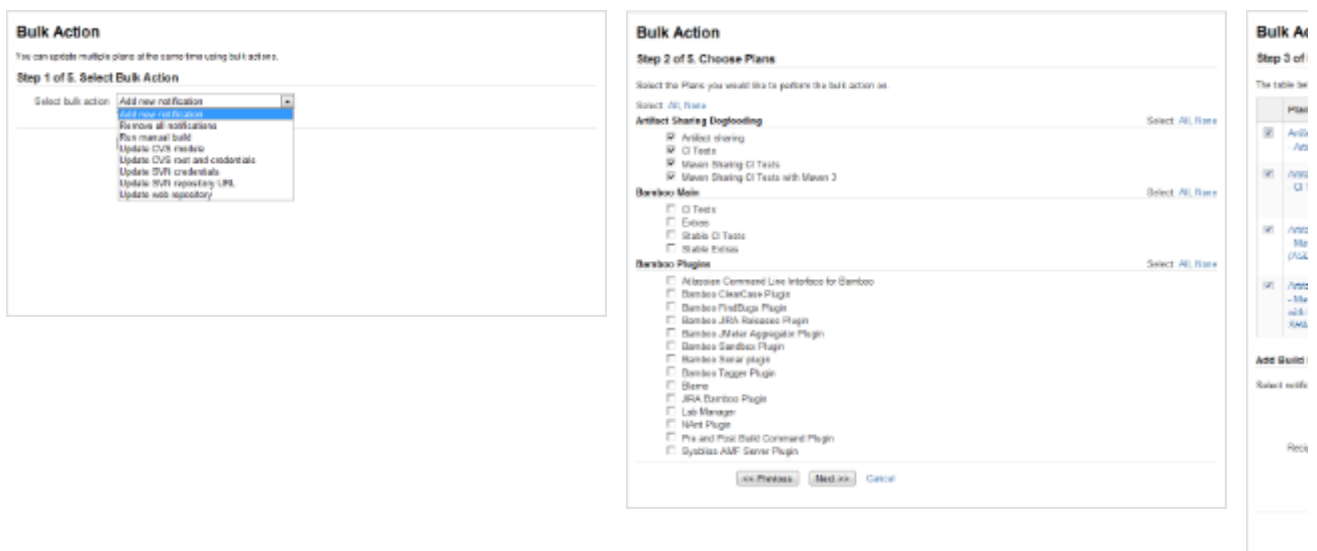
To use bulk actions:

1. Click **Administration** in the top navigation bar.
2. Click **Bulk Action** in the left-hand panel (under 'Plans').
3. Choose the required bulk action and follow the on-screen instructions to complete the 5 steps.

The following bulk actions are available:

Bulk Action	Notes
Add new notification	See Configuring notifications for a plan and its jobs for further details.
Remove all notifications	See Configuring notifications for a plan and its jobs for further details.
Disable Plan	See Disabling or deleting a plan for further details.
Enable Plan	
Run manual build	You have the option to disable dependencies when running the manual builds for the selected plans.
Update CVS module	See CVS documentation for further details.
Update CVS root and credentials	See CVS documentation for further details.
Update SVN credentials	See Subversion documentation for further details.
Update SVN repository URL	See Subversion documentation for further details.
Update web repository	See the Subversion , CVS or Perforce documentation for further details.
Update Maven 2 dependencies	You have the option for Bamboo to determine plan dependencies from your Maven pom.xml file, for all plans.

Screenshots: Adding a notification to multiple plans example (click to view full-sized images)



Step 1

Step 2

Moving plans to a different project

Moving a plan to a different project involves changing the plan's project key (as well as possibly the plan name and plan key), which will also change the build key for all of the plan's build results.

Moving a plan does not affect the plan's [configuration](#), nor any [comments](#) or [labels](#) that have been applied to job build results within the plan.

You need to be a Bamboo administrator to move a plan.

 Note that moving a plan will require Bamboo to re-index all its data, so your Bamboo system may run slowly for a few minutes.

Before you begin:

- We recommended that you back up your Bamboo build results before you move a plan. See [Exporting data for backup](#) for instructions.

To move a plan to a different project:

- Click **Administration** in the top navigation bar.
- Click **Move Plans** (under 'Plans') in the left-hand panel.
- Select either an existing project or **New Project** from the **Destination Project** list. For a new project, enter a new **Project Name** and a unique **Project Key**.
- Select one or more plans to move.
- Click **Move** to display the 'Configure New Plan Details' page (as shown in [Screenshot 2 below](#)).

6. Edit the new name and new key for each plan, if necessary. You may need to do this if the destination project already has a plan with the same plan name or key, or if you wish to change these.
7. Click **Move**.

Screenshot 1: 'Moving Plans - Select Plans'

Move Build Plan Wizard

 It is strongly recommended that you ensure that all agents are disabled before you perform the move. [Disable all agents](#)

Select Plans

You can move a plan to another project with this wizard. Simply select the plans you want to move and the destination project. As names and keys may conflict, you'll then be asked to enter new names and keys for the plans. Note that because we are changing plan keys, this operation requires some slow operations (e.g. indexing of all builds) and may take a few minutes.

Destination Project

The project you want to move your plans to

Project Name*

How do you want to call the Project within Bamboo? e.g. "Issue Tracking Application".

Project Key*

This is the unique Project key to identify a Project. The key must contain only uppercase alphanumeric characters. e.g. "ITA".

Select: [All](#), [None](#)

Artifact Sharing Dogfooding

Select: [All](#), [None](#)

- ☐ Artifact sharing
- ☒ CI Tests
- ☐ Maven Sharing CI Tests
- ☐ Maven Sharing CI Tests with Maven 3

Bamboo Main

Select: [All](#), [None](#)

- ☐ CI Tests
- ☐ Extras
- ☒ Stable CI Tests
- ☐ Stable Extras

UI

Select: [All](#), [None](#)

- ☐ Default
- ☐ MC Test
- ☐ WebDriver Tests
- ☐ WebDriver Tests JDK 1.6

Screenshot 2: Moving Plans - Choose new build keys and build names

Configure New Plan Details

Choose new build keys and build names

The following plans will be moved to project A New Project. You can update the plan names and keys below.

Original Project	Original Name	New Name	Original Key	New Key
Artifact Sharing Dogfooding (ASD)	CI Tests	CI Tests	CI	CI
Bamboo Main (BAM)	Stable CI Tests	Stable CI Tests	CITESTSTABLE	CITESTSTABLE

[Cancel](#)

Configuring concurrent builds

Bamboo's concurrent builds feature allows you to build a plan concurrently on several agents. You might find this useful if a plan is likely to be [triggered](#) again before the current build completes.

You can configure a default value for the maximum number of builds of a plan that your Bamboo server can run concurrently, using the Bamboo administration console. This value is a default – it can be overridden on the **Miscellaneous** tab of a plan's configuration. You need to be a Bamboo [administrator](#) to configure concurrent builds.

To configure the number of concurrent builds of a plan allowed by Bamboo:

1. Click **Administration** in the top navigation bar.
2. Click **Concurrent Builds** in the left panel (under 'Plans'), then click **Enable**.
3. Click **Edit**.
4. Edit the value for **Default number of concurrent builds allowed**.
5. Click **Save**.

Configuring the hanging build event

The hanging build event is thrown when Bamboo determines that a build has become unresponsive according to two criteria:

- Expected Build Time — defined as Build Time Multiplier x Average Build Time
 - Build Time Multiplier is a user-defined setting.
 - Average Build Time is calculated by Bamboo using an average of previous build times (in minutes).
- Log Quiet Time — the length of time (in minutes) between log entries for a build.

The Expected Build Time and Log Quiet Time must *both* be exceeded for Bamboo to throw a hanging build event.

This event is currently used by Bamboo to send [notifications](#).

You can also [disable build monitoring](#) altogether so that the hanging build event never occurs.

On this page:

- [Configure the hanging build event](#)
- [The check interval for hung builds](#)

Related pages:

- [Configuring notifications for a plan and its jobs](#)
- [Configuring the build queue timeout event](#)
- [Disabling build monitoring](#)
- [Configuring tasks](#)

Configure the hanging build event

You can change the criteria governing when a hanging build event is thrown.

Note, the hanging build criteria can be also be set for a specific job, when [specifying a job's builder](#). Job-level criteria will override the global criteria described on this page (including disabling this event).

To edit the hanging build event settings:

1. Click **Administration** in the top navigation bar.
2. Click **Build Monitoring** (under 'Plans') in the left panel.
3. Click **Edit** and update the values for **Build Time Multiplier** and **Log Quiet Time** as required.
4. Click **Save**.

Screenshot: Editing the hanging build event settings

Build Monitoring

Default Build Monitoring Criteria

You can configure the default settings for detecting hanging builds on this page. Hanging build detection can also be disabled altogether on this page. You can override these settings for individual plans in the builder configuration of each plan.

Bamboo determines that a build has hung, if **both** of the criteria below are exceeded.

- Expected Build Time - calculated as the average build time * build time multiplier
- Log Quiet Time - the length of time Bamboo goes without receiving any log messages for that build

Build Time Multiplier
Multiplier to be applied to the average build time (e.g. 2.5)

Log Quiet Time
The length of time (in whole minutes) without logs being received (e.g. 10 minutes)

You can configure also default settings for detecting build waiting in build queue longer than specified time. Bamboo determines that build time has been exceeded if build waits in the queue longer specified timeout.

- Build Queue Timeout - the length of time build waits in the queue before notification

Build Queue Timeout
The length of time (in whole minutes) before queue timeout would be detected (e.g. 60 minutes)

The check interval for hung builds

By default, Bamboo will check whether a hanging build event has been thrown every 60 seconds.

You can change this check interval by configuring the system property, `bamboo.buildHangingMonitor.checkInterval`. (This property is specified in seconds.)

Please read [Configuring system properties](#) for instructions on how to configure the `bamboo.buildHangingMonitor.checkInterval` system property.

Configuring the build queue timeout event

The build queue timeout event is thrown when a build has been waiting in the build queue for longer than a specified period of time.

This event is currently used by Bamboo to send [notifications](#).

Configuring the build queue timeout event

You can change the criteria governing when the build queue timeout event is thrown. You can also [disable build monitoring](#) altogether so that the build queue timeout event never occurs.

On this page:

- [Configuring the build queue timeout event](#)
- [Disabling the build queue timeout event](#)
- [The check interval for build queue timeouts](#)

Related pages:

- [Configuring notifications for a plan and its jobs](#)
- [Disabling build monitoring](#)

To edit the build queue timeout event settings:

1. Click **Administration** in the top navigation bar.
2. Click **Build Monitoring** (under 'Plans') in the left panel.
3. Click **Edit** and update the value for **Build Queue Timeout** as required.
4. Click **Save**.

Screenshot: Editing build queue timeout event settings

Build Monitoring

Default Build Monitoring Criteria

You can configure the default settings for detecting hanging builds on this page. Hanging build detection can also be disabled altogether on this page. You can override these settings for individual plans in the builder configuration of each plan.

Bamboo determines that a build has hung, if **both** of the criteria below are exceeded.

- Expected Build Time - calculated as the average build time * build time multiplier
- Log Quiet Time - the length of time Bamboo goes without receiving any log messages for that build

Build Time Multiplier
Multiplier to be applied to the average build time (e.g. 2.5)

Log Quiet Time
The length of time (in whole minutes) without logs being received (e.g. 10 minutes)

You can configure also default settings for detecting build waiting in build queue longer than specified time. Bamboo determines that build time has been exceeded if build waits in the queue longer specified timeout.

- Build Queue Timeout - the length of time build waits in the queue before notification

Build Queue Timeout
The length of time (in whole minutes) before queue timeout would be detected (e.g. 60 minutes)

Disabling the build queue timeout event

You can disable the build queue timeout event by disabling build monitoring for your Bamboo installation. See [Disabling build monitoring](#).

Please note, you cannot disable the build queue timeout event without disabling all build monitoring features for your Bamboo installation.

The check interval for build queue timeouts

By default, Bamboo will check whether a build queue timeout event has been thrown every 60 seconds.

You can change this by configuring the system property, `bamboo.buildQueueMonitor.checkInterval`. (This property is specified in seconds.)

Please read [Configuring system properties](#) for instructions on how to configure the `bamboo.buildQueueMonitor.checkInterval` system property.

Disabling build monitoring

To disable build monitoring:

1. Click **Administration** in the top navigation bar.
2. Click **Build Monitoring** in the left panel.
3. Click **Disable**. This will disable all build monitoring for your Bamboo installation, including the build hanging event and build queue timeout notifications. It is not possible to disable build monitoring features separately.

Screenshot: Disabling build monitoring

Build Monitoring

Hanging Build Detection

You can configure the default settings for detecting hanging builds on this page. Hanging build detection can also be disabled altogether on this page. You can override these settings for individual plans in the builder configuration of each plan.

Bamboo determines that a build has hung, if **both** of the criteria below are exceeded.

- Expected Build Time - calculated as the average build time * build time multiplier
- Log Quiet Time - the length of time Bamboo goes without receiving any log messages for that build

Build Queue Timeout Detection

You can configure also default settings for detecting build waiting in build queue longer than specified time. Bamboo determines that build time has been exceeded if build waits in the queue longer specified timeout.

- Build Queue Timeout - the length of time build waits in the queue before notification

Default Build Monitoring Criteria

Build Time Multiplier **2.5**

Log Quiet Time **20**

Build Queue Timeout **5**

Edit

Disable

Connecting to code repositories

A core part of setting up your continuous integration build process in Bamboo is to specify the code repositories that Bamboo should work with.

- When you [create a new plan](#), the source repository you specify becomes the default. It is used by the plan's 'Default Job' and can be used by other jobs added to this plan.
- You can specify additional repositories for a Bamboo plan to work with, perhaps for tasks in later stages of the build. See [Checking out code](#).
- You can set up [shared source repositories](#) that are then available globally to all plans and jobs configured on the Bamboo server. Doing this can save you from having to reconfigure the source repositories in multiple places if these ever change. Changes to a shared repository are applied to every plan or job that uses the repository.

Bamboo is able to connect to a variety of SCMs; for details regarding a particular repository type, please refer to [Specifying the source repository](#) and the pages listed below:

- [Bitbucket - Mercurial](#)
- [CVS](#)
- [Git](#)
- [GitHub](#)
- [Mercurial](#)
- [Perforce](#)
- [Subversion](#)

Specifying the source repository

When you [create a new plan](#), you specify the default source repository to be used for your plan builds. Bamboo checks out the code from the repository before performing all the subsequent tasks for the build.

Bamboo is able to connect to a variety of version control systems. For details, please refer to the following pages:

- [Bitbucket - Mercurial](#)
- [CVS](#)
- [Git](#)

- [GitHub](#)
- [Mercurial](#)
- [Perforce](#)
- [Subversion](#)

On this page:

- [Viewing the source repository for a plan](#)
- [Viewing the source repository for a job](#)

Related pages:

- [Connecting to code repositories](#)
- [Checking out code](#)
- [Configuring a shared source repository](#)
- [Configuring plans](#)

If you are a Bamboo server administrator, you can [configure a shared source repository](#). Shared repositories are available to all plans on the server.

Note that if you need to use an unsupported type of repository, a number of [third-party Source Repository plugin modules](#) are available (e.g. the ClearCase plugin). You can also write a [Source Repository Module plugin](#) to enable Bamboo to connect to your repository.

Screenshot: Configuring a job — Subversion Source Repository

The screenshot displays the 'Source Code Checkout' configuration page in Bamboo. The left sidebar shows a list of tasks, with 'Source Code Checkout' selected. The main area is titled 'Source Code Checkout Configuration'. It includes a 'Task Description' field with the text 'Checkout Default Repository'. Below this is a paragraph explaining that you can check out one or more repositories, choosing between the Plan's Default Repository or a Specific Repository. A 'Repository' dropdown menu is set to 'Bitbucket'. Below the dropdown is a note: 'Default always points to Plan's default repository.' There is a 'Checkout Directory' field with a placeholder text: '(Optional) Specify an alternative sub-directory to which the code will be checked out.' A checkbox labeled 'Force Clean Build' is present, with a description: 'Removes the source directory and checks it out again prior to each build. This may significantly increase build times.' At the bottom, there is an 'Add Repository' button with a plus icon. The page also shows a 'Save' button and a 'Cancel' button.

Viewing the source repository for a plan

To navigate to the source repository settings for a plan:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list and click its  icon. The plan's configuration pages will be displayed.
3. Click the **Source Repositories** tab to see all the repositories configured for this plan.
4. Click the name of a repository to see its particular settings (see [screenshot below](#)).

Screenshot: Viewing the details for a source repository of a plan

Plan Details
Source Repositories
Dependencies
Permissions
Notifications
Variables
Miscellaneous
Audit Log
Run
Actions

Source Repositories

One or more repositories can be added to this Plan, which will be available to every Job in the Plan. The first repository in the list is the Plan's default repository. The repositories to which the Build Strategy applies can be selected in the [Plan Details configuration](#).

- Bitbucket (default)
- Angry Nerds Tests

Add Repository

Source Repository

Bitbucket

Note: the Bitbucket repository currently supports Mercurial only. To connect to a Git repository on Bitbucket, choose Git from the Source Repository drop-down.

ID*

Angry Nerds Tests

The ID identifies the repository when you are using multiple repositories in a Plan.

Username*

glionetti

The Bitbucket user required to access the repositories.

Change password?

Repository

☒ My Repositories
☐ URL

glionetti/tets-project

Load Repositories

Select the repository you want to use for your Plan.

Branch

default

Choose a branch you want to check out your code from.

Advanced Options

Save Repository
Cancel

Viewing the source repository for a job

To navigate to the source repository settings for a job:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list which contains the job you wish to configure and click the plan's name. The plan's 'Plan Summary' page will be displayed.
3. Choose **Actions** > **Configure Plan**.
4. Click the name of the job in the 'Plan Navigator' on the left. The job's Summary will be displayed.
5. Click the **Tasks** tab and then the **Source Code Checkout** task, to see the repository settings for your job. (Note that this may not be present in your job configuration if it has been explicitly removed by the user.)

For a description of configuring the Source Code Checkout task see [Checking out code](#).

Bitbucket - Mercurial

This page describes how to configure Bamboo to use a Bitbucket Mercurial source repository for either a plan or a job.

Before you start:

- You will not be able to create plans or jobs that use a Bitbucket Mercurial repository without specifying the shared local Mercurial capability first. Read more about [configuring a Version Control capability](#).

Related pages:

- [Specifying the source repository](#)
- [Mercurial](#)

Configuring a Bitbucket Mercurial source repository

To add a new Bitbucket repository, navigate to the source repository settings for a plan or job, as described on [Specifying the source repository](#).

1. Either click **Add Repository** to add a new repository, or edit an existing repository configuration.
2. Choose **Bitbucket** from the **Source Repository** list.
3. Enter a **Display Name** to help identify the repository in Bamboo.
4. Add your Bitbucket **Username** and **Password**.
5. You can configure the following settings for a BitBucket source repository for your plan:

Repository	Retrieves all repositories you have explicit permissions to access from Bitbucket when you click Load Repositories .
Branch	Pick a branch if you want to check out code from a branch other than the default branch.

Advanced Options

Command timeout	This is useful to stop hung Bitbucket processes. On slower networks, you may consider increasing the default timeout to allow Bamboo time to make an initial clone of the Mercurial repository.

Verbose logs	Turns on --verbose and --debug options in hg commands and passes the output to build logs. Use this option if you encounter problems with Mercurial in Bamboo.
Enable Quiet Period	Specifies a delay after a single commit is detected before the build is started. This allows multiple commits to be aggregated into a single build.
Include/Exclude Files	Allows you to specify the files that Bamboo should, or should not, use to detect changes. Enter into File Pattern a regular expression to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See sub page for examples.
Exclude Changesets	Enter a regular expression to match the commit messages for changesets that should not start a build.
Web Repository	If your repository can be viewed in a web browser, select the repository type. This allows links to relevant files to be displayed in the 'Code Changes' section of a build result. Mercurial Web Repository – select one of the following viewer schemes: • BitBucket Web Repository Scheme (if you use BitBucket) • Default Web Repository Scheme (hgserve) (Mercurial's own default web server) Stash – specify the following details for the repository: • Stash URL – the URL of your Stash instance (e.g. 'https://stash.mycompany.com'). • Stash Project Key – the key of the project in Stash (e.g. 'CONF'). • Repository Name – the name of the repository in Stash (e.g. 'conf-dev'). See Integrating Bamboo with Stash for more information. FishEye – specify the following details for the repository: • FishEye URL – the URL of your FishEye repository (e.g. 'https://atlassian.atlassian.com/'). • Repository Name – the name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path. • Repository Path – the path for your FishEye repository (e.g. '/atlassian/bamboo/'). See Integrating Bamboo with FishEye for more information.  How do I determine my Repository Path? If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

CVS

The instructions on this page describe how to configure Bamboo to use a CVS source repository for either a plan or a job.

Related pages:

- [Specifying the source repository](#)

Configuring a CVS source repository

Navigate to the source repository settings for a plan or job, as described on [Specifying the source repository](#).

1. Either click **Add Repository** to add a new repository, or edit an existing repository configuration.
2. Choose **CVS** from the **Source Repository** list.
3. Enter a **Display Name** to help identify the repository in Bamboo.
4. You can configure the following settings for a CVS source repository for your plan:

CVS Root	The full path to your CVS repository root (e.g. <code>:pserver:me@cv.s.atlassian.com:/cvsroot/atlassian</code>). Bamboo supports pserver, ext (ssh) and local repository access methods. Note that you can use global variables in this field (see Bamboo variables). i If you are importing a Maven 2 Project, this location should contain your project's <code>pom.xml</code> file.
Authentication Type	Password – choose this option if you want to authenticate with a password. SSH – if you choose to authenticate using SSH, you need to provide the following details: • Private Key — the absolute path of your SSH private key. • Passphrase — the passphrase for your SSH private key.
Quiet Period	This setting is used to avoid starting a build while someone is in mid-checkin. Bamboo will only initiate a build for this plan when no more changes are detected within the Quiet Period following the last known change. Type the number of seconds Bamboo should wait. Please note that this parameter is mandatory for CVS, as CVS allows partial checkouts. <i>(Only available when configuring an existing plan)</i>
Module	Type the name of the CVS module that contains the source-code. i Currently Bamboo has limited support for CVS ampersand modules. To use an ampersand module, you will need to define a regular module with the same name as the ampersand module (since Bamboo expects there to be a directory with the specified checkout module name). For example: a. Create a module (e.g. <code>allbuilds</code>). b. Define an ampersand module with the same name. (The ampersand module can be empty.) c. In the Module field, enter the following: <code>allbuilds allbuilds &project2 &project2 &project3</code>
Version of module	The version of the module that Bamboo should build: • HEAD • Branch/Tag – supply the name of the branch or tag. i Note that you can use global variables in this field (see Bamboo variables).

Advanced Options

Include/Exclude Files	Allows you to specify the files that Bamboo should, or should not, use to detect changes. Enter into File Pattern a regular expression to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See sub page for examples.
Exclude Changesets	Enter a regular expression to match the commit messages for changesets that should not start a build.
Web Repository	If your repository can be viewed in a web browser, select the repository type. This allows links to relevant files to be displayed in the 'Code Changes' section of a build result.

Generic Web Repository

- **Web Repository URL** – the URL of the repository.
- **Web Repository Module** — the particular repository required for this plan or job, if the **Web Repository URL** above points to multiple repositories.

Stash – specify the following details for the repository:

- **Stash URL** – the URL of your Stash instance (e.g. 'https://stash.mycompany.com').
- **Stash Project Key** – the key of the project in Stash (e.g. 'CONF').
- **Repository Name** – the name of the repository in Stash (e.g. 'conf-dev').

See [Integrating Bamboo with Stash](#) for more information.

FishEye

- **FishEye URL** — the URL of your FishEye repository (e.g. 'https://atlaseye.atlassian.com/').
- **Repository Name** — the name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path.
- **Repository Path** — the path for your FishEye repository (e.g. '/atlassian/bamboo/').

See [Integrating Bamboo with FishEye](#) for more information.



How do I determine my Repository Path?

If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to [view the code changes](#) and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when [browsing a repository](#) (e.g. trunk)). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

Configuring source code management triggers for CVS

This page provides instructions on how to configure CVS to send message events that trigger the execution of Bamboo plans.

You only need to configure CVS to send these message events if **The repository triggers the build when changes are committed** trigger has been configured for one or more of your Bamboo plans.

Configuring CVS to trigger a build

This section explains how to configure CVS to trigger a build when the repository is changed. This involves installing two scripts:

1. A pre-commit trigger keeps track of the last directory to be processed, so we know when the commit has completed.
2. A post-commit trigger that waits until it has processed the last directory of the commit before instructing the Bamboo server to execute the relevant plan(s).

On this page:

- [Configuring CVS to trigger a build](#)
- [Notes](#)

Related pages:

- [CVS](#)
- [Configuring source code management triggers for Subversion](#)



The following commands and script files assume that your CVS server runs on a UNIX- or Linux-based operating system. If your CVS server runs on any other operating system, then you will need to modify the script files and if necessary, the commands below to suit that operating system.

Step 1. Checking out the CVSROOT

First check out your repository's CVSROOT directory into a temporary directory:


```
cvs -d cvsroot-to-your-repository checkout CVSR00T
```

where:

- cvsroot-to-your-repository is the root directory pathname of the CVS repository.
-  Using -d cvsroot-to-your-repository overrides the any \$CVSR00T environment variable setting.

The following files should be checked out:

```
U CVSR00T/checkoutlist
U CVSR00T/commitinfo
U CVSR00T/config
U CVSR00T/cvswrappers
U CVSR00T/editinfo
U CVSR00T/logininfo
U CVSR00T/modules
U CVSR00T/notify
U CVSR00T/rcsinfo
U CVSR00T/taginfo
U CVSR00T/verifymsg
```

Step 2. Install the pre-commit trigger

Add a line like the following example's to the CVSR00T/commitinfo pre-commit trigger file. The CVSR00T/commitinfo file contains the list of programs to run whenever a file is about to be committed to the repository.

```
^Moo /path-to-your-bamboo-installation/scripts/cvs-triggers/preCommit.sh
%r/%p
```

where:

- ^Moo is the regular expression used to identify the name of the module (called Moo) being updated.
- /path-to-your-bamboo-installation/scripts/cvs-triggers/preCommit.sh is the Bamboo shell script used to detect the last file of the check in.
-  If your Bamboo installation and CVS server are on different machines, refer to the [note below](#).

Step 3. Install the post-commit trigger

Add a line like the following example's to the CVSR00T/logininfo post-commit trigger file. The CVSR00T/logininfo file contains the list of programs to run whenever a file has been successfully committed into the repository.

```
^Moo
/path-to-your-bamboo-installation/scripts/cvs-triggers/postCommitBuildTrigger.sh %r/%p http://bamboo-base-url M00-KEY
```

where:

- ^Moo is the regular expression used to identify the name of the module (called Moo) being updated.
- /path-to-your-bamboo-installation/scripts/cvs-triggers/postCommitBuildTrigger.sh is the Bamboo shell script to trigger the build.
-  If your Bamboo installation and CVS server are on different machines, refer to the [note below](#).
- %r/%p is how CVS tells the postCommitBuildTrigger.sh script which directory it is committing.
- http://bamboo-base-url is the URL of the Bamboo server.
- M00-KEY the key of the Bamboo plan to be executed.

Step 4. Save the changes back to CVS

Commit the changes you made to the CVSR00T/commitinfo and CVSR00T/logininfo files in step 2 and 3, respectively, back to the repository.

```
cvs -d cvsroot-to-your-repository commit
```

where:

- `cvsroot-to-your-repository` is the root directory pathname of the CVS repository.



Using `-d cvsroot-to-your-repository` overrides the any `$CVSROOT` environment variable setting.

Step 5. Do a test commit

Conduct a 'test' commit. Bamboo should start building the relevant plan after a few seconds.

The Bamboo log file should contain an entry like this:

```
[INFO] com.atlassian.bamboo.build.UpdateAndBuild - Bamboo build was triggered by remote http call from 127.0.0.1
```



- The `postCommitBuildTrigger.sh` is only triggered when the last file of the commit has been committed.
- The `preCommit.sh` and `postCommitBuildTrigger.sh` must have sufficient privileges to be executed by the CVS user.
- If your Bamboo installation is not running on the same machine as the CVS server, you will also need add the Bamboo `preCommit.sh` and `postCommitBuildTrigger.sh` files to the `CVSROOT` directory and add the names of these files to the end of the `checkoutList` file.

Notes

Build Trigger Security — Bamboo will only accept remote build triggers if the triggers originated from the CVS server(s) identified in the **CVS root** paths of any Bamboo plans. Requests originating from other CVS servers will be rejected by Bamboo.

Git

This page describes how to configure Bamboo to use a Git source repository.

You need to have previously defined a Git capability before you can configure a Git source repository – see [Defining a new version control capability](#).



Note that Bamboo comes with its own built-in Git implementation. However, you need to use native Git to be able to use symbolic links, submodules, automatic branch detection and automatic merging - these are not supported by the built-in Git.

You can download Git from the following locations:

- Windows: <http://code.google.com/p/msysgit/downloads/list?can=3&q=official+Git>
- Linux and Mac: <http://git-scm.com/download>

Related pages:

- [Specifying the source repository](#)
- [GitHub](#)
- [Defining a new version control capability](#)

Configuring a Git source repository

To add a new Git repository, navigate to the source repository settings for a plan or job, as described on [Specifying the source repository](#).

1. Either click **Add Repository** to add a new repository, or edit an existing repository configuration.
2. Choose **Git** from the **Source Repository** list.
3. Enter a **Display Name** to help identify the repository in Bamboo.
4. You can configure the following settings for a Git source repository for your plan:

Repository URL	The full path to your Git repository (eg: <code>git://github.org/atlas</code>) Valid URLs are of the form: • <code>git://host.xz[:port]/path/to/repo.git</code> • <code>ssh://[user@]host.xz[:port]/path/to/repo.git</code> • <code>[user@]host.xz[:port]/path/to/repo.git</code> • <code>http[s]://host.xz[:port]/path/to/repo.git</code> • <code>/path/to/repo.git</code> • <code>file:///path/to/repo.git</code>
Branch	Type the name of the relevant branch (or tag) you want to work on. Leave empty to work on the master branch.

Authentication Type	None – choose none if you want to access the repository anonymously.
	Username/password – authenticate with a username and password.
	SSH private key – upload an SSH Key and provide the corresponding SSH Passphrase .
Use shallow clones	Allows Bamboo to perform shallow clones (i.e. history truncated to a specified number of revisions). This should increase the speed of the initial code checkouts, however if your build depends on the full repository history, we recommend that you do not use this option. Shallow clones are enabled by default.
Location of POM file	The path to your project's pom.xml file, relative to the root of your Git Repository URL (defined above). (Only available when importing a Maven 2 project)

Advanced Options

Use submodules	Select to enable submodules support if these are defined for the repository. If native Git capability is not defined for agent submodules support will be disabled.
Command timeout	This is useful to stop hung Bitbucket processes. On slower networks, you may consider increasing the default timeout to allow Bamboo time to make an initial clone of the Git repository.
Verbose logs	Turns on --verbose and --debug options in hg commands and passes the output to build logs. Use this option if you encounter problems with Git in Bamboo.
Enable Quiet Period	Specifies a delay after a single commit is detected before the build is started. This allows multiple commits to be aggregated into a single build.
Include/Exclude Files	Allows you to specify the files that Bamboo should, or should not, use to detect changes. When you configure the Include option, it means that you want Bamboo to use only the mentioned files for change detection because by default Bamboo checks all the files. The same way, if you configure the Exclude option, Bamboo will not consider the excluded files for detecting changes. Enter into File Pattern a regular expression to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See sub page for examples.
Exclude Changesets	Enter a regular expression to match the commit messages for changesets that should not start a build.
Web Repository	If your repository can be viewed in a web browser, select the repository type. This allows links to relevant files to be displayed in the 'Code Changes' section of a build result.
	Stash – specify the following details for the repository: • Stash URL – the URL of your Stash instance (e.g. 'https://stash.mycompany.com'). • Stash Project Key – the key of the project in Stash (e.g. 'CONF'). • Repository Name – the name of the repository in Stash (e.g. 'conf-dev'). See Integrating Bamboo with Stash for more information.

FishEye – specify the URL and other details for the repository:

- **FishEye URL** — the URL of your FishEye repository (e.g. ' <https://atlaseye.atlassian.com/> ').
- **Repository Name** — the name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path.
- **Repository Path** — the path for your FishEye repository (e.g. '/atlassian/bamboo/').

See [Integrating Bamboo with FishEye](#) for more information.



How do I determine my Repository Path?

If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to [view the code changes](#) and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when [browsing a repository](#) (e.g. trunk)). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

GitHub

The instructions on this page describe how to configure Bamboo to use a GitHub source repository.

Related pages:

- [Specifying the source repository](#)
- [Git](#)

Configuring a GitHub source repository

To add a new GitHub repository, navigate to the source repository settings for a plan or job, as described on [Specifying the source repository](#).

1. Either click **Add Repository** to add a new repository, or edit an existing repository configuration.
2. Choose **GitHub** from the **Source Repository** list.
3. Enter a **Display Name** to help identify the repository in Bamboo.
4. Enter your GitHub **Username** and **Password**.
5. Click **Load Repositories**.
6. You can configure the following advanced options for a GitHub source repository for your plan:

Advanced Options

Use submodules	Select to enable submodules support if these are defined for the repository. If native GitHub capability is not defined for agent submodules support will be disabled.
Command timeout	This is useful to stop hung Bitbucket processes. On slower networks, you may consider increasing the default timeout to allow Bamboo time to make an initial clone of the GitHub repository.
Verbose logs	Turns on <code>--verbose</code> and <code>--debug</code> options in hg commands and passes the output to build logs. Use this option if you encounter problems with GitHub in Bamboo.
Enable Quiet Period	Specifies a delay after a single commit is detected before the build is started. This allows multiple commits to be aggregated into a single build.
Include/Exclude Files	Allows you to specify the files that Bamboo should, or should not, use to detect changes. Enter into File Pattern a regular expression to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See sub page for examples.
Exclude Changesets	Enter a regular expression to match the commit messages for changesets that should not start a build.

Web Repository

If your repository can be viewed in a web browser, select the repository type.

This allows links to relevant files to be displayed in the ['Code Changes'](#) section of a build result.

Stash – specify the following details for the repository:

- **Stash URL** – the URL of your Stash instance (e.g. 'https://stash.mycompany.com').
- **Stash Project Key** – the key of the project in Stash (e.g. 'CONF').
- **Repository Name** – the name of the repository in Stash (e.g. 'conf-dev').

See [Integrating Bamboo with Stash](#) for more information.

FishEye – specify the URL and other details for the repository:

- **FishEye URL** – the URL of your FishEye repository (e.g. 'https://atlasseye.atlassian.com/').
- **Repository Name** – the name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path.
- **Repository Path** – the path for your FishEye repository (e.g. '/atlassian/bamboo/').

See [Integrating Bamboo with FishEye](#) for more information.



How do I determine my Repository Path?

If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to [view the code changes](#) and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when [browsing a repository](#) (e.g. trunk)). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

Mercurial

This page describes how to configure Bamboo to use a Mercurial source repository.

Before you start:

- **Please use Mercurial 2.1.1 or later.** Mercurial 2.1 has a [bug](#) that makes it incompatible with Bamboo.
- You will not be able to create plans or jobs that use a Mercurial repository without specifying the shared local Mercurial capability first. Read more about [configuring a Version Control capability](#).

Related pages:

- [Specifying the source repository](#)
- [Bitbucket - Mercurial](#)

Configuring a Mercurial source repository

To add a new Mercurial repository, navigate to the source repository settings for a plan or job, as described on [Specifying the source repository](#).

1. Either click **Add Repository** to add a new repository, or edit an existing repository configuration.
2. Choose **Mercurial** from the **Source Repository** list.
3. Enter a **Display Name** to help identify the repository in Bamboo.
4. You can configure the following settings for a Mercurial source repository for your plan:

Repository URL	The full path to your Mercurial repository (eg: git://bitbucket.org/atlassian/bamboo-git-plugin.git) Valid URLs are of the form: • local/filesystem/path[#revision] • file://local/filesystem/path[#revision] • http[s]://[user[:pass]@]host[:port]/[path][#revision] • ssh://[user[:pass]@]host[:port]/[path][#revision] (for further references visit Mercurial documentation)	
Branch	The name of the relevant branch (or tag) you want to work on. Leave empty to work on <i>default</i> branch.	
Username	The username (if any) required to access the repository.	
Authentication	Password	Choose Password if you want to authenticate with a username and password.
	Keyfile with passphrase	Upload an SSH Key and provide the corresponding SSH Passphrase .
	Keyfile without passphrase	Upload an SSH Key .
	Default Mercurial credentials	Bamboo will rely on default hg authentication. Use this option, for example, if you had set up the Bamboo server manually with SSH servers defined in <code>.ssh/config</code> , valid SSH identity files, etc.

Advanced Options

Command timeout	Type the number of minutes bamboo should wait for hg commands to finish. This is useful to stop hung Mercurial processes. On slower networks you may consider increasing default timeout to allow Bamboo to make an initial clone of the Mercurial repository.
Verbose logs	Turns on <code>--verbose</code> and <code>--debug</code> options in hg commands and passes the output to build logs. Use that option if you encounter problems with Mercurial in Bamboo.
Location of POM file	Type the path to your project's pom.xml file which is relative to the root of your Mercurial Repository URL (defined above). <i>(Only available when importing a Maven 2 project)</i>
Disable repository caching	Select this option to enable subrepositories support.
Enable Quiet Period	Specifies a delay after a single commit is detected before the build is started. This allows multiple commits to be aggregated into a single build.
Include/Exclude Files	Allows you to specify the files that Bamboo should, or should not, use to detect changes. Enter into File Pattern a regular expression to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See sub page for examples.
Exclude Changesets	Enter a regular expression to match the commit messages for changesets that should not start a build.
Web Repository	If your repository can be viewed in a web browser, select the repository type. This allows links to relevant files to be displayed in the 'Code Changes' section of a build result.
	Mercurial Web Repository – select one of the following viewer schemes: • BitBucket Web Repository Scheme (if you use BitBucket) • Default Web Repository Scheme (hgserve) (Mercurial's own default web server)

	Stash – specify the following details for the repository: • Stash URL – the URL of your Stash instance (e.g. 'https://stash.mycompany.com'). • Stash Project Key – the key of the project in Stash (e.g. 'CONF'). • Repository Name – the name of the repository in Stash (e.g. 'conf-dev'). See Integrating Bamboo with Stash for more information.
	FishEye – specify the URL and other details for the repository: • FishEye URL – the URL of your FishEye repository (e.g. 'https://atlaseye.atlassian.com/'). • Repository Name – the name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path. • Repository Path – the path for your FishEye repository (e.g. '/atlassian/bamboo/'). See Integrating Bamboo with FishEye for more information.
	 How do I determine my Repository Path? If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

Upgrading remote agents for Mercurial

The [remote agent installer](#) has been modified for Bamboo 2.7 to handle Mercurial source code repositories. This update only impacts Mercurial plans or jobs that use the "SSH/Keyfile with passphrase" option to access the remote repository:

Screenshot: Choosing the 'Keyfile with passphrase' option for a plan or job

Repository URL:	* ssh://lucyfer:8888/hg/Reporting/SR	The URL of Mercurial repository.
Branch:	default	The name of the branch containing source code.
Username:		Username you want to use to access the repository.
Authentication:	Keyfile with passphrase  Password Keyfile with passphrase Keyfile without passphrase Default Mercurial credentials	supports passphrase.
SSH Passphrase:		

If you need to access a Mercurial repository using the SSH protocol with a passphrase-protected keyfile, then you need to upgrade your remote agents to version 2.7 or later. Otherwise, you can keep your old agent. To upgrade your remote agents:

1. Obtain the Bamboo remote agent for version 2.7 or later (i.e. `atlassian-bamboo-agent-installer-x.x.jar` where 'x.x' is 2.7 or later). Refer to [Bamboo remote agent installation guide](#) for more information.
2. Use this file to replace your existing `atlassian-bamboo-agent-installer.x.x.jar` (where 'x.x' is 2.6 or earlier) on the computers running your Bamboo remote agents.
3. Restart the remote agent (i.e. kill it along with accompanying wrapper processes and then issue the command `java -jar atlassian-bamboo-agent-installer-2.7.jar yourBambooAgentServer`).

This procedure should prepare your agent to build Mercurial plans using passphrase-protected SSH keyfiles.

Perforce

The instructions on this page describe how to configure Bamboo to use a Perforce source repository.

Configuring a Perforce source repository

To add a new Perforce repository, navigate to the source repository settings for a plan or job, as described on [Specifying the source repository](#).

1. Either click **Add Repository** to add a new repository, or edit an existing repository configuration.
2. Choose **Perforce** from the **Source Repository** list.
3. Enter a **Display Name** to help identify the repository in Bamboo.
4. You can configure the following settings for a Perforce source repository for your plan:

On this page:

- [Configuring a Perforce source repository](#)
- [Notes](#)

Related pages:

- [Specifying the source repository](#)

Port	Type either the port to which the Perforce client will connect, or the Perforce server itself. This is the Perforce P4PORT environment variable that tells Bamboo which p4d (Perforce server) to use.
Client (Workspace) ⁽³⁾	The name of the Perforce Client Workspace which Bamboo will use. The Client Workspace determines which portions of the depot are visible in your Workspace Tree. Do not create two Plans/Jobs that use the same client (e.g. one client set to manage, the other client set to not manage). This setup will create major issues in your builds.
Depot View	The client view of the depot that contains the source code files for this Plan/Job. This is typically in the form <code>//<client name>/<workspace_mapping>/...</code> For details please see the <i>Perforce User's Guide</i> . Bamboo sets the client root to its working directory , which means that code will be checked out to the 'working directory/<workspace_mapping>' location. Please take note of this, when specifying the 'Artifact Copy Pattern' for your Build Artifacts .)
Username	The Perforce username that Bamboo will use when it accesses the server ('Port'). Leave this field blank if you want Bamboo to use the default Perforce user (i.e. the OS username).
Password	Type the password required by the Perforce username (if applicable).
Let Bamboo manage your workspace ⁽⁴⁾	This field indicates whether or not you want Bamboo to manage your workspace.
Use Client Mapping For Change Detection	Select this option if you use overlay mappings for your workspace. Your workspace must be available on the Bamboo Server

Advanced Options

Enable Quiet Period	Specifies a delay after a single commit is detected before the build is started. This allows multiple commits to be aggregated into a single build.
Include/Exclude Files	Allows you to specify the files that Bamboo should, or should not, use to detect changes. Enter into File Pattern a regular expression to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See sub page for examples.
Exclude Changesets	Enter a regular expression to match the commit messages for changesets that should not start a build.
Web Repository	If your repository can be viewed in a web browser, select the repository type. This allows links to relevant files to be displayed in the 'Code Changes' section of a build result.

	Generic Web Repository • Web Repository URL – the URL of the repository. • Web Repository Module — the particular repository required for this plan or job, if the Web Repository URL above points to multiple repositories. Stash – specify the following details for the repository: • Stash URL – the URL of your Stash instance (e.g. 'https://stash.mycompany.com'). • Stash Project Key – the key of the project in Stash (e.g. 'CONF'). • Repository Name – the name of the repository in Stash (e.g. 'conf-dev'). See Integrating Bamboo with Stash for more information. FishEye – specify the URL and other details for the repository: • FishEye URL — the URL of your FishEye repository (e.g. 'https://atlaseye.atlassian.com/'). • Repository Name — the name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path. • Repository Path — the path for your FishEye repository (e.g. '/atlassian/bamboo/'). See Integrating Bamboo with FishEye for more information.  How do I determine my Repository Path? If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk)). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

Notes

1. **You will not be able to create plans or jobs that use a Perforce repository without specifying the shared local Perforce capability first.** Read more about [configuring a VCS capability](#).
2. **Keep your Perforce configuration up to date** — If you are using Perforce as your repository, you must ensure your Perforce configuration in Bamboo is in sync with any changes to your Perforce repository (such as client, depot or user credential changes). If not, your Perforce repository changes may cause unexpected behaviour in Bamboo when Bamboo tries to access the repository. See the notes in the configuration instructions below for further details.
3. **Issue when running Bamboo with Perforce prior to Bamboo 2.0.7** — A known issue exists when running Bamboo with Perforce prior to Bamboo 2.0.7 (See [BAM-2866](#) and [BAM-2849](#)). If you change the name of your Perforce client (i.e. via an update) without updating your Perforce configuration in Bamboo, Bamboo will not be able to find the Perforce client to run against. Perforce will then create a default client in your running directory. This can lead to situations where Bamboo will attempt to clear out data from your running directory (e.g. force build). To avoid this problem, ensure that you update the '**Client**' in your Perforce configuration whenever you change your Perforce client.
4. Please be aware of the following implications when either letting Bamboo manage or preventing Bamboo from managing your workspace:
 - **If you let Bamboo manage your workspace,**
 - We recommend this configuration if your Jobs will be running on many different machines or different operating systems, as Bamboo sets the client root for you.
 - Bamboo will make configuration changes to the Client Workspace to manage builds (e.g. Bamboo will modify the host and root). You need to ensure that you enter a Client Workspace in the '**Client**' field that will be used **solely for Bamboo**.
 - Under this configuration, you should configure one client per Job to avoid conflicts when updating the client root.
 - **If you do not let Bamboo manage your workspace,**
 - We recommend this configuration if you wish to reuse your client for several Jobs, as Bamboo will retrieve the client root directory from Perforce and use it to run builds.
 -  **Setting the client root in Perforce:** We strongly recommend that you choose a directory that is dedicated for Bamboo's use only, when you are specifying the client root in your Perforce repository. This directory may get cleaned (i.e. files and sub-directories deleted) if you choose to force clean builds.
 - Under this configuration, you need to ensure that the client root directory exists on all machines that the Job will be

- built on.
- Please note that alternate roots does not currently work in Bamboo. See issue [BAM-2377](#) for further details.

Using Perforce with Bamboo - limitations and workarounds

There are some limitations to using Bamboo with Perforce. Please read the following information carefully before setting up a build plan to use Perforce.

On this page:

1. Running builds on multiple remote agents or machines
2. Using Perforce Overlay and Exclusionary Mappings in Bamboo

1. Running builds on multiple remote agents or machines

Limitation

You will not be able to run builds on **multiple remote agents and/or multiple remote machines** using a **Perforce repository**, without using one of the workarounds described below. If you try to do so, you will run into problems with change detection that could **cause your agents to build incorrect code**. This problem **does not** affect the running of builds on multiple local agents.

Background

Perforce is a client/server SCM (software configuration management) system that manages your changes/files by storing the change information on its server. However, storing change information on the Perforce server can cause problems when you have clients on multiple agents/machines. If you have downloaded a particular change with a Perforce client, the change will be marked as downloaded by the Perforce server. If you use the same Perforce client on another machine, the Perforce server will incorrectly assume that you have already downloaded that particular change and will not download it. Hence, your agents may not pick up changes correctly and could build incorrect code.

Workarounds

There are a few workarounds available for this issue, if you are using Perforce with Bamboo:

- **Restrict your plan to use a single machine** — you can use one or more remote agents to build a plan, if they are *running on the same machine* and you *set the client root yourself* (i.e. do not let Bamboo [manage your workspace](#)) so that your agents will build to the same directory.
- **Make Bamboo force a clean build every time it builds** — this will ensure that your agents are always building the correct code. However, it can be an inefficient setup for big projects.
- **Use alternate roots for different machines** — specifying alternate roots for different machines will allow you to work around the change detection issue, as long as the roots on each machine are unique. Please note however, you will be *restricted to three machines* (with three different roots) due to Perforce limitations.

Please see the following JIRA issues for further information, [BAM-2843](#) and [BAM-2774](#).

2. Using Perforce Overlay and Exclusionary Mappings in Bamboo

Limitation

You will not be able to control how Bamboo *detects changes* using [exclusionary mappings](#) or [overlay mappings](#).

i Please note, this issue does not affect you if you only trigger your builds [on a schedule](#) or [manually](#), as Bamboo agents still build the correct code when triggered.

i You may want to try the "Use Client Mapping For Change Detection" available in the [Bamboo Perforce repository](#) type.

Background

Bamboo currently uses the depot view, not the client view, when detecting changes. Hence, any exclusionary and overlay mappings will not be available during change detection.

For example, if a p4 client uses an overlay mapping like this one:

```
//depot/Prj/... //clientName/depot/Prj/...  
+//depot/Dep/... //clientName/depot/Prj/Dep/...
```

and the 'Depot' specified in a [plan's repository configuration](#) is:

```
//clientName/depot/Prj/...
```

then Bamboo will lookup the corresponding depot view and detect changes by running the following command:

```
p4 changes //depot/Prj/...
```

Consequently, **no changes** to files in `//clientName/depot/Prj/Dep/...` will be picked up by change detection, despite the overlay mapping.

Hence, if you set up your build to **trigger when code is updated** it will not trigger correctly.

Workarounds

A partial workaround is available in Bamboo, if you wish to use exclusionary mappings for your client workspace. Specify your build plan to exclude files that match a specified pattern by choosing 'Exclude all changes that match the following pattern' from the '**Include / Exclude Files**' dropdown (under the 'Common repository configuration' section). See [this document](#) for further details. Please note, this will only **exclude one pattern** whereas multiple exclusions can be specified in an exclusionary mapping.

Unfortunately, there is no workaround for overlay mappings in Bamboo.

Please note, we are aware of these problems and are working to address them — see the following JIRA issue for further information, [BAM-3323](#).

Subversion

The instructions on this page describe how to configure Bamboo to use a Subversion source repository.

On this page:

- [Configuring a Subversion source repository](#)
- [Notes](#)

Related pages:

- [Specifying the source repository](#)
- [Setting the SVN workspace format](#)

Configuring a Subversion source repository

Navigate to the source repository settings for a plan or job, as described on [Specifying the source repository](#).

1. Either click **Add Repository** to add a new repository, or edit an existing repository configuration.
2. Choose **Subversion** from the **Source Repository** list.
3. Enter a **Display Name** to help identify the repository in Bamboo.
4. You can configure the following settings for a Subversion source repository for your plan:

Repository URL	The location of your Subversion repository e.g. <code>http://svn.collab.net/repos/svn/trunk</code> ⓘ Note that you can use global variables in this field (see Using Global or Build-specific Variables). ⓘ If you are importing a Maven 2 Project, this location should contain your project's <code>pom.xml</code> file.
Username	(Optional) The Subversion username (if any) required to access the repository.
Authentication Type	Password – choose this option if you want to authenticate with a username and password. SSH – if you choose to authenticate using SSH, you need to provide the following details: • Private Key — the absolute path of your SSH private key. • Passphrase — the passphrase for your SSH private key. ⓘ If you are planning to use remote agents the ssh private key file has to be copied to the agent box into the same location as specified. SSL Client Certificate – if you choose to authenticate using an SSL Client Certificate, you need to provide the following details: • Private Key — the absolute path of your SSL client certificate . • Passphrase — the passphrase for your SSL client certificate. ⓘ Please note, the client certificate has to be in PKCS12 format and the client certificate file must be passphrase protected, otherwise a runtime exception is thrown by the JDK security engine while opening the user key.

Advanced Options

Detect Changes in Externals	Select this if your Subversion repository uses svn:externals to link to other repositories (your externals must be in the root of the checkout directory, not in a subdirectory). Please note that you only need to select this check box if you require Bamboo to detect changes in the externals. If your externals reference a particular (static) revision, you do not need to check this box.
Use SVN Export	This option will speed up the first-time checkout, but updates are not supported. Implies Force Clean Build.
Enable Commit Isolation	Ensures that a build will only have one change, allowing you to isolate your build failures.
Automatically detect root URL for branches	Specifies whether the VCS Branching Task automatically determines the location of created branches.
Automatically detect root URL for tags	Specifies whether the VCS Tagging Task automatically determines the location of created branches.
Enable Quiet Period	Specifies a delay after a single commit is detected before the build is started. This allows multiple commits to be aggregated into a single build. <i>(Only available when configuring an existing plan.)</i>
Include/Exclude Files	Allows you to specify the files that Bamboo should, or should not, use to detect changes. Enter into File Pattern a regular expression to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See sub page for examples.
Exclude Changesets	Enter a regular expression to match the commit messages for changesets that should not start a build.
Web Repository	If your repository can be viewed in a web browser, select the repository type. This allows links to relevant files to be displayed in the 'Code Changes' section of a build result.
	Generic Web Repository • Web Repository URL – the URL of the repository. • Web Repository Module — the particular repository required for this plan or job, if the Web Repository URL above points to multiple repositories.
	Stash – specify the following details for the repository: • Stash URL – the URL of your Stash instance (e.g. 'https://stash.mycompany.com'). • Stash Project Key – the key of the project in Stash (e.g. 'CONF'). • Repository Name – the name of the repository in Stash (e.g. 'conf-dev'). See Integrating Bamboo with Stash for more information.

FishEye – specify the URL and other details for the repository:

- **FishEye URL** — the URL of your FishEye repository (e.g. 'https://atlassian.atlassian.com/').
- **Repository Name** — the name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path.
- **Repository Path** — the path for your FishEye repository (e.g. '/atlassian/bamboo/').

See [Integrating Bamboo with FishEye](#) for more information.



How do I determine my Repository Path?

If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to [view the code changes](#) and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when [browsing a repository](#) (e.g. trunk)). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

Notes

- If you are having problems **connecting to Subversion**, consult our documentation on [troubleshooting Subversion connections](#).
- If you use **pre-1.5 Subversion client** to access code checked out by Bamboo, you may encounter problems with your builds. This is due to the SVNKit upgrade in Bamboo 2.1.4. Please read this [knowledge base article](#) for further details.
- You can add the `-Dsvnkit.http.methods=Basic,NTLM` system property to SVNKit to have NTLM authentication work with Bamboo.

Configuring source code management triggers for Subversion

This page provides instructions on how to configure Subversion to send message events that trigger the execution of Bamboo plans. You only need to configure Subversion to send these message events if **The repository triggers the build when changes are committed** [build strategy](#) has been specified for one or more of your Bamboo plans.

Configuring Subversion to trigger a build

This section explains how to configure Subversion to trigger a build when the repository is changed. A Subversion hook script is used to perform the trigger action whenever a Subversion repository is changed.

The following commands and script files assume that your Subversion server runs on a UNIX- or Linux-based operating system. If your Subversion server runs on any other operating system, then you will need to modify the script files and if necessary, the commands below to suit that operating system.

On this page:

- [Configuring Subversion to trigger a build](#)
- [Notes](#)

Related pages:

- [Subversion](#)
- [Configuring source code management triggers for CVS](#)

Step 1. Enable the Subversion post-commit hook

To do this, run the following commands:

```
cd svn-repository-containing-the-build-source-code
cd into the hooks/ directory
```

The Subversion post-commit file is not installed by default. If it does not exist, make a copy of the `post-commit.tmp1` file in the `hooks/` directory, name it `post-commit` and make it executable:

```
cp post-commit.tmpl post-commit
chmod a+rx post-commit
```

Step 2. Install the post-commit trigger

Add a line like the following to the post - commit file, for running Bamboo's build trigger script file.

```
/path-to-your-bamboo-installation/scripts/svn-triggers/postCommitBuildTr
igger.sh base-url BUILD-KEY
```

where:

- base-url is the base URL of the Bamboo server. For example:
`http://<name-of-machine>:8085`
- BUILD-KEY is the key of the Bamboo plan to be executed.

Make Bamboo's build trigger script file executable (using `chmod`) so that the Subversion user can execute it.

Step 3. Do a test commit

Conduct a 'test' commit. Bamboo should start building the relevant plan after a few seconds.

The Bamboo log file should contain an entry like this:

```
[INFO] com.atlassian.bamboo.build.UpdateAndBuild - Bamboo build was
triggered by remote http call from 127.0.0.1
```

Notes

Build Trigger Security — Bamboo will only accept remote build triggers if the triggers originated from the Subversion server(s) identified in the Subversion **Repository URL** of any Bamboo plans. Requests originating from other Subversion servers will be rejected by Bamboo.

Configuring a shared source repository

You can set up shared source repositories that are then available globally to all plans and jobs configured on the Bamboo server. Doing this can save you from having to reconfigure the source repositories in multiple places if these ever change. Changes to a shared repository are applied to every plan or job that uses the repository.

To configure shared source repositories:

1. Click **Administration** and then **Shared Repositories** (under 'Build Resources').
2. Either add a new repository, or edit an existing shared repository.
3. Follow the on-screen instructions to configure the repository. Bamboo is able to connect to a variety of SCMs; for details regarding a particular repository type, please refer to the pages listed below:
 - [Bitbucket - Mercurial](#)
 - [CVS](#)
 - [Git](#)
 - [GitHub](#)
 - [Mercurial](#)
 - [Perforce](#)
 - [Subversion](#)

Triggering builds

Triggering in Bamboo allows plan builds to be started automatically. Bamboo has the following trigger methods:

Trigger a build when code is updated:

- [Polling the repository for changes](#)
- [Repository triggers the build when changes are committed](#)

Trigger a build based on a schedule:

- [Cron-based scheduling](#)
- [Single daily build](#)

Trigger a build depending on the builds of other plans:

- Plan builds are triggered by [preceding successful builds](#) of other plans.

- Plan only builds if [other specified plans are building successfully](#).

On this page:

- [Choosing a triggering strategy](#)
- [Conditional build triggers](#)

Related pages:

- [Running a plan build manually](#)
- [Setting up plan build dependencies](#)

Note that a plan that has no configured triggers can only be started [manually](#), or if it is [dependent](#) on the successful build of another plan.

From Bamboo 4.3, you can configure multiple triggers for each plan. This allows a plan to be triggered by different trigger types, and to have triggering scenarios such as "every 5 minutes between 9:00am and 10:00am, and every 20 minutes between 1:00pm and 10:00pm".

Triggers can only be configured by a Bamboo administrator.

Choosing a triggering strategy

This table lists the ways in which plan builds can be triggered in Bamboo.

Triggering option	Description
Polling the repository for changes	Bamboo will 'poll' the selected source code repositories for code changes, using either a specified interval (that is, periodically) or a schedule. If Bamboo detects code changes, a build of the plan is triggered. • Your VCS must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in a repository. See Polling the repository for changes.
Repository triggers the build when changes are committed	Bamboo waits to receive a message about changed code from any of the selected source code repositories. When Bamboo receives such a message, a build of the plan is triggered. • This option minimises server load, because message events are sent only when code changes to a repository are committed. • You must configure your source code management system to send message events to Bamboo about code changes in the repositories. See Repository triggers the build when changes are committed.
Cron-based scheduling	Bamboo will trigger scheduled builds of this plan based on a cron expression. • This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. • Scheduled builds are triggered irrespective of any code changes in the source code repositories. See Cron-based scheduling.
Single daily build	Bamboo will trigger a build of the plan once per day at a specified time. • Can be set up to run at a time of your choice. • This option is suitable if a build of this plan takes a long time to complete. • Scheduled builds are triggered irrespective of any code changes in the source code repositories. See Single daily build.

Conditional build triggers

This field is only available when configuring an existing plan.

You can choose to run builds of this plan only if other specified plans are currently passing. Those plans must build successfully before this plan will be built. See also [Setting up plan build dependencies](#).

Note that, because a plan may have multiple triggers configured, each of which can have differing trigger conditions, it is possible for the plan build to be started by one trigger, even though another trigger is currently blocked.

To specify build trigger conditions:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list and click the edit icon () to display the plan's configuration pages.
3. Click the **Triggers** tab, and click either an existing trigger or **Add Trigger**.
4. Select **Only run Build if other Plans are currently passing**, under 'Trigger Conditions'.
5. Specify one or more other plans by adding their full keys.

Trigger Conditions

☒ Only run Build if other Plans are currently passing

Plan Keys

FULL_KEY

Enter comma seperated Plan Keys.

Polling the repository for changes

You can configure Bamboo to poll the repository for source code changes, either:

- periodically (e.g. every 180 seconds), or
- based on a schedule (e.g. the second Sunday of every month at 5:00 am).

If Bamboo detects a change in the source code, a build of your plan is triggered.

Related pages:

- [Triggering builds](#)
- [Repository triggers the build when changes are committed](#)
- [Cron-based scheduling](#)
- [Single daily build](#)

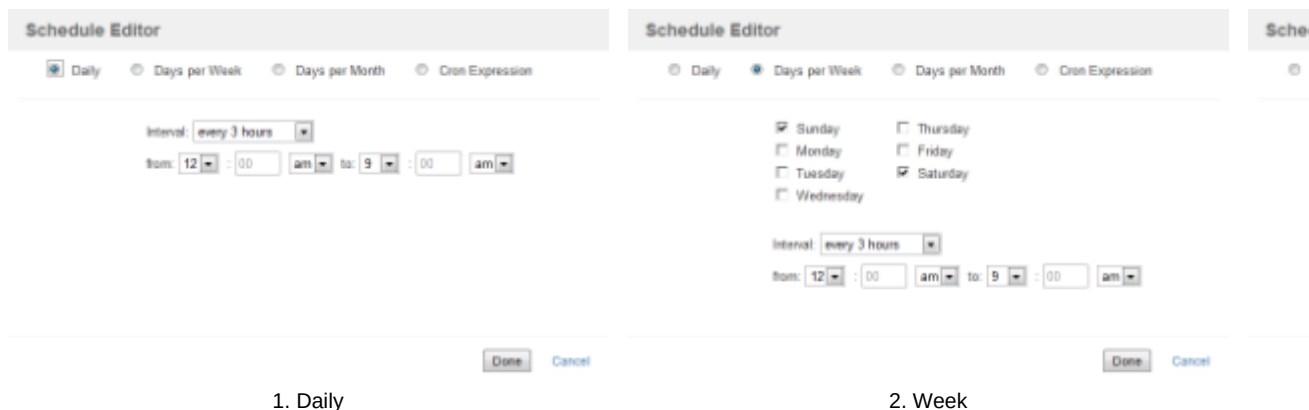
To configure Bamboo to poll the repository for source code changes:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list and click the edit icon () to display the plan's configuration pages.
3. Click the **Triggers** tab, then click either an existing trigger or **Add Trigger**.
4. Optionally, enter a trigger description.
5. Choose **Trigger type** > **Polling the Repository for changes**.
6. Bamboo displays the available repositories for the plan, as previously configured on the **Source Repositories** tab. Choose the repositories that this trigger should apply to.
7. Choose a polling strategy:

Periodically	Enter a Polling Frequency value (in seconds) for the time between when Bamboo checks for repository changes.
Scheduled	Click the edit icon () to use the Schedule Editor to set the polling schedule. Note, this is a schedule for polling your repository: a plan build will only be triggered if there are source code changes. See Triggering builds. Note that for the Cron Expression option, a cron expression consists of 6 mandatory and one optional field. The fields in sequential order are: seconds, minutes, hours, day-of-month, month, day-of-week and (optional) year. For example, <code>0 0 1 ? * 1#2</code>. For information on Cron expressions, see this FAQ: How do I construct a cron expression in Bamboo

8. Click **Save Trigger**.

Screenshots: *Scheduling polling for changes*



Repository triggers the build when changes are committed

Using the source repository to trigger the build of a plan is one of the available methods for [triggering builds](#) in Bamboo.

"Repository triggers the build when changes are committed" has the advantage of placing minimal load on your Bamboo server. However, it requires that your source repository is configured to fire an event to the Bamboo server (which the configured Plan will 'listen for').

Configuring the repository to trigger the build when changes are committed requires two changes:

1. [Configuring your source repository](#)
2. [Configuring Bamboo to respond to post-commit messages](#)

The overall process is: a commit to the repository causes a post-commit message to be sent to Bamboo. Bamboo responds by checking the repository for unbuilt changes. If changes are found, Bamboo triggers a build.

Related pages:

- [Triggering builds](#)
- [Polling the repository for changes](#)
- [Cron-based scheduling](#)
- [Single daily build](#)

1. Configuring your source repository

Configure your source code management system's repository to send post-commit event messages to Bamboo. These messages tell Bamboo to begin building the plans that use this repository.

▼ [For CVS, click here to expand...](#)

Edit two files in the CVSROOT module: `commitinfo` and `loginfo`.

- For `commitinfo`, add a line like this:

```
^jira(/|$) /pathto/preCommit.sh
```

where "jira" is your module.

- For `loginfo`, add a line like this:

```
^jira(/|$) /pathto/postCommitBuildTrigger.sh %{}
http://bambooserver JIRA-MAIN JIRA-BRANCH
```

where JIRA-MAIN and JIRA-BRANCH are the Bamboo plans that you would like to trigger, JIRA being the project key and BRANCH or MAIN being the plan key.

Please refer to [Configuring source code management triggers for Subversion](#).

▼ [For Subversion, click here to expand...](#)

If you are using a remote SVN server, copy file "`atlassian-bamboo/repositoryScripts/svn-triggers/postCommitBuildTrigger.sh`" to the SVN repository `.../hook/post-commit` folder so that the `postCommitBuildTrigger.sh` file can be accessible from `post-commit` trigger file.

Edit the Subversion repository's `hooks/post-commit` trigger file with something like:

```
/path/to/postCommitBuildTrigger.sh http://bambooserver JIRA-MAIN  
JIRA-BRANCH
```

Please refer to [Configuring source code management triggers for Subversion](#).

▼ [For Perforce, click here to expand...](#)

Add the script as a change-commit trigger.

```
triggerName change-commit //myDepot/...  
"/usr/local/bin/postCommitBuildTrigger.sh http://bambooServer/  
MYPLAN-DEFAULT"
```

▼ [For Mercurial, click here to expand...](#)

Edit the Hg repository's .hg/hgrc settings file with something like:

```
[hooks]  
changegroup.update = /path/to/postCommitBuildTrigger.sh  
http://bambooserver JIRA-MAIN JIRA-BRANCH
```

▼ [For Git, click here to expand...](#)

Edit the Git repository's .git/hooks/post-receive trigger file with something like:

```
/path/to/postCommitBuildTrigger.sh http://bambooserver JIRA-MAIN  
JIRA-BRANCH
```

1. Copy the scripts to your repository. If you are using the Bamboo distribution, the scripts are located in the /scripts folder of your [Bamboo Installation Directory](#). If you are using Bamboo EAR-WAR distribution, you can find them in the /repositoryScripts folder. You can also download the scripts by following this [link](#).
2. Depending on which operating system your repository is running on, you may need to edit the scripts. The scripts assume that 'wget' is in '/usr/bin/'; if this isn't the case for your repository (e.g. Solaris 10 has it in /usr/sfw/bin/), edit the scripts and change '/usr/bin/' to the appropriate location.
3. Ensure that the user which Bamboo is running as has appropriate file permissions to execute the scripts, i.e. the scripts should be executable by non-root user(s).
4. [Enable Bamboo's remote API](#) so that the scripts can use Bamboo's REST-style remote API to access Bamboo's data.

2. Configuring Bamboo to respond to post-commit messages

Before you begin:

- Triggering a build when there is no repository update — Bamboo will ignore build triggers if the local working copy and the repository copy have the same revision numbers. When testing your build triggers, ensure that the local working copy is not the latest version - if this is the case, Bamboo will take no further action.

To configure Bamboo to trigger a build on code check in:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list and click the edit icon to display the plan's configuration pages.
3. Click the **Triggers** tab, then click either an existing trigger or **Add Trigger**.
4. Optionally, enter a trigger description.
5. Choose **Trigger type** > **Repository triggers the build when changes are committed**.
6. Bamboo displays the available repositories for the plan, as previously configured on the **Source Repositories** tab. Choose the repositories that this trigger should apply to.
7. Only enter an IP address in **Trigger IP Addresses** if you want Bamboo to trigger on post-commit messages from other than the primary IP address for the repository.



If you use a Mercurial or Git repository then you *must* type the IP address of your repository host in **Trigger IP Addresses**.

8. Click **Save Trigger**.

Screenshot: Build Strategy – repository triggers the build when changes are committed

Repository Polling

Repository Polling

Add Trigger

Trigger Configuration

Trigger Description

Repository Polling

Trigger type*

Repository triggers the build when changes are committed

How should Bamboo trigger Builds for this Plan? (Dependent Builds are automatically triggered)

Repositories

☒ Bitbucket
 ☒ SVN Shared Repo test

Which repositories should the Trigger apply to?

Trigger IP Addresses

(Optional) Bamboo ensures that triggers originate from IP addresses of the repository server(s). You can authorise additional IP addresses here, separated by a comma.

Trigger Conditions

☐ Only run Build if other Plans are currently passing

Save Trigger

Cancel

Cron-based scheduling

Using a cron-based schedule to trigger the build of a plan is one of the available methods for [triggering builds](#) in Bamboo. This schedule is configured using the Schedule Editor.

The schedule can be daily (times per day), weekly (days per week), monthly (days per month) or based on a cron expression.

Related pages:

- [Triggering builds](#)
- [Polling the repository for changes](#)
- [Repository triggers the build when changes are committed](#)
- [Single daily build](#)

To schedule a plan build using a cron expression:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list and click the edit icon (✎) to display the plan's configuration pages.
3. Click the **Triggers** tab, then click either an existing trigger or **Add Trigger**.
4. Optionally, enter a trigger description.
5. Choose **Trigger type** > **Cron Based Scheduling**.
6. Click the edit icon (✎) next to the current schedule to display the Schedule Editor.
7. Use the Schedule Editor (see screenshots below), to specify the build schedule for your plan. For information about cron expressions, see this FAQ: [How do I construct a cron expression in Bamboo](#).
8. Click **Save Trigger**.

Screenshots: Schedule Editor options

☒ Daily Interval: once per day At: 12 : 00	☒ Days per Week ☐ Sunday ☐ Tuesday ☐ Monday ☐ Friday ☐ Tuesday ☐ Saturday ☐ Wednesday Interval: once per day At: 12 : 00 am	☒ Days per Month At: 12 : 00 a ☒ The 1st day of every month ☐ The first Sunday of every month	☒ Cron Expression 0 0 0 ? * *

Single daily build

Triggering the build of a plan to run at a particular time each day is one of the available methods for [triggering builds](#) in Bamboo. A "Single daily build" runs at a time of your choice. This is particularly suitable for builds that take a long time to complete.

Related pages:

- [Triggering builds](#)
- [Polling the repository for changes](#)
- [Repository triggers the build when changes are committed](#)
- [Cron-based scheduling](#)

To schedule a plan build at a specified time each day:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list and click the edit icon to display the plan's configuration pages.
3. Click the **Triggers** tab, then click either an existing trigger or **Add Trigger**.
4. Optionally, enter a trigger description.
5. Choose **Trigger type** > **Single daily build**.
6. Specify the time of day at which the build should run in **Build Time**. Use hh:mm format, with a 24-hour clock.
7. Click **Save Trigger**.

Running a plan build manually

Typically in Bamboo, your build plans are configured to be [automatically triggered](#) when code changes are committed to the working repository, or according to a schedule.

However, there can be scenarios where you do not want the plan to be automatically triggered:

- The plan should only ever be run manually.
- You want to choose the revision of the default repository that should be used for the build.
- You want to run a customised build, so as to override [global variables](#) or [plan variables](#).
- You want to select particular manual stages to run.
- You want the plan to be [triggered by other plans](#) that build successfully first.

This page describes how to run a plan build manually, and the options available when running a customised plan build.

Running a plan build manually

To start a plan build manually:

1. Locate the relevant plan on the [Dashboard](#).
2. Click the Run icon  for the plan.

Alternatively, if you are viewing the plan, simply click the **Run** menu.

On this page:

- [Running a plan build manually](#)
- [Running a customised manual build](#)

Related pages:

- [Triggering builds](#)
- [Setting up plan build dependencies](#)
- [Stopping an active build](#)
- [Defining plan variables](#)

Running a customised manual build

If you trigger a plan build manually, you can customise the following aspects of how the plan is run (when these are available):

- Choose the revision of the default repository that should be used.
- Override any [global variables](#) or [plan variables](#) with your own parameters when triggering a build manually. This is referred to as running a 'parameterised plan build'.
- Select which [manual stages](#) to run, if manual stages have been configured for the plan.

To run a customised plan build:

1. Locate the relevant plan on the [Dashboard](#).
2. Click the plan name to go to the Plan Summary.
3. Choose **Run** > **Run Customised**.
4. Customise the following aspects of the plan:

Revision	Choose a repository revision to use for the build. Note that: • You can only choose revisions from the default repository. • The build is not included in plan statistics or telemetry. • SVN repositories use the revision number • Perforce projects use the changelist number • Git repositories use the changeset number • Mercurial repositories use the tag  Note for Subversion repositories that make use of externals When running a build with a custom revision on a Subversion repository with externals, Bamboo will choose the latest revision in the external repository. This is because Subversion externals always use the latest version and cannot be fixed at a specific revision.
Build Variables	Click Override a variable to choose another variable to override.
Stages	Choose the stages that should be run.

5. Click **Run**.

Run Customised Plan

Revision latest revision

Use a specific revision of the default repository in this build

Build Variables funcMavenOpts

-Xmx512m -Xms64m -XX:ReservedCodeC

Override a variable

Stages

☒ Prepare Sandbox Stage
☐ Test the sources build
☐ Promote Sandbox Stage
☐ Copy Artifacts to WWW

Run

Cancel

Rerunning a failed stage

If a stage has failed in your build, you can choose to rerun the stage (with exactly the same data) instead of the entire plan.

To rerun a stage:

1. Navigate to the failed build result, as described on [Viewing a build result](#).
2. Choose **Run >**

Rerun all failed Jobs to run the stage again.

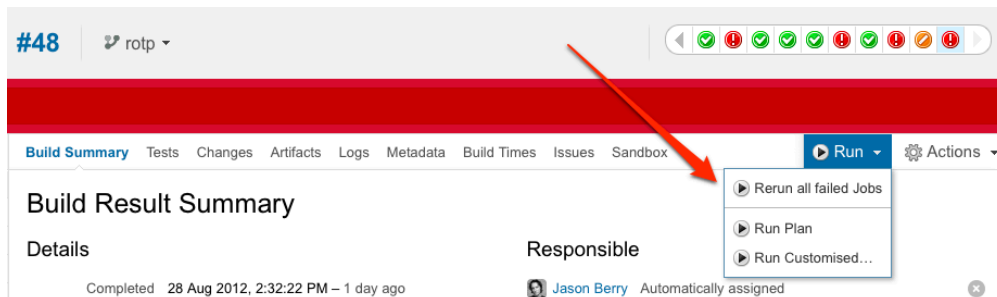
Note that:

- Only failing jobs will be re-run.
- Subsequent stages will be executed automatically, unless they are [manual stages](#).
- You might want to add a comment to the build result to record the reason for failure. The existing build result will be overwritten (Bamboo will not create a new build) and the previous failure reason will not be retained.
- For plans based on a Subversion repository, you can only rerun the failed job or the whole plan.

Related pages:

- [Running a plan build manually](#)
- [Editing a plan's configuration](#)
- [Configuring jobs](#)
- [Using stages in a plan](#)

Screenshot: Rerunning a failed stage



Using stages in a plan

Stages group (or 'map') *jobs* to individual steps within a *plan's build process*. For example, you may have an overall plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo stages to represent each of these steps.

A stage:

- Has a single *job*, by default, but can be used to group multiple jobs.
- Processes its jobs in **parallel**, on **multiple agents** (where available).
- Must successfully complete all its jobs before the next *stage* in the *plan* can be processed.
- May produce *artifacts* that can be made available for use by a subsequent stage.

Each new plan created in Bamboo contains at least one stage (for the *default job*) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

On this page:

- [About manual stages](#)
- [Navigating to the stages for a plan](#)

Related pages:

- [Creating a stage](#)
- [Editing a stage](#)
- [Deleting a stage](#)

About manual stages

Any stage in a plan can be configured to be a manual stage. If you run a plan with manual stages, Bamboo will pause the execution of the plan every time it reaches a manual stage. The plan build will only continue once a user has manually triggered the stage. Please note:

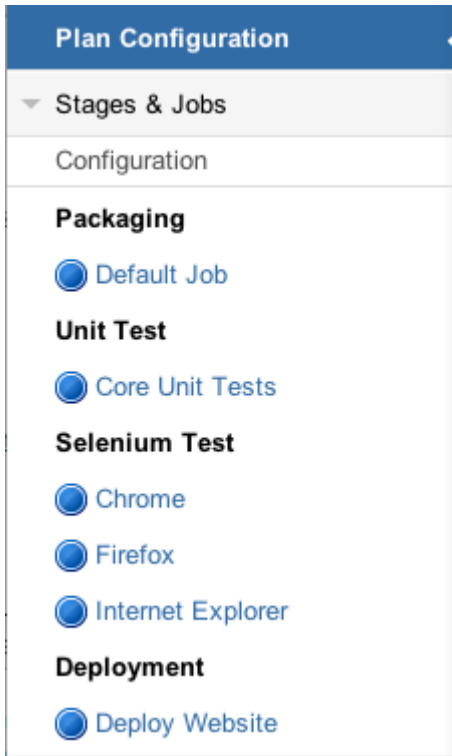
- A manual stage can only be triggered if the previous stage has completed successfully.
- Manual stages must be executed in the order that they are configured in the plan. You cannot skip a manual stage.
- Manual stages will be displayed in the Plan Navigator with either this icon  (not due to be triggered) or this icon  (pending execution).
- You need 'Build' [permission](#) on the plan to run a manual stage.

Navigating to the stages for a plan

To navigate to the stages for a plan:

1. Click **Dashboard** and then the **All Plans** tab.
2. In the list of plans, click the name of the desired plan. The plan's 'Plan Summary' page will be displayed.
3. Choose **Actions** > **Configure Plan**.
4. Expand the **Stages & Jobs** section of the left navigation panel to see the stages of the plan.

Screenshot: The stages for a plan in the left navigation panel



Creating a stage

This page describes how to create a stage in a plan.

Stages group (or 'map') [jobs](#) to individual steps within a [plan's build process](#). For example, you may have an overall plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo stages to represent each of these steps.

A stage:

- Has a single [job](#), by default, but can be used to group multiple jobs.
- Processes its jobs in **parallel**, on **multiple agents** (where available).
- Must successfully complete all its jobs before the next [stage](#) in the [plan](#) can be processed.
- May produce [artifacts](#) that can be made available for use by a subsequent stage.

Each new plan created in Bamboo contains at least one stage (for the [default job](#)) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

Related pages:

- [Configuring a stage \(About manual stages\)](#)
- [Editing a stage](#)
- [Deleting a stage](#)
- [Creating a job](#)
- [Configuring jobs](#)
- [Configuring a job's build artifacts](#)

To create a stage within a plan:

1. Click **Dashboard**, then the **All Plans** tab, then the name of the plan in the list, to get to the plan you want to edit.
2. Choose **Actions > Configure Plan**.
3. Click the **Stages** tab, and then **Create Stage** on the right. The 'Create a new Stage' dialog will appear.
4. Complete the form and click **Create**. For information about manual stages, see [Configuring a stage \(About manual stages\)](#).
5. (optional) You may want to do one or more of the following with your new stage:
 - Order your new stage in the list of stages, by dragging and dropping it.
 - Create a new job to your new stage.
 - Move a job from another stage to your new stage by dragging and dropping the job.

Note that you may break [artifact dependencies](#) by moving stages, or by moving jobs between stages. Bamboo will warn you, if a dependency will be broken by moving a stage or a job.

Screenshot: Creating a stage

Create a new Stage

Stage Name*

Deployment

How do you want to identify the new Plan Stage?

Description

Deployment tasks

Choose a meaningful description for this Plan Stage

☒ Manual

Manual stages require user interaction in order to execute

Create

Cancel

Editing a stage

To edit a stage for a plan:

1. Click **Dashboard**, then the **All Plans** tab, then the name of the plan in the list, to get to the plan you want to edit.
2. Choose **Actions > Configure Plan**.
3. Click the **Stages** tab
4. Edit the stage as required:
 - To edit the name and description of the stage or configure whether it is a [manual stage](#), click the cog (⚙️) icon and choose **Configure Stage**.
 - To move the stage, drag and drop the stage to the desired place in the plan.

Note that you may break [artifact dependencies](#) by moving stages. Bamboo will warn you, if a dependency will be broken by moving a stage.

Related pages:

- [Configuring a stage \(About manual stages\)](#)
- [Creating a stage](#)
- [Deleting a stage](#)
- [Configuring a job's build artifact](#)

Screenshot: Stages for a plan

Stages

[Create Stage](#)

Each Stage within a Plan represents a step within your build process. A Stage may contain one or more Jobs which Bamboo can execute in parallel. For example, you might have a Stage for compilation Jobs, followed by one or more Stages for various testing Jobs, followed by a Stage for deployment Jobs.

⚙️

Packaging

⚙️

⚙️

Default Job

Disable | Delete

+

Create Job

↓

⚙️

Unit Test

⚙️

⚙️

Core Unit Tests

Disable | Delete

+

Create Job

↓

⚙️

Selenium Test

⚙️

⚙️

Chrome

Disable | Delete

⚙️

Firefox

Disable | Delete

⚙️

Internet Explorer

Disable | Delete

+

Create Job

↓

⚙️

Deployment

⚙️

⚙️

Deploy Website

Disable | Delete

+

Create Job

Deleting a stage

Before you begin, please see the [Notes](#) on this page.

To delete a stage from a plan:

1. Click **Dashboard**, then the **All Plans** tab, then the name of the plan in the list, to get to the plan you want to edit.
2. Choose **Actions > Configure Plan**.
3. Click the **Stages** tab.
4. Click the cog (⚙️) icon for the relevant stage and choose **Delete Stage**.
5. Click **Confirm** to delete the stage. Note that a deleted stage cannot be recovered.

Related pages:

- [Creating a stage](#)
- [Editing a stage](#)
- [Configuring a job's build artifact](#)

Screenshot: *Confirming deletion of a stage*

Delete Functional Tests

 You are about to delete Stage "Functional Tests". This Stage has 0 Job. If you proceed with this action Bamboo will irrevocably delete all Job configurations, artifacts, logs and results related to this Stage.

Confirm

Cancel

Notes

- Deleting a stage will delete all job configurations, artifacts, logs and results related to the stage. These cannot be recovered after the stage is deleted.
- You may break [artifact dependencies](#) by deleting a stage.

Jobs and tasks

The following pages contain information about configuring jobs and tasks for your Bamboo plans. If you are looking for information about Bamboo builds, please see [Working with builds](#).

- [Creating a job](#)
- [Configuring jobs](#)
- [Disabling or deleting a job](#)
- [Configuring tasks](#)

Jobs

A Bamboo *job* is a single build unit within a [plan](#). One or more jobs can be organised into one or more [stages](#). The jobs in a stage can all be run at the same time, if enough Bamboo agents are available. A job is made up of one or more [tasks](#).

A job:

- Processes a series of one or more [tasks](#) that are run **sequentially** on the **same agent**.
- Controls the order in which tasks are performed.
- Collects the [requirements](#) of individual tasks in the [job](#), so that these requirements can be matched with agent [capabilities](#).
- Defines the [artifacts](#) that the build will produce.
- Can only use artifacts produced in a previous stage.
- Specifies any [labels](#) with which the [build result](#) or build artifacts will be tagged.

Each new plan created in Bamboo contains at least one job known as the 'Default Job'.

Projects and plans can only be configured by Bamboo administrators (see [Creating a plan](#)).

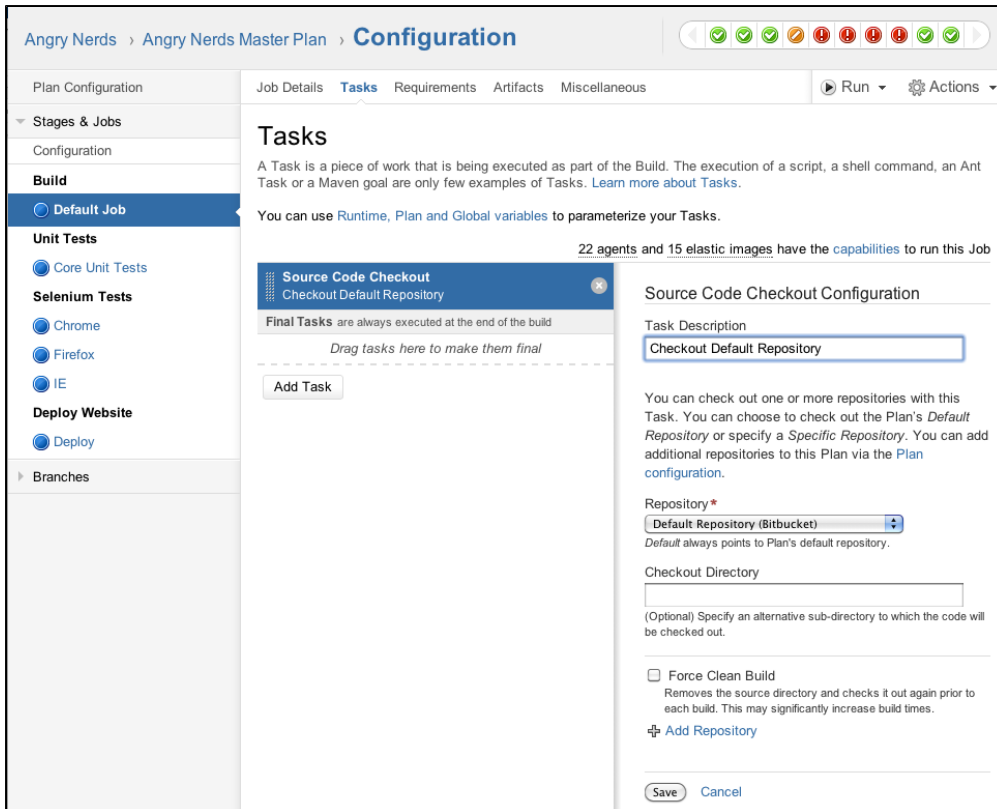
Tasks

A task:

- Is a small discrete unit of work, such as source code checkout, executing a Maven goal, running a script, or parsing test results.
- Is run sequentially within a [job](#) on a Bamboo working directory.

Tasks may make use of an [executable](#) if required. Once a task is defined in the Bamboo system, it can then be specified in jobs by a plan administrator. A job can be configured to execute a number of tasks, on the same working directory. For example, before executing a Maven goal, the user could substitute specific files within the working directory, substitute version numbers, checkout source repositories or execute a script.

Screenshot: Configuring tasks for a job



Creating a job

This page describes how to create a Bamboo [job](#) in a [stage](#) of a [plan](#).

- You can either create a new job, or clone an existing job.
- You must have the 'Admin' or 'Create Plan' [global permission](#) to create jobs.
- A job allows you to collect together a number of [tasks](#) that you want to be run sequentially on the same agent.

Related pages:

- [Configuring plans](#)
- [Using stages in a plan](#)
- [Configuring jobs](#)
- [Disabling or deleting a job](#)

To create a new job for a plan:

1. Click **Dashboard** and then the **All Plans** tab.
2. Click the name of the plan in the list.
3. Choose **Actions > Configure Plan**.
4. Click on the **Stages** tab.
5. Click **Create Job** in the stage where you want the new job.
6. Click either **Create a New Job** or **Clone an Existing Job**.
7. If cloning a job, complete the 'Job to clone from' section:
 - **Plan to clone from** — Select the plan containing the job you wish to clone. Plans are grouped by project in the list.
 - Only plans for which you have the 'Clone' and/or 'Admin' [plan permission](#) are shown.
 - **Job to clone** — Select the job you wish to clone from your selected plan. Jobs are grouped by stage in the list.
8. Complete the 'Job Details' section.
9. Select **Yes please!** to enable this job, if required. Enabling the job instructs Bamboo to execute the job whenever the job's plan is built.
10. Click **Create Job**.

If you wish to configure tasks for the job, such as configuring a Repository Checkout, please refer to [Configuring jobs](#).

Screenshot: Cloning an existing job

On this page, you can make a copy of a Job and its entire configuration.

Plan to clone from

Plan to clone from

Job to clone

This will be used as the template for creating your new job.

Job Details

Job Name*

How do you want to identify the Job within the Plan? Names must be unique within the Plan. e.g. "Core Branch".

Job Key*

This is the key for the job which must be unique within a plan. In conjunction with the plan key, it is used to identify a build in URLs, trigger scripts and API calls. The key must contain only uppercase alphanumeric characters. e.g. "CORE"

Job Description

This is an additional description of the Job. It will help you distinguish your Jobs.

Enable this Job?

☒ Yes please!

By selecting this option, your Job will be available for building. Do not select this option if you have advanced configuration changes to make after creation.

Configuring jobs

A Bamboo *job* is a single build unit within a [plan](#). One or more jobs can be organised into one or more [stages](#). The jobs in a stage can all be run at the same time, if enough Bamboo agents are available. A job allows you to collect together a number of [tasks](#) that you want to be run sequentially on the same agent.

You must have the 'Admin' or 'Create Plan' [global permission](#) to configure jobs.

Related pages:

- [Creating a job](#)
- [Disabling or deleting a job](#)
- [Viewing a job's Maven dependencies](#)

To configure an existing job in a Bamboo plan:

1. Go to the [configuration pages](#) for the plan that has the job.
2. Click on the required job (under 'Stages & Jobs'), and then appropriate tab to begin editing that aspect of your job:
 - **Job Details** — Note that Job Key is not editable.
 - **Tasks** — see [Configuring tasks](#), including Repository Checkout tasks and builder tasks.
 - **Requirements** — see [Configuring a job's requirements](#).
 - **Artifacts** — see [Configuring a job's build artifacts](#).
 - **Miscellaneous** — see [Configuring miscellaneous settings for a job](#) and [Configuring automatic labelling of build results](#).

Configuring a job's requirements

This page describes how to configure the requirements of a job.

A *requirement* is specified in a [job](#) or a [task](#). A requirement specifies a [capability](#) that an [agent](#) must have for it to [build](#) that job or task. A job inherits all of the requirements specified in its tasks.

Together, capabilities and requirements control which agents can execute builds for particular [jobs](#). Each job can only be built by agents whose capabilities match the job's requirements.

There are four types of capabilities in Bamboo that can be specified by job and task requirements:

- **Executable capabilities** — Define external programs that can be called by Bamboo, for example [Ant](#), [Maven](#), [MSBuild](#) or [PHPUnit](#). See [Defining a new executable capability](#).
- **JDK capabilities** — Define the JDK versions to be used by the job or task. See [Defining a new JDK capability](#).
- **Version control capabilities** — Specify the VCS client application that Bamboo should use to check out source code. See [Defining a new version control capability](#).
- **Custom capabilities** — Can be used to control which [jobs](#) will be built by a particular agent. For example, if the builds for a particular job should only run in a Windows environment, you could create a custom capability of 'operating.system=WindowsXP' for the appropriate agent(s), and specify it as a requirement for this job. See [Defining a new custom capability](#).

Before you can specify a requirement in your job, you must first define that capability in your Bamboo system.

On this page:

- [Specifying extra requirements for a job](#)
- [Viewing current capable agents](#)

Related pages:

- [Configuring jobs](#)
- [Configuring tasks](#)
- [Viewing a capability's agents and jobs](#)
- [Viewing a Bamboo agent's details](#)

Specifying extra requirements for a job

A job will inherit the requirements of its tasks by default. However, you can specify extra requirements for a job, in addition to its task requirements.

To specify extra requirements for a job:

1. Navigate to the desired job's configuration pages, as described on [Configuring jobs](#).
2. Click the **Requirements** tab (see [screenshot](#) below). This page shows a list of all the job's current requirements and the number of 'Matching Agents' and 'Matching Images' (i.e. agents/elastic images which meet the job's requirements and can run a build for this job). See [Viewing current capable agents](#) below for more information.
3. If you have previously [set up an agent capability](#), you can select it from the **Requirement** list in the 'Add Extra Requirement' section. If you are setting up a new custom requirement, select **New custom requirement** from that list instead.
4. Complete the form for the requirement:
 - a. **Key**(*new custom requirement only*) — enter a key of the new capability.
 - b. Select the value for the requirement from the list:
 - **exists** — this job can be built by any agent that has a capability with the same key.
 - **equals** — this job can be built by any agent that has the capability with the same key and value.
 - **matches** — this job can be built by any agent that has a capability with the same key, and the value matches the regular expression. For more information about regular expressions, see <http://download.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html#sum>.
5. Click **Add**. The numbers of 'Matching Agents' and 'Matching Images' will be updated, as the plan can now only be built by agents with capabilities that meet the new custom requirement you have specified.

Screenshot: Specifying requirements for a job

Job DetailsTasksRequirementsArtifactsMiscellaneous

RunActions

Requirements

This job can only be built by agents whose capabilities meet the 3 requirements below. 18 agents are capable of building this Job. This Job is capable of building in the Elastic Cloud ⚡ on 11 images.

Capability	Required by	Matching Agents	Matching Images
JDK	JDK exists Unit Tests Task	31	12
Maven 2	Executable exists Unit Tests Task	25	11
Mercurial	Mercurial exists Checkout Default Repository Task	22	15

Add Extra Requirement

The 'Requirement' dropdown contains JDKs and Executables that are setup as capabilities on agents, as well as custom capabilities.

Requirement

New custom requirement...

Key

exists

AddCancel

Viewing current capable agents

To view details about agents or elastic images that are currently able to build your job:

1. On the job's **Requirements** tab ([described above](#)), click the name of the requirement in the table (e.g. 'Maven 2').
2. The summary page for the capability will be displayed, showing the agents and elastic images that have the capability. See [Viewing a capability's agents and jobs](#) for more information.

Configuring a job's build artifacts

Artifacts are files created by a [job build](#) (e.g. JAR files). Artifact definitions are used to specify which artifacts to keep from a build and are configured for individual jobs.

See [Configuring artifact sharing between jobs](#).

This page describes how to define the artifacts that should be kept from a job's build. For example, you may wish to keep reports, websites or files (e.g. JAR files) generated by a job build.

You can also configure artifact sharing between jobs in a plan. For example, you may want to run acceptance tests on a build, and then share the WAR from one job to another, without rebuilding the WAR each time. Artifact sharing is described on this page: [Configuring artifact sharing between jobs](#).

On this page:

- [Defining which artifacts to keep for a job](#)
- [Notes](#)

Related pages:

- [Viewing a build's artifacts](#)
- [Configuring artifact sharing between jobs](#)

Atlassian blog posts:

- [Artifact passing for agile teams](#)

Screenshot: Artifact definitions for a job

Job DetailsTasksRequirementsArtifactsMiscellaneous

RunActions

Artifact Definitions

Create Definition

Jobs may create Artifacts you want to keep or share with others Jobs (e.g. Clover, Agilifier, JAR files, etc).Artifact Definitions allow you to define which Artifacts you care about.

Name	Location	Copy Pattern	Operations
Distributions	target/checkout/distribution/standalone/target/install4j	**/*.*	Unshare Edit Delete
Source	target/checkout/distribution/source/target	*.zip	Unshare Edit Delete
WAR	target/checkout/distribution/war/target/	*.war	Unshare Edit Delete

Artifact Dependencies

Jobs can use shared Artifacts that have been created in previous Stages. Specify the Artifacts your Job will use and the directory where it expects to find them.

Artifact	Destination directory	Operations
Angry Nerds Stable Website	Root of Working Directory	Edit Delete

Defining which artifacts to keep for a job

You can specify which artifacts to keep by setting up an artifact definition for the job. The artifacts will be available after each build of a job.

To set up an artifact definition for a job:

1. Navigate to the desired job, as described on [Configuring jobs](#).
2. Click the **Artifacts** tab, and then **Create Definition**.
3. Complete the fields on the screen (see [screenshot](#) below) and click **Create**. For example, if you want to keep the latest version of a JAR you have built, you could specify **Copy Pattern** to be `**/*.jar` and the **Location** to be `'target'`.

Please note:
 - The location is relative to the build directory. Do not use the absolute path to refer to the location.
 - The copy pattern is relative to the location specified.
 - Asterisks are not supported for **Location**. For this field, provide the folder name where the file would be located.
 - If you want to share artifacts with other jobs in the plan, you will need to mark the artifacts as shared. Sharing artifacts is described on this page: [Configuring artifact sharing between jobs](#).

Screenshot: Creating an artifact definition

Create Definition

Create a new Artifact Definition

Name *

Latest JAR

The Artifact name must be unique within the Plan when the Artifact is marked as "shared"

Location

target

Specify the directory (including path) where Bamboo will look for your artifact. e.g. *build/clover*

Copy Pattern *

**/*.jar

Specify the name (or [Ant file copy pattern](#)) of the artifact(s) you want to keep. e.g. ***/*.**

☐ Shared

Make the Artifact available to other Jobs in this Plan.

Notes

Artifacts are copied to a subdirectory (`/JOB_KEY/download-data/`) under your 'Build Directory' folder (see [Locating important directories and files](#)). Artifacts which you define in the plan are listed in each build result as *artifacts* (see [Viewing a build's artifacts](#)).

Configuring artifact sharing between jobs

This page describes how to configure artifact sharing between jobs in a plan. Artifact sharing allows you to pass an artifact from one job to a job in a subsequent stage. That is, the artifact is copied to the subsequent job's agent. Note, you cannot pass artifacts between jobs in the same stage.

For example, you may want to run acceptance tests on a build, sharing the same WAR from one job to another without rebuilding it each time.

This page also describes how to configure artifact sharing when you are using a Maven builder. In this case, the artifact is deployed to, and resolved from, a Maven repository by jobs, rather than being copied from agent to agent.

Sharing artifacts between jobs

You can share artifacts between jobs in different stages using artifact dependencies. Each time the artifact is shared with a subsequent job, it is copied to the job's agent.

To share an artifact between two jobs in different stages:

1. Navigate to the configuration pages for the job that will produce the artifact, as described on [Configuring jobs](#).
2. Click the **Artifacts** tab (see [Configuring a job's build artifacts](#)).
3. Click **Edit** for the artifact that you want to share. The artifact definition will be displayed.
4. Select the **Shared** check box and then click **Save**.
5. Navigate to the job in a subsequent stage that will consume the artifact, and click the **Artifacts** tab.
6. Click **Create Dependency**.
7. Complete the fields on the screen (see [screenshot](#) below) and click **Create**.

Please note:

- The **Artifact** list only shows artifacts from jobs in previous stages that have been marked as shared. This is described in [Configuring a job's build artifacts](#).
- **Destination directory** is relative to the build directory. Do not use the absolute path to refer to the destination directory.

On this page:

- [Sharing artifacts between jobs](#)

Related pages:

- [Viewing a build's artifacts](#)
- [Configuring a job's build artifacts](#)
- [Pattern matching reference](#)

Atlassian Blogs:

- [Boost Your Build Automation with Artifact Sharing](#)

Screenshot: Creating an artifact dependency

Create Dependency

This will create new Artifact Dependency for this Job

Artifact*

Core POM

▼

Name of the Shared Artifact required by this Job.

Destination directory

target

The directory where the Artifact will be available for your build, relative to the Build's working directory.

Create

Cancel

Configuring miscellaneous settings for a job

For each job of a plan, you can optionally specify a number of miscellaneous settings including:

- Build hanging detection
- NCover output
- Clover code coverage

To configure the miscellaneous settings for a job:

1. Navigate to the desired job, as described in [Configuring jobs](#).
2. Edit the desired settings as follows:

Override default hanging build detection	Override the default build hanging detection settings. These settings determine when a build hung event is thrown (e.g. you can configure your notifications to trigger from this event). Build Time Multiplier — Calculate the 'Expected Build Time' for the build (i.e. 'Expected Build Time' = 'Build Time Multiplier' multiplied by 'Average Build Time'). 'Average Build Time' is calculated by using an average of previous build times. Log Quiet Time — The amount of time since Bamboo last recorded an entry in the build log for a build. The 'Expected Build Time' and 'Log Quiet Time' must <i>both</i> be exceeded for Bamboo to throw the build hung event. Build Queue Timeout — The amount of time that a build will wait in a build queue before an timeout event is thrown. Setting this value will override the global build queue timeout setting (see Configuring the build queue timeout event).
NCover output will be produced	<i>Do not select this option. NCover is a code coverage tool that supports .NET projects.</i>
Use Clover to collect Code Coverage for this build	Select this check box if: • This job will be building a Java or Groovy-based project using a builder such as Ant, Maven or Grails. • You are running Atlassian Clover and want to collect code coverage data to view from within Bamboo (see Viewing the Clover code-coverage for a build).

Automatically integrate Clover into this build

- **Generate a Clover Historical Report** — shows the current coverage results compared with previous Clover code coverage reports.
- **Generate a JSON report** — gives the Clover results in a format ready for embedding into applications or external report views.

You will also need to insert a Clover license ([evaluation licenses](#) are available) into the field provided. See [Enabling the Clover add-on](#).

Clover is already integrated into this build and a clover.xml file will be produced

Use this option when you already have Clover-for-Ant or Clover-for-Maven configured to generate a report.

- **Clover XML Location** — specify where the Clover XML report is generated. Include the name of the directory, including path, *relative* to your job build's root directory, for example: `target/site/clover/clover.xml`

3. Click **Save**.

Configuring automatic labelling of job build results

For each job of a plan, you can (as an option) specify a label that can be applied to the job's build results automatically after each build of that job.

Automatic labelling of job builds is built into Bamboo itself. There are a number of [third-party plugin modules](#) available that can provide additional 'post' actions (e.g. the Pre-Post Build Command plugin). You can also write your own plugins to provide additional post actions for a job. See the [Bamboo Plugin Guide](#) for further details.

A *label* is a convenient way to tag and group *build results* that are logically related to each other. Labels can also be used to define [RSS feeds](#) and to control [build expiry](#).

Labels can be applied to build results automatically, by specifying the label(s) in a plan (note that only Bamboo administrators can do this). Labels can also be [applied ad hoc](#) to build results by Bamboo users.

On this page:

- [Specifying labels for a job's build results](#)
- [Regex examples](#)

Related pages:

- [Configuring jobs](#)

Specifying labels for a job's build results

To specify labels for a job's build results:

1. Navigate to a job's configuration pages, as described on [Configuring jobs](#).
2. Click the **Miscellaneous** tab.
3. Using **Regex Pattern**, you can either:
 - Specify a regular expression to match content in the log files of this job's builds. Labels will be applied to a build of this job if this regular expression matches content in the build's log files (see the examples below).
 -  For more information about regular expressions, please refer to the [Java documentation on regular expression constructs](#).
 - Leave this field blank to label every build of this job.
4. In the **Labels** field, type the word (or multiple words, separated by commas and/or spaces) with which the plan's build results are to be [labelled](#).
5. Click **Save**.

Regex examples

See <http://www.regular-expressions.info/reference.html> for examples.

A simple regex example:

```
'There are \d+ results'
```

In the above regex, `\d+` represents *any number with one or more digits*. ('`d`' means 'any digit', and '`+`' means 'one or more times'. When combined, they mean 'any sequence of one or more digits'.) Therefore, positive matches would include:

- 'There are 0 results'
- 'There are 123 results'

A regex example with multiple labels:

You can use "capturing groups" with Bamboo 1.2.1 or later to create different labels for different purposes.

For example, the following settings will label your builds with PERFORMANCE_IMPROVED if "PERFORMANCE_IMPROVED" appears in the build log, and PERFORMANCE_DETERIORATED if "PERFORMANCE_DETERIORATED" appears in the build log. If both strings appear in a log, then both labels are applied to the build.

- Enter the following into the **Regex Pattern** field:

```
( PERFORMANCE_IMPROVED | PERFORMANCE_DETERIORATED )
```

- Enter the following into the **Labels** field:

```
\1
```

Viewing a job's Maven dependencies

If you have configured a job to use a Maven builder (Maven 2 or later), you can choose to have dependencies generated from your Maven pom.xml (see [documentation for setting up Maven as a builder](#) for instructions). After the initial build, Maven will parse the pom.xml file, determine the artifacts produced by the build and generate the dependencies. You can view these dependencies in two places:

- On the **Dependencies** tab when configuring your plan, as described in [Setting up plan build dependencies](#).
- On the **Artifacts** tab when viewing a job's build result, as described below.

Before you begin:

- The Maven dependencies for a build will only become known to Bamboo after a build. If you cannot see the Maven dependencies for a build, try running it first without triggering any other dependencies. See [Modifying multiple plans in bulk](#) if you want to run multiple builds.

To view the Maven dependencies for a job's build result:

- Navigate to the desired job, as described on [Configuring jobs](#).
- Click the desired build result number in the 'Recent History' of the Job Summary.
- Click the **Artifacts** tab for the build results. The produced Maven artifacts and Maven artifact dependencies will be listed.

Screenshot: Maven 2 dependencies for a job's build result

Job SummaryTestsChanges**Artifacts**LogsMetadataIssues

▶ Run ⚙ Actions

Artifacts

Produced Artifacts

This Job has produced the following Artifacts:

Name	Size
 Angry Nerds Stable Website	1 MB

Shared artifacts can be found under:
/Volumes/Pharlap/opt/dogfood/panda/home/artifacts/ANGRY-ANMP/shared/build-00043

Artifact Dependencies

This Job has no Artifact Dependencies.

Produced Maven Artifacts

No maven artifacts have been produced for this Job

Maven Artifact Dependencies

There are no maven artifacts depended on for this Job

Disabling or deleting a job

Bamboo allows you to disable or delete jobs that you don't want to be built:

- **Disabling a job** prevents it from being built. Disabling a job prevents Bamboo from building that particular job within a plan, allowing the rest of the plan's jobs to be built. You can re-enable the job, if you want to build it again.

For example, if a job's latest build is broken and cannot be fixed quickly, you may want to disable it temporarily to stop the job from being built.

- **Deleting a job** removes it completely from your Bamboo system. You will need to recreate a new job from scratch, if you want to build it again.

For example, if a job is no longer relevant, you may want to delete it.

On this page:

- [Disabling a job](#)
- [Deleting a job](#)

Related pages:

- [Creating a job](#)
- [Configuring jobs](#)
- [Disabling or deleting a plan](#)
- [Deleting a job's current working files](#)

Disabling a job

To disable a job:

1. Navigate to the job configuration, as described on [Configuring jobs](#).
2. Choose **Actions** > **Disable Job**.

Deleting a job

Deleting a job deletes everything related to that job, including the job's configuration, build results, artifacts, labels and comments. However, everything else related to the job's plan, and this plan's other jobs, is retained by Bamboo.

Before you begin:

- The 'Admin' [global permission](#) is required to delete a job.
- A job that is currently being built cannot be deleted. If you need to delete such a job, stop the plan's build first. Refer to [Stopping an active build](#) for more information.
- If you need to keep a permanent record of the job build results, see [Exporting data for backup](#).

To delete a job:

1. Navigate to the job configuration, as described on [Configuring jobs](#).
2. Choose **Actions** > **Delete Job**.

Deleting a job's current working files

If you only run a single Bamboo server (i.e. with no remote or elastic agents) and you:

- need to ensure that a plan's job cleanly checks out its source code when Bamboo next executes a build of that plan *and*
- do not to use the **Force Clean Build** option when [specifying the source repository](#) for a job.

Simply delete the current working files for that job to ensure its source code is cleanly checked out.

Before you begin:

- Only people with the 'Admin' [global permission](#) or the 'Admin' [plan permission](#) can delete current working files.

Related pages:

- [Specifying the source repository](#)
- [Disabling or deleting a job](#)
- [Deleting the results of a plan build](#)

To delete a job's current working files:

1. Navigate to the job configuration, as described on [Configuring jobs](#).
2. Click the **Files** tab. Note that the **Files** tab is only available if:
 - the current working files resulting from that job's previous build reside on the Bamboo server (not a remote/elastic agent) and
 - working files exist in this directory.
3. At the end of this page (scroll down if necessary), click **Delete all build files**.

Configuring tasks

A task:

- Is a small discrete unit of work, such as source code checkout, executing a Maven goal, running a script, or parsing test results.
- Is run sequentially within a [job](#) on a Bamboo working directory.

Tasks may make use of an [executable](#) if required. Once a task is defined in the Bamboo system, it can then be specified in jobs by a plan administrator. A job can be configured to execute a number of tasks, on the same working directory. For example, before executing a Maven goal, the user could substitute specific files within the working directory, substitute version numbers, checkout source repositories or execute a script.

Creating a task for a job

When creating a new [job](#) or configuring an existing one, you need to specify the [tasks](#) that will execute the job's builds. You must specify an [executable](#) for each task. If you specify an [Ant](#), [Grails](#) or [Maven](#) executable, you will also need to choose a [JDK](#).

When [creating a new plan](#), you can configure the tasks for the plan's default job.

On this page:

- [Creating a task for a job](#)
- [Ordering the tasks in a job](#)
- [Notes](#)

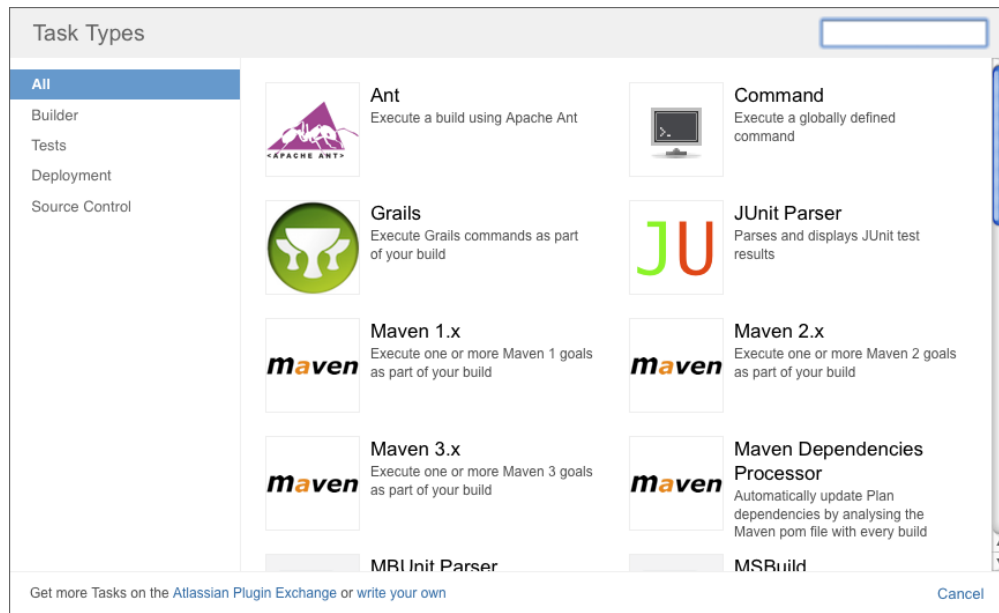
Related pages:

- [Checking out code](#)
- [Configuring a builder task](#)
- [Configuring a test task](#)
- [Configuring jobs](#)
- [Creating a plan](#)
- [Pattern matching reference](#)

To create a task for a job:

1. Navigate to the tasks configuration for the desired job, i.e.
 - navigate to the **Tasks** tab when [configuring an existing job](#), or
 - [create a plan](#) (you will be configuring tasks for the default job).
2. Click **Add Task**.
3. Click the desired task type in the 'Task Types' dialog.
4. Update the values to configure the task as desired. The fields displayed will vary depending on the [executable](#) chosen. See the following pages for further details:
 - [Checking out code](#)
 - [Configuring a builder task](#)
 - [Configuring a test task](#)
 - [Configuring a deployment task](#)
 - [Pattern matching reference](#)
5. Click **Save**.

Screenshot: *Specifying a task for a job — Task Types*



Ordering the tasks in a job

Tasks can be designated as *build tasks* or *final tasks* in a job:

- Build Tasks will run sequentially in the order specified in the job. If a Build Task fails, all subsequent tests will not be executed.
- Final Tasks will run sequentially, once the build tasks have completed. Final Tasks will always be executed, regardless of whether any Build Tasks or other Final Tasks fail. Final Tasks will be executed even if you stopped the build manually.

To order the tasks for a job:

1. Navigate to the tasks for the desired job.
2. Drag and drop the tasks into the desired order in the table on the left. If you want to change a Build Task to a Final Task or vice versa, drag and drop it under the desired header in the table. Your changes will be saved immediately.

Screenshot: Existing Task — Maven 2

Tasks

A Task is a piece of work that is being executed as part of the Build. The execution of a script, a shell command, an Ant Task or a Maven goal are only few examples of Tasks. [Learn more about Tasks.](#)

You can use [Runtime](#), [Plan](#) and [Global variables](#) to parameterize your Tasks.

23 agents are capable of building this Job. This Job cannot be built in the Elastic Cloud. ⚡

Source Code Checkout

Maven 2.x

Final Tasks are always executed at the end of the build

Drag tasks here to make them final

Add Task

Maven 2.x Configuration

Task Description

Executable

Maven 2 [Add New Executable](#)

Goal *

clean test

The goal you want to execute. You can also define system properties such as `-Djava.awt.headless=true`.

Build JDK *

JDK [Add New JDK](#)

Which JDK do you need to use for the build?

Override Project File

(Optional) Path to the project file, relative to the working sub directory. If left blank maven will use the pom.xml in the root of the working sub directory

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. `JAVA_OPTS="-Xmx256m -Xms128m"`. You can add multiple parameters separated by a space.

Working Sub Directory

(Optional) Specify an alternative sub-directory as working directory for the task.

Where should Bamboo look for the test result files?

☒ The build will produce test results.
If checked, the build will fail if no tests are found. Test output must be in [JUnit XML](#) format.

Test Results Directory

☒ Look in the standard test results directory.

☐ Specify custom results directories

Where should Bamboo look for the test result files?

Save Cancel

Notes

- **Adding new executables** — At least one executable is configured automatically after installing Bamboo. You can add more executables of different types as described in [Configuring a New Executable](#).
- **Adding new JDKs** — At least one JDK is configured automatically after installing Bamboo. You can add more JDKs as described in [Defining a new JDK capability](#).
- **About the 'Compatibility Task'** — The 'Compatibility Task' is created by Bamboo when upgrading from Bamboo 3.0 or earlier and Bamboo cannot match a builder to a task. This may occur if you are using a builder enabled by a custom plugin. For more information, see the [Bamboo 3.1 Upgrade Guide](#).

Checking out code

Bamboo uses the 'Source Code Checkout' task to check out repositories into the working directory on the agent.

Use this task it is also possible to:

- Check out multiple repositories to a custom directory path in the working directory
- Specify multiple checkouts that occur at different stages of the build. (This can be achieved by simply adding another 'Source Code Checkout' task to a job at any point in the plan.)

To configure a new Source Code Checkout task:

1. Navigate to the job that should perform the task, as described on [Specifying the source repository](#).
2. Click the **Tasks** tab, and select an existing 'Source Code Checkout' task in the tasks list, or add a new one using the **Add Task** button.
3. Configure the task:

Task Description	Enter a description of the task, for display in Bamboo.
Disable this task	Check, or clear, to selectively run this task.

Repository	Select the desired repository. If you wish to add different types of repositories, they must have been previously defined on the plan's Source Repositories tab. See Specifying the source repository for a list of supported SCMs.
Checkout Directory	The location to which the contents of the selected repository will be checked out to when the task executes.
Force Clean Build	Deletes the previously checked out directory and checks it out again prior to the next build. This may significantly increase build times.

- Click **Add Repository**, at the bottom of the 'Task' screen, to add another repository to the task.
- Click **Save**.

Screenshot: Configuring a Source Code Checkout task

Job Details
Tasks
Requirements
Artifacts
Miscellaneous

Run
Actions

Tasks

A Task is a piece of work that is being executed as part of the Build. The execution of a script, a shell command, an Ant Task or a Maven goal are only few examples of Tasks. [Learn more about Tasks](#).

You can use [Runtime](#), [Plan](#) and [Global variables](#) to parameterize your Tasks.

70 agents have the [capabilities](#) to run this Job

Source Code Checkout
Checkout Default Repository

Final Tasks are always executed at the end of the build
Drag tasks here to make them final
Add Task

Source Code Checkout Configuration

Task Description

Checkout Default Repository

☐ Disable this task

You can check out one or more repositories with this Task. You can choose to check out the Plan's *Default Repository* or specify a *Specific Repository*. You can add additional repositories to this Plan via the [Plan configuration](#).

Repository*

Default Repository (Bitbucket)

Default always points to Plan's default repository.

Checkout Directory

(Optional) Specify an alternative sub-directory to which the code will be checked out.

☐ Force Clean Build
Removes the source directory and checks it out again prior to each build. This may significantly increase build times.

+ Add Repository

Save Cancel

Notes

- A number of source repositories are supported 'out of the box', as described on the [Connecting to code repositories](#) page.
- If you need to use a type of repository that is not supported, a number of [third-party Source Repository plugin modules](#) are available (e.g. ClearCase plugin). You can also write a [Source Repository Module plugin](#) to enable Bamboo to connect to your repository.

Configuring a builder task

A builder task allows you to connect your Bamboo plan (or job) to a build tool such as Ant, Maven or MSBuild. The build tool uses its existing configuration when the plan (or job) is built.

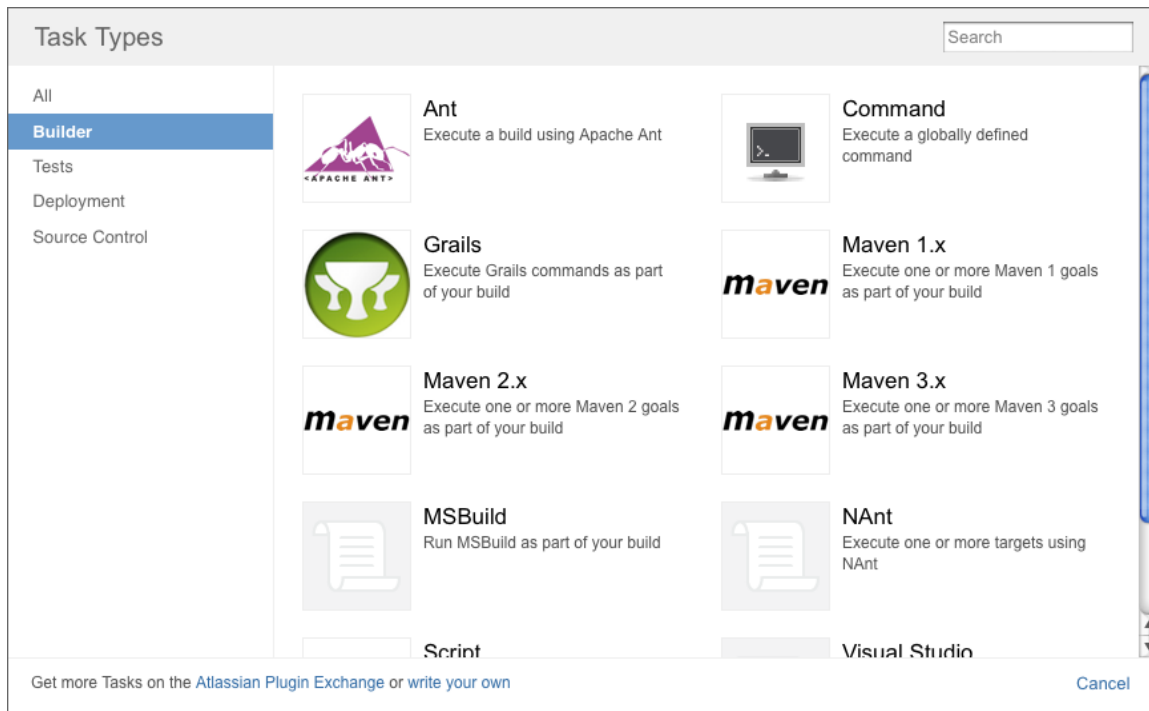
You can connect Bamboo to the following build tools:

- Ant
- Custom command executable
- Grails
- Maven
- MSBuild
- NAnt
- Script
- Visual Studio
- Xcode

Related pages:

- Configuring tasks
- Configuring a test task
- Checking out code

Screenshot: Choosing a Bamboo build task



Ant

This page describes how to configure a Bamboo task to use Ant.
See [Configuring a builder task](#) for an overview of Bamboo builder tasks.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Pattern matching reference](#)

To configure an Ant task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing Ant task, or click **Add Task** and then **Ant** to create a new task.
3. Complete the following settings:

Task Description	A description of the task, which is displayed in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
Executable	The Ant executable that is available to perform the task. The executable that you select will become one of the task's (and so, the job's) requirements. You can add other executables , if required.
Build File	The name of your existing build file (e.g. <code>build.xml</code>). You can include variables (see Using Global or Build-specific Variables).

Target	The Ant target that you want this Bamboo task to execute (e.g. <code>test</code>). You can use <code>'-D'</code> to define one or more JVM parameters (e.g.: <code>-Djava.awt.headless="true"</code>). You must use double quotes around the parameter value; single quotes are considered as part of the actual value. Multiple Ant targets can be specified with a space-delimited list. You can also include variables (see Using Global or Build-specific Variables).
Build JDK	The JDKs that are available to perform the task. The JDK that you select will become one of the task's (and so, the job's) requirements. You can add other JDKs, if required.
Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Using global, plan or build-specific variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. <code>ANT_OPTS="-Xms200m -Xmx700m"</code>).
Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.
The build will produce test results	Select to specify the directory, relative to the root directory, where test results will be created. You can use Ant-style patterns such as <code>**/test-reports/*.xml</code>. Bamboo requires test results to be in JUnit XML format.  For jobs that use CVS, the root directory is <code><bamboo-home>/xml-data/build-dir/JOB_KEY/<cvsmodule></code>.

4. Click **Save**.

Ant Configuration

Task Description

☐ Disable this task

Executable

Ant

Build File

Target*

clean test

The target you want to execute
Djava.awt.headless=true.

Build JDK*

JDK 1.7

Which JDK do you need to use?

Environment Variables

(Optional) Any extra environment variables
Xmx256m -Xms128m". You

Working Sub Directory

(Optional) Specify an alternative working sub directory

Where should Bamboo

☒ The build will produce reports
If checked, the build will

Specify custom results

**/test-reports/*.xml

Where does the build place its results?
This is a comma separated list of paths
**/test-reports/*.xml

Save Cancel

Custom command executable

This page describes how to configure a Bamboo task that uses a command (e.g. Bash) executable.

See [Configuring a builder task](#) for an overview of Bamboo builder tasks.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)

To configure a command task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing Command task, or click **Add Task** and then **Command** to create a new task.
3. Complete the following settings:

Task Description	A description of the task, which is displayed in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
Executable	The command executable that is available to perform the task (e.g. Bash). The executable that you select will become one of the task's (and so, the job's) requirements. You can add other executables , if required.
Argument	<i>(Optional)</i> The relevant argument to pass to the command. Note that arguments which contain spaces must be quoted. You can include variables (see Bamboo variables).
Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Using global, plan or build-specific variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. ANT_OPTS="-Xms200m -Xmx700m").
Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.

4. Click **Save**.

Command Conf

Task Description

☐ Disable this tas

Executable

Argument

(Optional) Argument yc
them must be quoted

Environment Variat

(Optional) Any extra er
JAVA_OPTS="-Xmx25
by a space.

Working Sub Direct

(Optional) Specify an a

Grails

This page describes how to configure a Bamboo Grails task.

Bamboo supports Grails versions 1.2.x, 1.3.x, and 2.x.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Defining a new JDK capability](#)

To configure a Grails task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing Grails task, or click **Add Task** and then **Grails** to create a new task.
3. Complete the following settings:

Task Description	A description of the task, which is displayed in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
Executable	The Grails executable that is available to perform the task. The executable that you select will become one of the task's (and so, the job's) requirements. You can add other executables , if required.

Grails Commands	The Grails commands that you want Bamboo to execute. See the Grails Command Line Reference documentation for more details on Grails commands. - You can use ' -D ' to define one or more JVM parameters, e.g.: -Djava.awt.headless=true will pass the parameter 'java.awt.headless' with a value of 'true'. - You can include variables (see Bamboo variables).
Build JDK	The JDKs that are available to perform the task. The JDK that you select will become one of the task's (and so, the job's) requirements. You can add other JDKs, if required.
Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Using global, plan or build-specific variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. ANT_OPTS="-Xms200m -Xmx700m").
Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.
The build will produce test results	Choose one of the following: Look in the standard test results directory – Bamboo looks in the standard directory for the test results. Use this unless you've customised your test runner to output the results to a different location. Specify custom results directories -- Specify the custom directory, relative to the root directory, where test results will be created. You can use Ant-style patterns such as **/test-reports/*.xml. Bamboo requires test results to be in JUnit XML format.  For jobs that use CVS, the root directory is <bamboo-home>/xml-data/build-dir/JOB_KEY/<cvsmodule>.

4. Click **Save**.

Grails Configuration

Task Description

☐ Disable this task

Executable

Grails  [Add New Executable](#)

Grails Commands*

```
clean
test-app
```

Use a new line to separate Grails commands. Bamboo will automatically append "-non-interactive" to each command

Build JDK*

JDK 1.7  [Add New JDK](#)

Which JDK do you need to use for the build?

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters separated by a space.

Working Sub Directory

(Optional) Specify an alternative sub-directory as working directory for the task.

Where should Bamboo look for the test result files?

- ☒ The build will produce test results.
If checked, the build will fail if no tests are found. Test output must be in [JUnit](#) XML format.

Test Results Directory

- ☒ Look in the standard test results directory.
☐ Specify custom results directories

Where should Bamboo look for the test result files?

[Save](#) [Cancel](#)

Maven

This page describes how to configure a Bamboo task to use a Maven executable. [Apache Maven](#) is a tool used for building and managing Java-based projects.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Viewing a job's Maven dependencies](#)
- [Defining a new JDK capability](#)

Atlassian blogs:

- [Forgetful Maven Users, Rejoice! A new Bamboo task “releases” you from worry.](#)

To configure a Maven task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing Maven task, or click **Add Task** and then a Maven option (e.g. **Maven 2.x**) to create a new task.
3. Complete the following settings:

Task Description	A description of the task, which is displayed in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
Executable	The Maven executable that is available to perform the task. The executable that you select will become one of the task's (and so, the job's) requirements. You can add other executables , if required.
Goal	The Maven goal that Bamboo will execute. • You can use '-D' to define one or more JVM parameters. For example, <code>-Djava.awt.headless=true</code> will pass the parameter 'java.awt.headless' with a value of 'true'. • Multiple maven goals can be specified, separated spaces. • You can include variables (see Using Global or Build-specific Variables).
Use Maven Return Code	Select to have Bamboo skip log parsing.
Build JDK	The JDKs that are available to perform the task. The JDK that you select will become one of the task's (and so, the job's) requirements. You can add other JDKs , if required.
Override Project File	<i>(Optional: Maven 2.x and later only)</i> The path to your Maven project file, relative to the working sub directory specified. If this is not specified, Maven will use the <code>pom.xml</code> in the root of the working sub directory.
Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Using global, plan or build-specific variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. <code>ANT_OPTS="-Xms200m -Xmx700m"</code>).

Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.
The build will produce test results	Choose one of the following: Look in the standard test results directory or Specify custom results directories – Specify the alternative directory, relative to the root directory, where test results will be created. You can use Ant-style patterns such as <code>**/test-reports/*.xml</code> . Bamboo requires test results to be in JUnit XML format. ⚠ For jobs that use CVS, the root directory is <code><bamboo-home>/xml-data/build-dir/JOB_KEY/<cvcs-module></code> .

4. Click **Save**.

Figure 3.8 Configuration

Task Description

☐ **Disable this task**

Executable

MSBuild

Goal

The goal you want to execute. You can also define system properties such as `<java.awt.headless>`.

☐ **Use Maven Platform Code**

When downloading code to test, Bamboo creates Maven index code and starts the task for "BUILD-2007-03-07". By checking this option, you will also require Bamboo to start the task. This may be the case when running continuous systems.

Build IDE

MSB 1.2

Which IDE do you want to use for the build?

Override Project File

(Optional) Path to the project file (usually build.xml) in the working subdirectory. If it is blank, Bamboo will use the project in the root of the working subdirectory.

Environment Variables

(Optional) Any extra environment variables you want to pass to your build (e.g. `BAMBOO_OPTS`, `MSB_OPTS`, `MSB_OPTS`). You can add multiple parameters by separating them with a space.

Working Sub Directory

(Optional) Specify an alternative subdirectory as working directory for the task.

Where should Bamboo look for the test result files?

☒ **The build will produce test results.**

Choosing the subdirectory to look for test results. The output must be in JUnit XML format.

Test Results Directory

☒ **Look in the standard test results directory**

☐ **Specify custom results directories**

Where should Bamboo look for the test result files?

MSBuild

This page describes how to configure a Bamboo task to use an MSBuild executable.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)

To configure an MSBuild task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of the desired MSBuild task, or click **Add Task** and then **MS Build** if creating a new task.
3. Complete the following settings:

MSBuild configuration	
Task Description	A description of the task, which is displayed in Bamboo.
Executable	The MSBuild executable that is available to perform the task. The executable that you select will become one of the task's (and so, the job's) requirements. You can add other executables , if required.
Project File	The name of the solution, project file or MSBuild project to execute, e.g. ExampleSolution.sln. You can include variables (see Bamboo variables).
Options	The MSBuild command line options that you want to include. You can also include variables (see Bamboo variables).

4. If required, specify environment variables and working directory settings:

Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Bamboo variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. ANT_OPTS="-Xms200m -Xmx700m").
Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.

5. Click **Save**.

Note that you cannot use Clover to collect code coverage for MSBuild builds, as Clover only supports builders of Java/Groovy-based projects, such as Ant, Maven or Grails.

MSBuild Configuration

Task Description

Executable*

MSBuild v2.0 (32bit)  Add

[New Executable](#)

Project File*

YourSolution.sln

The Solution, Project File or MSBuild project to execute when this Job Builds

Options

The MSBuild.exe command line switches you wish to include.

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters separated by a space.

Working Sub Directory



(Optional) Specify an alternative sub-directory as working directory for the task.

Save

Cancel

NAnt

This page describes how to configure a Bamboo task to use a NAnt executable.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)

To configure a NAnt task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of the desired NAnt task, or click **Add Task** and then **NAnt** if creating a new task.
3. Complete the following settings:

NAnt configuration

Task Description

A description of the task, which is displayed in Bamboo.

Executable	The NAnt executable that is available to perform the task. The executable that you select will become one of the task's (and so, the job's) requirements. You can add other executables , if required.
Build File	The relevant file name (e.g. default.build). You can include variables (see Bamboo variables).
Targets	The NAnt target that you want Bamboo to execute. for example: run. You can also include variables (see Bamboo variables).
Options	The NAnt command line options that you want to include. You can also include variables (see Bamboo variables).

4. If required, specify environment variables and working directory settings:

Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Bamboo variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. ANT_OPTS="-Xms200m -Xmx700m").
Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.

5. Click **Save**.

Note that you cannot use Clover to collect code coverage for NAnt builds, as Clover only supports builders of Java/Groovy-based projects, such as Ant, Maven or Grails.

NAnt Configuration

Task Description

Executable*

[Add New Executable](#)

Build File

The name of the NAnt build file that you want to execute when this Job builds

Targets*

The NAnt targets you want Bamboo to execute when this Job builds

Options

The NAnt command line options you wish to include.

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters separated by a space.

Working Sub Directory



(Optional) Specify an alternative sub-directory as working directory for the task.

Save

Cancel

Script

This page describes how to configure a Bamboo task to use a script executable. You can use Bash on Linux, and batch files on Windows.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)

To configure a script task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of the desired script task, or click **Add Task** and then **Script** if creating a new task.
3. Complete the following settings:

Script configuration

Task Description

A description of the task, which is displayed in Bamboo.

Script location	Select the location of the script file. • File — enter the location of the file in the Script file field. This can be either relative to the repository root of the plan, or absolute. You can include variables (see Bamboo variables). • Inline — enter the script in the Script body field.
Argument	Specify an argument to pass to the script. Arguments that contain spaces must be quoted. You can include variables (see Bamboo variables).

4. If required, specify environment variables and working directory settings:

Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Bamboo variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. ANT_OPTS="-Xms200m -Xmx700m").
Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.

5. Click **Save**.

Script Configuration

Task Description

Script location

File

Script file*

Argument

(Optional) Argument you want to pass to the command. Arguments with spaces in them must be quoted

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters separated by a space.

Working Sub Directory

 (Optional) Specify an alternative sub-directory as working directory for the task.

Save

Cancel

Visual Studio

This page describes how to configure a Bamboo task to use a [Visual Studio](#) (devenv.exe) executable.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)

To configure a Visual Studio task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of the desired MSBuild task, or click **Add Task** and then **Visual Studio** if creating a new task.
3. Complete the following settings:

Visual Studio configuration	
Task Description	A description of the task, which is displayed in Bamboo.
Executable	The Visual Studio executable that is available to perform the task. The executable that you select will become one of the task's (and so, the job's) requirements. You can add other executables , if required.

Solution	The name of the Visual Studio solution file that you want Bamboo to execute. For example: RegexDemo/RegexDemo.sln. You can also include variables (see Bamboo variables).
Options	Specify any Visual Studio command-line options that you want to include (e.g. /build Debug). You can also include variables (see Bamboo variables).
Platform	Select the platform toolset required to compile your solution. This is provided as an argument to Vcvarsall.bat (see this MSDN article for more details).

4. If required, specify environment variables and working directory settings:

Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Bamboo variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. ANT_OPTS="-Xms200m -Xmx700m").
Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.

5. Click **Save**.

Visual Studio Configuration

Task Description

Executable*

Visual Studio 2010  Add

[New Executable](#)

Solution*

The Visual Studio solution file you want Bamboo to execute when this Job builds

Options*

The devenv command line options you wish to include.

Platform*

x86 

The platform toolset required to compile your Solution.

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters separated by a space.

Working Sub Directory



(Optional) Specify an alternative sub-directory as working directory for the task.

Save

Cancel

Xcode



Xcode support is undergoing testing and as such is unsupported by Atlassian at this time. If you would like to see it officially supported, please vote for this [improvement request](#).

- [Prerequisites](#)
- [Testing iOS applications](#)
 - [Configuring your Xcode project automated simulator tests](#)
 - [Configuring the Xcode task for testing](#)

Prerequisites

- **Apple Xcode 4** – or later version
- **Certificates and provisioning profiles** – You must install all required developer certificates and provisioning profiles on every machine that Bamboo will use to run your build. See the [App Distribution Guide](#) for more information.
- **Bamboo Xcode support plugin** – The latest [Xcode plugin](#) installed in your Bamboo server.
- **ios-sim** (optional when building Mac applications) – a command line utility used to launch the iOS Simulator from the command line. If you have homebrew installed, you can install it by running `brew install ios-sim`. For other installation methods, see the [ios-sim](#) website.
- **Cocoapods** (optional if you do not have a Podfile in your project) – Cocoapods is the library dependency manager for Mac OS X. In order for Bamboo to install dependencies from your Podfile (if you have created one), Bamboo will need it installed on all systems where the build should run.

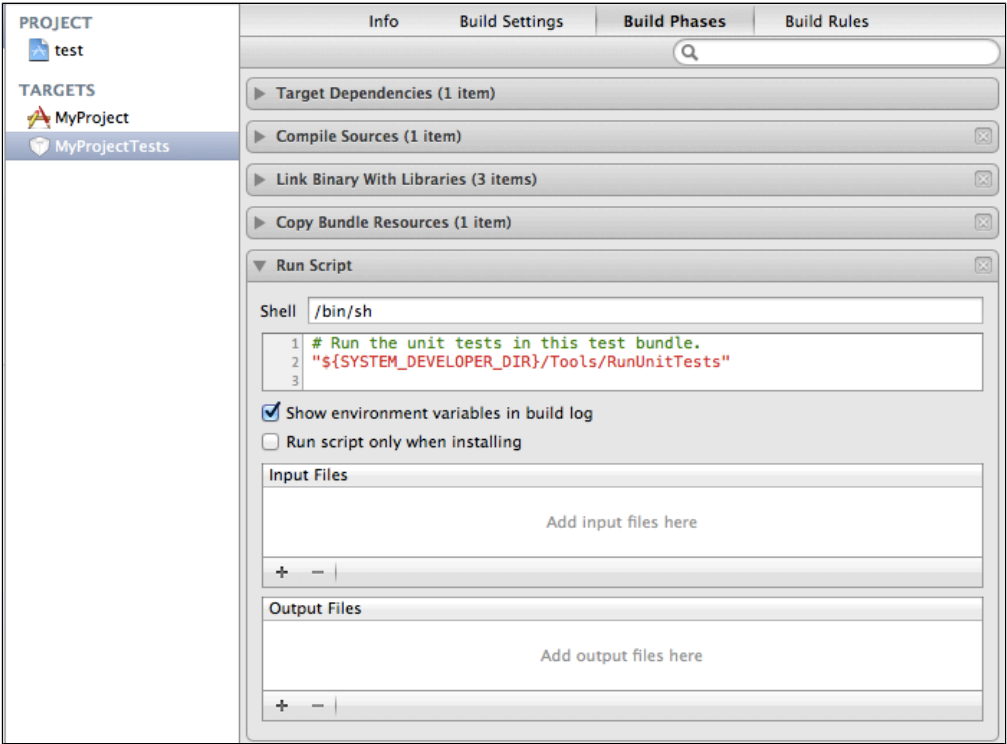
Testing iOS applications

To have tests automatically run on the iOS Simulator and reported within Bamboo you must make some changes to your Xcode project's test bundles and add the Xcode build task to your Job within Bamboo.

Configuring your Xcode project automated simulator tests

Without modifications, Apple does not support running unit tests in the simulator using the `xcodebuild` terminal utility which Bamboo uses to automate builds and tests.

Using `ios-sim` and a small modification to the `RunUnitTests` script phase in the test bundle its possible to overcome this limitation.



Change the content of the script to:

```
if [ "$RUN_UNIT_TEST_WITH_IOS_SIM" = "YES" ]; then
test_bundle_path="$BUILT_PRODUCTS_DIR/$PRODUCT_NAME.$WRAPPER_EXTENSION"
ios-sim launch "$(dirname "$TEST_HOST")" --setenv
DYLD_INSERT_LIBRARIES=/../../Library/PrivateFrameworks/IDEBundleInjection.framework
/IDEBundleInjection --setenv XCInjectBundle="$test_bundle_path" --setenv
XCInjectBundleInto="$TEST_HOST" --args -SenTest All "$test_bundle_path"
echo "Finished running tests with ios-sim"
else
"${SYSTEM_DEVELOPER_DIR}/Tools/RunUnitTests"
fi
```

Configuring the Xcode task for testing

To configure a Xcode to test an iOS project task:

- 1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
- 2. Click the name of an existing Xcode task, or click **Add Task** then Xcode to create a new task.
- 3. Complete the following settings:

Task Description	A description of the task, which is displayed in Bamboo.

- [JUnit Parser](#)
- [JUnit Runner](#)
- [PHPUnit](#)
- [TestNG](#)

Community test task plugins

There are numerous test task plugins available on the [Atlassian Marketplace](#). These plugins are unsupported by Atlassian for the time being but the source code has been made freely available.

Bamboo plugin	Testing framework	Languages and Platforms	Supported by Atlassian?	Source code	Issue tracking adding official support
Bamboo Xcode Task	OCUnit	Objective-C, Apple iOS, Cocoa and Mac OS X	NO	Available on Bitbucket	 BAM-6149 - Provide official support for the Bamboo Xcode plugin ( Open)
Bamboo Ruby Plugin	RSpec	Ruby	NO	Available on Github	BAM-12328 - Authenticate to see issue details
Bamboo CppUnit Task	CppUnit	C++	NO	Available on Bitbucket	 BAM-7839 - Support the CppUnit task plugin ( Open)

JUnit Parser

This page describes how to configure a Bamboo task to parse [JUnit](#) test results.

Because TestNG uses the JUnit XML format, the JUnit Parser task is also able to parse TestNG test results.

Before you begin:

- Java builder tasks in Bamboo (e.g. Maven) parse test information as part of the task. You do not need to configure a test task, if you have specified that test results will be produced as part of the builder task.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Configuring a test task](#)

Atlassian blogs:

- [So you want to run tests in parallel... now what?](#)

To configure a JUnit Parser task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing JUnit Parser task, or click **Add Task** and then **JUnit Parser** to create a new task.
3. Update the task settings:

Task Description	Enter a description of the task, for display in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
Specify custom results directories	Enter the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as <code>**/test-reports/*.xml</code> where the base directory is the "working directory" – this can be found at the start of your build log. Do not specify an absolute path. For jobs that use CVS, the job build's root directory is <code><bamboo-home>/xml-data/build-dir/JOB_KEY/<cvsmodule></code> .

4. Click **Save**.

JUnit Parser Configuration

Task Description

☐ Disable this task

Specify custom results directories

Where does the build place generated test results?

This is a comma separated list of test result directories. You can also use ant style patterns such as `*/test-reports/*.xml`

MBUnit Parser

This page describes how to configure a Bamboo task to parse [MBUnit](#) test results.

Before you begin:

- .NET builder tasks in Bamboo (e.g. NAnt) do not parse test information as part of the task. You must configure a test task (e.g. MBUnit Parser), if you want test results from the builder task to be parsed.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Configuring a test task](#)

To configure a MBUnit Parser task:

1. Navigate to the **Tasks** configuration tab (this will be the default job if creating a new plan).
2. Click the name of an existing MBUnit Parser task, or click **Add Task** and then **MBUnit Parser** to create a new task.
3. Update the task settings:

Task Description	Enter a description of the task, for display in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
MBUnit Test Results File	Enter the name of the test results file. The test file must be in MBUnit XML format. For more information on MBUnit, see http://www.mbunit.com/ .

4. Click **Save**.

MSTest Parser

This page describes how to configure a Bamboo task to parse [MSTest](#) results.

.NET builder tasks in Bamboo (for example NAnt) do not parse test information as part of the task. To have the test results parsed, you need to configure a test task such as MSTest Parser.

i Note that each test results file must have a unique name. You can use Bamboo [variables](#) to achieve this. Here is a customer-supplied example that includes the revision and build numbers in the name of the test file:

```
<Project-Test-Subfolder>\TestResults\<Project>TestResults-Rev_${bamboo.repository.revision.number}-Build_${bamboo.buildNumber}.trx
```

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Configuring a test task](#)

To configure a MSTest Parser task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing MSTest Parser task, or click **Add Task** and then **MSTest Parser** to create a new task.
3. Update the task settings:

Task Description	Enter a description of the task, for display in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
MSTest Test Results File	Enter the name of the test results file. The test file must be in MSTest format. For more information on MSTest, see this MSDN page .

4. Click **Save**.

MSTest Parser Configuration

Task Description

☐ Disable this task

MSTest Test Results File/Directory*

The test files must be in MSTest format.

MSTest Runner

This page describes how to configure a Bamboo [MSTest Runner](#) task. The MSTest Runner task runs and parses tests for .NET builds.

Before you begin:

- .NET builder tasks in Bamboo (e.g. NAnt) do not parse test information as part of the task. You must configure a test task (e.g. MSTest Parser), if you want test results from the builder task to be parsed.
- If Bamboo is running as a Windows service, ensure that the Service is running as a [local user](#) instead of a System User (Bamboo will install itself as the SYSTEM user on Windows).

To configure a Bamboo MSTest Runner task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default Job if creating a new plan).
2. Click the name of an existing MSTest Runner task, or click **Add Task** and then **MSTest Runner** to create a new task.
3. Update the task settings:

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Configuring a test task](#)

Task Description	A description of the task, for display in Bamboo.
Disable this task	Check, or clear, to selectively run this task.

Executable	The MSTest Runner executable that you wish to use for this task (e.g. "Visual Studio 2010"). The executable that you select will become one of the task's capability requirements (and hence, one of the job's requirements). For details, please see Configuring a job's requirements.  Specifically for MSTest, we recommend that the executable be defined with the Visual Studio IDE folder path. Example: <pre data-bbox="927 405 1388 568">C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\</pre> This will allow Bamboo to find the necessary resources.
Environment Variables	Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m".
Container	The test container, i.e. the file that contains the tests you want to run. For example, tests.dll. The value of this field is passed to the MSTest.exe as the /testcontainer parameter. See MSTest.exe Command-Line Options (MSDN) .
Test Metadata	The path to the Test Metadata file relative to the working directory. For example, "MyApp\MyApp.vsmdi"
Result Filename	The file that you want to save the test results to. For example, testResults.trx. The value of this field is passed to the MSTest.exe as the /resultsfile parameter. See MSTest.exe Command-Line Options (MSDN) .
Run Configuration	The run configuration that you want to use. For example, localtestrun.Testrunconfig. The value of this field is passed to the MSTest.exe as the /runconfig parameter. See MSTest.exe Command-Line Options (MSDN) .

MSTest Runner Configuration

Task Description

☐ Disable this task

Executable*

Visual Studio 2010

⌵

Add New Executable

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters separated by a space.

Container*

The file that contains the tests. For example, "MyTests\bin\Debug\MYTests.dll"

Test Metadata

Path to the Test Metadata file relative to the working directory. For example, "MyApp\MyApp.vsmdi"

Result Filename*

The name Bamboo should give to the results file produced by MSTest. Must end with the extension '.trx'

Run Configuration

Use this option to specify a run configuration file

Save

Cancel

NUnit Parser

This page describes how to configure a Bamboo [NUnit Parser](#) task.

Before you begin:

- .NET builder tasks in Bamboo (e.g. NAnt) do not parse test information as part of the task. You must configure a test task (e.g. MSTest Parser, NUnit Parser), if you want test results from the builder task to be parsed.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Configuring a test task](#)

To configure a NUnit Parser task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing NUnit Parser task, or click **Add Task** and then **NUnit Parser** to create a new task.
3. Update the task settings:

Task Description	Enter a description of the task, for display in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
NUnit Test Results File/Directory	Enter the name of the test results file/directory. The test files must be in NUnit XML format. For more information on NUnit, see http://www.nunit.org/ .

4. Click **Save**.

NUnit Parser Configuration

Task Description

☐ Disable this task

NUnit Test Results File/Directory*

**/test-reports/*.xml

The test files must be in NUnit XML format. For more information on NUnit visit: [NUnit.org](http://www.nunit.org)

Save

Cancel

NUnit Runner

This page describes how to configure a Bamboo task to run [NUnit](http://www.nunit.org) tests, and then parse the test results.

Before you begin:

- .NET builder tasks in Bamboo (e.g. NAnt) do not parse test information as part of the task. You must configure a test task (e.g. MSTest Parser, NUnit Parser), if you want test results from the builder task to be parsed.

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Configuring a test task](#)

To configure a NUnit Runner task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing NUnit Runner task, or click **Add Task** and then **NUnit Runner** to create a new task.
3. Update the task settings:

Task Description	A description of the task, which gets displayed in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
Executable	The NUnit Runner executable that is available to perform the task. The executable that you select will become one of the task's (and so, the job's) requirements. You can add other executables , if required.
NUnit Test Files	The name of an assembly (.dll), Visual Studio project (.csproj), or NUnit Test Suite (.nunit) to test. See http://www.nunit.org/ .
Result Filename	The name to be used for the XML results file.
Tests to Run	The name of the test case, test fixture or namespace to run.
Test Categories to Include	Specify one or more test categories, separated by commas, to be included in the test run.
Test Categories to Exclude	Specify one or more test categories, separated by commas, to be excluded from the test run. Exclusions take precedence over inclusions.
Command Line Options	Specify any command line options or switches you wish to include when running NUnit.
Environment Variables	Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m".

4. Click **Save**.

For more information on NUnit, see <http://www.nunit.org/>.

NUnit Runner Configuration

Task Description

☐ Disable this task

Executable*

 [Add New Executable](#)

NUnit Test Files*

Specify an assembly (.dll), Visual Studio project (.csproj), or NUnit Test Suite (.nunit) to test

Result Filename*

The name Bamboo should give to the results file produced by NUnit. This is an XML file.

Tests to Run

Specify the full name of the test to run. The name of the test may be that of a test case, test fixture or namespace. Specify multiple tests by separating names with commas (without spaces).

Test Categories to Include

Specify one or more test categories, separated by commas, to be included in the test run.

Test Categories to Exclude

Specify one or more test categories, separated by commas, to be excluded from the test run. Exclusions take precedence over inclusions.

Command Line Options

Add any command line options or switches you wish to include when running NUnit

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters separated by a space.

[Save](#) [Cancel](#)

PHPUnit

This page describes how to configure a [PHPUnit](#) task.

Before you begin:

- To use this task, you will need to [install PHPUnit](#) and reference the path to your PHP command-line interpreter, (e.g. /usr/bin/phpunit on Ubuntu).

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)

To configure a PHPUnit task:

- Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
- Click the name of an existing task, or click **Add Task** and then **PHPUnit** (or another option, such as **PHPUnit 3.3.X**) to create a new task.
- Update the task settings:

Task Description	Enter a description of the task, for display in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
Executable	Select the PHPUnit executable that you wish to configure for this task (e.g. "PHPUnit 3.3.x" or "PHPUnit"). The executable that you select will become one of the task's capability requirements (and hence, one of the job's requirements). For details, please see Configuring a job's requirements .
Arguments	Type the name of the directory/files that will be analysed recursively by PHPUnit. The default value is "." (i.e. the working subdirectory, if specified). You must specify at least one argument.
Environment Variables	<i>(Optional)</i> Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see Using global, plan or build-specific variables). Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g. ANT_OPTS="-Xms200m -Xmx700m").
Working Sub Directory	<i>(Optional)</i> An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.
Log test execution to XML file	Select if you want PHPUnit to record test results in JUnit format.  This format is also used by TestNG. Test Result File — the relative location, and name, of the file to record PHPUnit test results.
Generate code coverage report in HTML format	Select if you want PHPUnit to generate code coverage data in HTML format (e.g. for PHPUnit HTML Code Coverage reports). HTML Code Coverage Directory — the relative location of the directory to store the code coverage report.

4. Click **Save**.

PHPUnit Configuration

Task Description

☐ Disable this task

Executable

PHPUnit 3/4 [Add New Executable](#)

Arguments*

Arguments passed to the PHPUnit executable each time this Job executes.

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters separated by a space.

Working Sub Directory

(Optional) Specify an alternative sub-directory as working directory for the task.

Where should PHPUnit store the test results file?

☒ Log test execution to an XML file

Test Results File

test-reports/phpunit.xml

A relative path to the file where PHPUnit should store the log of test execution in JUnit XML format (option --log-junit).

Where should PHPUnit store HTML code coverage data?

☒ Generate code coverage report in HTML format

HTML Code Coverage Directory

test-reports/coverage/html

A relative path to the directory where PHPUnit should store the code coverage report in HTML format (option --coverage-html).

[Save](#) [Cancel](#)

TestNG

This page describes how to configure a Bamboo task to parse [TestNG](#) test results.

Before you begin:

- Java builder tasks in Bamboo (e.g. Maven) parse test information as part of the task. You do not need to configure a test task, if you have specified that test results will be produced as part of the builder task.

To configure a TestNG Parser task:

Related pages:

- [Configuring tasks](#)
- [Configuring jobs](#)
- [Configuring a test task](#)

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing TestNG task, or click **Add Task** and then **TestNG** to create a new task.
3. Update the task settings:

Task Description

Enter a description of the task, for display in Bamboo.

Disable this task	Check, or clear, to selectively run this task.
Specify custom results directories	Enter the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as <code>*/test-reports/</code> . Please specify file path relative to your job build's root directory. Do not specify an absolute path. For jobs that use CVS, the job build's root directory is <code><bamboo-home>/xml-data/build-dir/JOB_KEY/<cvsmodule></code> .

4. Click **Save**.

TestNG Parser Configuration

Task Description

☐ Disable this task

Specify custom results directories

Where does the build place generated test results?

This is a comma separated list of test result directories. You can also use ant style patterns such as `**/test-reports/*.xml`

Save **Cancel**

Configuring a deployment task

Deployment tasks in Bamboo allow you to set up plans that can manage the continuous deployment and delivery of your application.

See the following pages for more information on configuring specific deployment tasks in Bamboo:

- [Using Tomcat with Bamboo for continuous deployment](#)
- [Using the SCP task in Bamboo](#)
- [Using the SSH task in Bamboo](#)
- [Using the Heroku task in Bamboo](#)

Using Tomcat with Bamboo for continuous deployment

You can use Bamboo to deploy and manage your Java web application with Tomcat 6 or 7, without having to directly interact with Maven, Ant or write special scripts.

Bamboo provides tasks that use the HTTP-based scripting interface to the Tomcat Manager application that ships with Tomcat. You can use the Bamboo tasks to perform the following Tomcat operations:

- Deploy an application to a Tomcat instance
- Start an application in a Tomcat instance
- Stop an application in a Tomcat instance
- Reload an application to a Tomcat instance
- Undeploy an application from a Tomcat instance

Each of these tasks run as part of a Bamboo job.

On this page :

- [Setting up Tomcat](#)
- [Deploying an application from Bamboo](#)
 - [Configuring the Tomcat tasks](#)

Related pages:

- [Configuring a deployment task](#)

Atlassian blogs:

- [Continuous deployment with Bamboo and Tomcat](#)

Setting up Tomcat

You will need to prepare the Tomcat server before Bamboo can manage and deploy applications to it.

1. Download the [Tomcat 7 distribution](#) and unzip it on your file system.
2. Add a new Tomcat user for Bamboo to use the [Tomcat Application Manager](#) by adding the following line in `conf/tomcat-users.xml` between the `<tomcat-users>` tags:

```
<user username="bamboo" password="bamboo" roles="manager-script,manager-gui"/>
```

3. Start Tomcat by running `bin/startup.sh` on Linux or Mac, or `bin/startup.bat` on Windows.
4. Test this setup by browsing to <http://localhost:8080/manager> and using the username and password you configured in the step above. You should see the "Tomcat Web Application Manager" page, and a list of the running applications on your instance.

For more information about the Tomcat Application Manager and its authentication and authorisation configuration see the [Tomcat documentation](#).

Deploying an application from Bamboo

You use Tomcat deployment tasks in the context of a job in a build plan in Bamboo. This plan should generate a deployable artifact, such as a WAR file. To deploy the artifact, you add a Tomcat deploy task to the plan, as follows:

1. Navigate to the task configuration for the job (this will be the default job if you are creating a new plan).
2. Click **Add Task** and then **Deploy Tomcat Application**.
3. Configure the Tomcat task settings, as described [below](#).
4. Click **Save**.
5. To deploy the application, simply run the plan.

You can check that the deployment has been successful by:

1. Navigating to the logs for the job. Towards the end you should see something like:

```
> Deploying application with war file 'target/tomcat-test-0.1.war' to context  
'/myapp' to server  
[http://localhost:8080/manager/](http://localhost:8080/manager/)  
> Application was successfully deployed.
```

This indicates that Bamboo completed the task successfully.

2. Now, browse to the expected address for your application. You should see the welcome page.

Configuring the Tomcat tasks

The Tomcat Deploy, Start, Stop, Undeploy and Reload tasks each make use of some or all of the following configuration settings:

Task Description	To help you to identify the task.
Disable this task	Check, or clear, to selectively run this task.
Tomcat Manager URL	The URL for the Tomcat Manager e.g. http://localhost:8080/manager/
Target Tomcat server is version 6.x	Choose this if deploying to a Tomcat 6.x server.
Tomcat Manager Username and Password	These should match the credentials set in <code>conf/tomcat-users.xml</code> when you configured Tomcat, as described above .
Application Context	Specifies where the application should sit on the Tomcat server once deployed.

WAR File	The path to the WAR file, relative to the Bamboo working directory, for example "target/tomcat-test-0.1.war"
Deployment Tag	The value used to tag the deployment within the Tomcat Manager. You can use Bamboo variables to build the tag value. For example, using the value \${bamboo.buildResultKey} will tag the deployment with the build number of the build that was used to deploy the application.

Deploy Tomcat Application Configuration

Task Description

☐ Disable this task

Tomcat Manager URL*

The URL to the Tomcat Application Server Manager eg "http://localhost:8080/manager"

☐ Target Tomcat server is version 6.x

The Manager application has been re-structured for Tomcat 7 onwards and some of the URLs have changed.

Tomcat Manager Username*

An authorized username for the Tomcat Application Server Manager

Tomcat Manager Password*

An authorized password for the Tomcat Application Server Manager

Application Context*

The Application Context to deploy the application to eg '/mywebapp'

WAR File*

The path of the WAR to deploy relative to the working directory

Deployment Tag

The value used to tag the deployment within the Tomcat Application Server Manager. You can use Bamboo Variables to build your own tag value

Save

Cancel

Using the SCP task in Bamboo

You can use the Bamboo SCP task to upload files from Bamboo directly to a remote server as part of a Bamboo job. The SCP task is able to copy multiple files and preserves the directory structure for the copied files.

See [Configuring a deployment task](#) for an overview of Bamboo deployment tasks.

Related pages:

- [Configuring a deployment task](#)
- [Using the SSH task in Bamboo](#)

To configure an SCP task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing SCP task, or click **Add Task** and then **SCP Task** to create a new task.
3. Complete the following settings:

Task Description	Helps you identify the purpose of the task.
Disable this task	Check, or clear, to selectively run this task.
Host	The hostname or IP address of the remote server to which the files will be copied.
Verify remote host fingerprint on connect	Enter the host fingerprint to be verified. See below for more details.
Port	The port number of the remote host that is used for the SSH connection. The default value is 22.
Username	The username to use to connect to the remote host.
Authentication Type	Password – the password associated with Username .
	Key without passphrase – browse to the SSH private key with which to authenticate with the remote host.
	Key with passphrase – browse to the SSH private key, and supply the passphrase, to use to authenticate with the remote host.
Local Path	The local path (relative to the Bamboo working directory) to the files you want to copy. Use commas to separate files and directories. You can also use Ant-style pattern matching to include multiple files, such as <code>**/target/*.jar</code> .
Remote Path	The path to the destination directory on the remote server.

4. Click **Save**.

Host fingerprint

You can determine the fingerprint for a host by running:

```
ssh-keygen -l -F <HOSTNAME>
```

The fingerprint is the part of the response shown in the screenshot below:

```
jdumay — bash — 88x13
north:~ jdumay$ ssh-keygen -l -F heck.dyn.syd.atlassian.com
# Host heck.dyn.syd.atlassian.com found: line 20 type RSA
2048 b9:81:c6:9f:4f:fc:9a:32:32:7c:8e:12:6c:9a:77:df heck.dyn.syd.atlassian.com (RSA)
north:~ jdumay$
```

SCP Task Config

Task Description

☐ Disable this task

Host*

Hostname or IP address of

☐ Verify remote host

Port

Port number for remote host
value is 22.

Username*

Username you want to use

Authentication Type*

Password*

Local Path*

Location of files to copy to
This is a comma separated

☐ Use Ant patterns to
Use Ant patterns to select
reference.

Remote Path*

[Cancel](#)

You can use the Bamboo SSH task to execute a SSH command on a remote computer as part of a Bamboo job. You can use the SSH task to do such things as:

- Calling database migration scripts
- Starting and stopping services
- Anything you can run on the command line on a remote machine

See [Configuring a deployment task](#) for an overview of Bamboo deployment tasks.

Related pages:

- [Configuring a deployment task](#)
- [Using the SCP task in Bamboo](#)

To configure an SSH task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing SSH task, or click **Add Task** and then **SSH Task** to create a new task.
3. Complete the following settings:

Task Description	Helps you identify the purpose of the task.
Disable this task	Check, or clear, to selectively run this task.
Host	The hostname or IP address of the remote server on which the SSH Command will be executed.
Verify remote host fingerprint on connect	Enter the host fingerprint to be verified. See below for more details.
Port	The port number of the remote host that is used for the SSH connection. The default value is 22.
Username	The username to use to connect to the remote host.
Authentication Type	Password – the password associated with Username .
	Key without passphrase – browse to the SSH private key with which to authenticate with the remote host.
	Key with passphrase – browse to the SSH private key, and supply the passphrase, to use to authenticate with the remote host.
SSH Command	The shell command to execute on the remote host. You can only enter a single command here.

4. Click **Save**.

Host fingerprint

You can determine the fingerprint for a host by running:

```
ssh-keygen -l -F <HOSTNAME>
```

The fingerprint is the part of the response shown in the screenshot below:


```
jdumay — bash — 88x13
north:~ jdumay$ ssh-keygen -l -F heck.dyn.syd.atlassian.com
# Host heck.dyn.syd.atlassian.com found: line 20 type RSA
2048 b9:81:c6:9f:4f:fc:9a:32:32:7c:8e:12:6c:9a:77:df heck.dyn.syd.atlassian.com (RSA)
north:~ jdumay$
```

SSH Task Config

Task Description

☐ Disable this task

Host*

Hostname or IP address

☐ Verify remote host

Port

22

Port number for remote host
value is 22.

Username*

Username you want to use

Authentication Type*

Password

Password*

SSH command*

Shell command to execute

Save Cancel

Pattern matching reference

Bamboo supports a powerful type of regular expression for matching files and directories (as with pattern matching in Apache Ant). These expressions use the following wild cards:

?	Matches one character (any character except path separators)
*	Matches zero or more characters (not including path separators)

**	Matches zero or more <i>path segments</i> .

Remember that Ant globs match *paths*, not just simple filenames.

- If the pattern does not start with a path separator i.e. / or \, then the pattern is considered to start with `/**/`.
- If the pattern ends with / then `**` is automatically appended.
- A pattern can contain any number of wild cards.

Also see the [Ant documentation](#).

Examples

*.txt	Matches /foo.txt and /bar/foo.txt but not /foo.txtty or /bar/foo.txtty/
/*.txt	Matches /foo.txt but not /bar/foo.txt
dir1/file.txt	Matches /dir1/file.txt, /dir3/dir1/file.txt and /dir3/ /dir2/dir1/file.txt
**/dir1/file.txt	Same as above.
/**/dir1/file.txt	Same as above.
/dir3/**/dir1/file.txt	Matches /dir3/dir1/file.txt and /dir3/dir2/dir1/file .txt but not /dir3/file.txt, /dir1/file.txt
/dir1/**	Matches all files under /dir1/

Sharing artifacts

This page describes how to share artifacts. Bamboo allows artifact sharing between:

- Jobs
- Build plans
- Build plans to deployment environments.

Sharing artifacts between jobs

You can share artifacts between jobs in different stages using artifact dependencies. For example, you may want to run acceptance tests on a build, sharing the same WAR from one job to another without rebuilding it each time.

Each time the artifact is shared with a subsequent job, it is copied to the job's agent.

To share an artifact between two jobs in different stages:

1. Navigate to the configuration pages for the job that will produce the artifact, as described on [Configuring jobs](#), and click the **Artifacts** tab (see [Configuring a job's build artifacts](#)).
2. Click **Edit** for the artifact that you want to share. The artifact definition will be displayed.
3. Select the **Shared** check box and then click **Save**.
4. Navigate to the job in a subsequent stage that will consume the artifact, and click the **Artifacts** tab. Click **Create Dependency**. Complete the fields on the screen (see [screenshot](#) below) and click **Create**.

On this page:

- [Sharing artifacts between jobs](#)
- [Sharing artifacts between build plans](#)
- [Sharing artifacts from a build plan to a deployment environment](#)

Related pages:

- [Viewing a build's artifacts](#)
- [Configuring a job's build artifacts](#)
- [Pattern matching reference](#)

Atlassian Blogs:

- [Boost Your Build Automation with Artifact Sharing](#)



1. The **Artifact** list only shows artifacts from jobs in previous stages that have been marked as shared. This is described in [Configuring a job's build artifacts](#).
2. **Destination directory** is relative to the build directory. Do not use the absolute path to refer to the destination directory.

Screenshot: Creating an artifact dependency

Create Dependency

This will create new Artifact Dependency for this Job

Artifact* 

Name of the Shared Artifact required by this Job.

Destination directory

The directory where the Artifact will be available for your build, relative to the Build's working directory.

Sharing artifacts between build plans

You can share artifacts between different build plans, however you need to use the 'Artifact downloader task' to do so. For example, you may want to run acceptance tests on a particular build from a different plan by sharing the same WAR from one plan to another without rebuilding it each time.

To share an artifact between two build plans:

1. Locate the build plan that you wish to associate an artifact with. Select **Configure plan** from the 'Actions' drop down menu. The plan configuration window will display:

 Build projects / Z Personal Branches / PW test

Configuration

Plan Configuration

Stages & jobs 1

Branches 2

default

default 1

Plan details

Stages

Source repositories

Triggers

Branches

Dependencies

Permissions

Notifications

Variables

Miscellaneous

Audit log

Plan details

You can update the names and description of the current project and plan.

Project name*

Plan name*

Description

☒ Plan enabled

2. In the left pane, click on **Stages & jobs** and select a job or create a new job if one does not already exist. Click on the **Tasks** tab for the selected job. The 'Tasks' pane will appear
3. Click the **Add task** button. The 'Task types' window will appear. Select **Artifact Downloader Task** to open the 'Artifact downloader task' configuration pane:

Source Code Checkout

Checkout Default Repository

Artifact Downloader Task

Final tasks Are always executed at the end of the build

Drag tasks here to make them final

Add task

Artifact Downloader Task configuration

Task description

Get Artifact

☐ Disable this task

Source Plan*

+core+ Bamboo - CI Tests

Source Plan where artifact is produced

Artifact Name*

All artifacts

Path

+ Add another artifact

Save

Cancel

Complete the configuration using the following options:

Field	Description	Optional?
Task description	A brief description of the artifact downloader task	✓
Disable this task	Check this box to disable the task	-
Source Plan	The build plan that is the source of the artifact you need to download	✗
Artifact Name	Use the drop down menu to locate the name of the artifact that you want to download	✗
Path	The location of the working directory into which you want the artifact downloaded	✓

- Click on **Add another artifact** to add another artifact to the download list. Alternatively, use the grey cross icon to delete an artifact from your configuration
- Click on **Save** to save your artifact download configuration.



- The **Artifact** drop down menu only shows artifacts from jobs in previous stages that have been marked as shared. This is described in [Configuring a job's build artifacts](#)
- Destination directory** is relative to the build directory. Do not use the absolute path to refer to the destination director

Sharing artifacts from a build plan to a deployment environment

You can also share artifacts from a build plan into a deployment environment. For example, you may wish to share a particular build result from a plan with a deployment environment. To do this, you need to add the 'Artifact downloader task' to a deployment environment during or after the environment creation process.

To share an artifact from a build plan to a deployment environment:

- Open your deployment project and expand the relevant environment panel. In the Other settings section, click on the **Tasks** button. The 'Set up tasks' screen will display:

Setup tasks for Regula 1

How deployment tasks work

What are all the tasks that need to happen to make this deployment a success?

29 agents and 21 elastic images have the capabilities to run this Deployment

Script

Final tasks Are always executed at the end of the build

Drag tasks here to make them final

Add task

Back to deployment project

No task selected

Select a task from the list on the left to configure it.

- Click on the **Add task** button. The 'Task type' selection window will display. Select **Artifact Downloader Task** to open the 'Artifact downloader task configuration' pane:

Setup tasks for Regula 1

How deployment tasks work

What are all the tasks that need to happen to make this deployment a success?

29 agents and 21 elastic images have the capabilities to run this Deployment

Script

Artifact Downloader Task

Final tasks Are always executed at the end of the build

Drag tasks here to make them final

Add task

Back to deployment project

Artifact Downloader Task configuration

Task description

☐ Disable this task

Artifact Name*

All artifacts

Path

[Add another artifact](#)

Save

Cancel

Complete the configuration using the following options:

Field	Description	Optional?
Task description	A brief description of the artifact downloader task	✓
Disable this task	Check this box to disable the task	-
Source Plan	The plan that is the source of the artifact you need downloading	✗
Artifact Name	Use the drop down menu to locate the name of the artifact that you want to download	✗
Path	The location of the working directory into which you want the artifact downloaded	✓

- Click on **Add another artifact** to add another artifact to the download list.
- Click on **Save** to save your artifact download configuration.



- The **Artifact** drop down menu only shows artifacts from jobs in previous stages that have been marked as shared. This is described in [Configuring a job's build artifacts](#)
- Destination directory** is relative to the build directory. Do not use the absolute path to refer to the destination director

Working with builds

The following pages contain information on working with your Bamboo [builds](#). If you are looking for information on configuring a [job](#), please see [Jobs and tasks](#).

- [Working with build results](#)
- [Working with comments](#)
- [Working with labels](#)
- [Quarantining failing tests](#)
- [Setting up plan build dependencies](#)
- [Viewing test statistics for a job](#)
- [Reordering jobs in the build queue](#)
- [Stopping an active build](#)

Working with build results

About builds

A build is the execution of either a [plan](#) or a [job](#). The execution of a plan is referred to as a 'plan build' and that of a job is a 'job build'.

Related pages:

- [Viewing a build result](#)
- [Deleting the results of a plan build](#)
- [Working with comments](#)
- [Working with labels](#)
- [Assigning responsibility for build failures](#)
- [Configuring build results expiry for a plan](#)

About build results

Every completed build has a *build result*:

- 'Successful' — the code compiled, with or without errors, and all tests completed successfully.
- 'Failed' — either the code did not compile, or at least one test failed.
- 'Incomplete' — the build was not completed, e.g. it may have been stopped manually.

Additionally,

- if the build result is 'Failed', and the previous build result was 'Successful', the build is said to be 'Broken'.
- if the build result is 'Successful', and the previous build result was 'Failed', the build is said to be 'Fixed'.

The latest build result for every plan is listed on the [Dashboard](#). Bamboo can also send [notifications](#) and generate [RSS feeds](#) about build results.

Viewing a build result

The instructions on this page describe how to view the build results for a plan.

Every completed build has a *build result*:

- 'Successful' — the code compiled, with or without errors, and all tests completed successfully.
- 'Failed' — either the code did not compile, or at least one test failed.
- 'Incomplete' — the build was not completed, e.g. it may have been stopped manually.

Additionally,

- if the build result is 'Failed', and the previous build result was 'Successful', the build is said to be 'Broken'.
- if the build result is 'Successful', and the previous build result was 'Failed', the build is said to be 'Fixed'.

Viewing the most recent build result for a plan

To view the most recent job build result of a plan:

1. Click **Dashboard** in the top menu.
2. Locate the plan on the **All Plans** tab, then click the build number.

On this page:

- [Viewing the most recent build result for a plan](#)
- [Viewing all build results for a plan](#)
- [Viewing all build results for a job](#)

Related pages:

- [Viewing test results for a build](#)
- [Viewing the code changes that triggered a build](#)
- [Viewing a build's artifacts](#)
- [Viewing a build log](#)
- [Viewing the metadata for a build result](#)
- [Viewing linked JIRA issues](#)
- [Reporting](#)

Screenshot: *Build Result Summary*

Status ribbon
Did the build succeed or fail?

Details
Scan details of the build easily.

Stages and jobs in the plan
Scan the success of job builds. Click an icon to see details.

Test summary
Quickly see how many tests are failing, and how many were fixed in this build.

Shared artifacts
See the artifacts shared from this build. Download them with a single click.

History
Scan the status of recent builds

Who is responsible?
Users who commit code are automatically assigned.

Code changes
See the code changes associated with this build.

JIRA issues
See the JIRA issues related to this build. Click through to the issue in JIRA for details.

Tab	Description
Build Summary	Displays a snapshot of the build result. ✓ indicates a successful build. ⏸ indicates a build that was not completed. For example, it may have been stopped manually. ❗ indicates a failed build. If a build has failed, you can run the entire build again or rerun just the failed stage.
Tests	Provides details of the build's test results .
Changes	Provides details of the code changes that triggered this build (if applicable).
Artifacts	Shows any artifacts relating to this build.
Logs	Displays a complete build log .
Metadata	Displays any metadata that relates to this build.
Build Times	Displays a histogram of build times for jobs, and a list of which agents were used to build each job.
Issues	Provides details of the JIRA issues linked to this build (if applicable). Availability depends on Bamboo's configuration.
Clover	Displays the Clover code-coverage that relates to this build (if applicable). The clover tab is located on the job level because a build can have more than one jobs, and each job might have different Clover results or not have clover tab at all. That's why in order to see the Clover tab, you need to drill down to the individual job that contains the clover report.

- You can click the **Comment** button to [add a comment](#).

- You can assign responsibility for a broken build, either to yourself (click **Claim full responsibility**) or to someone else in your team (click **Assign responsibility**).

Viewing all build results for a plan

To view all build results for a plan:

1. Click **Dashboard** in the top menu.
2. Click the plan on the **All Plans** tab. The ten most recent builds will be displayed in the 'Recent History' section on the **Plan Summary** tab. See [Viewing a plan's build information](#).
3. Click the **History** tab to view all builds for the plan.

Viewing all build results for a job

To view all build results for a job:

1. Navigate to the desired job, as described on [Configuring jobs](#). The ten most recent builds will be displayed in the 'Recent History' section of the **Job Summary** tab.
2. Click the **History** tab to view all builds for the job.

Viewing test results for a build

Bamboo provides a convenient summary of all the tests that were run when a particular build was executed — as well as full details of any errors. This is useful when you are investigating what caused a build to fail.

Note that for more meaningful display of test names within Bamboo, the word 'test' is stripped out of test case name names if it occurs at the beginning, and capitals and underscores are treated as word separators.

Related pages:

- [Viewing a test's history](#)

To view the test results for a particular build:

1. Navigate to the build results for the plan or job, as described in [Viewing a build result](#), and click the desired build result.
2. Click the **Tests** tab.
 - Click the test name to see a particular test's results for [other builds](#).

Screenshot: Test results for a build

Summary

Tests

Changes

Artifacts

Logs

Metadata

Issues

Actions

Test Results

56 tests in total

11 tests failed

11 failures are new

25 minutes taken in total.

New Test Failures (11)

Expand All

Collapse All

Test	View Job	Duration
BambooRegressionBugTesttestCvsDeleteBuildFiles	Func Test Repository	1 min
BuildSetupTesttestCanCreatePlanWithJobOfSameName	Func Test Misc	21 secs
BuildSetupTesttestSvnCloneBuild	Func Test Misc	29 secs
BuildSetupTesttestBasicChainValidationAndCreation	Func Test Misc	28 secs
BuildTesttestCvsSuccessfulBuild <pre>junit.framework.AssertionFailedError: Expected text not found in current page: [Random P junit.framework.AssertionFailedError: Expected text not found in current page: [Random P Page content was: [Random Project96753083 - Plan96753083 - Default Job: Build Configurat at junit.framework.Assert.fail(Assert.java:47) at net.sourceforge.jwebunit.junit.WebTester.assertTextPresent(WebTester.java:329) at net.sourceforge.jwebunit.junit.WebTestCase.assertTextPresent(WebTestCase.java: at com.atlassian.bamboo.acceptance.tests.AbstractBuildTest._verifyBuildDetails(Ab at com.atlassian.bamboo.acceptance.tests.BuildTest.testCvsSuccessfulBuild(BuildTe</pre>	Func Test Repository	1 min
BuildTesttestPerforceSuccessfulBuild	Func Test Repository	40 secs
BuildTesttestCvsScriptBuild	Func Test Repository	10 secs
BuildTesttestPerforceFailedBuild	Func Test Repository	27 secs

Viewing a test's history

A test's *history* shows you:

- The occasions when the test has failed. This can be useful when investigating what code changes were related to a failed test (see below).
- The test's average *duration* (running time), and whether the duration is increasing or decreasing across builds.

Related pages:

- [Viewing test results for a build](#)

To view a test's history:

- Navigate to the build results for the Plan/Job, as described in [Viewing a build result](#), and click the desired build result.
- Click the **Tests** tab.
- Click the name of the test in which you are interested. The test's latest result will be displayed.
- Click **View test case across builds**. The 'Test History' will be displayed, as shown below.

Screenshot: Test History

[Job Summary](#)
[Recent Failures](#)
[History](#)
[Tests](#)
[Logs](#)
Showing Last 25 builds ▾
Actions ▾

Test Summary

This page summarises information for **testCvsDeleteBuildFiles** in test class **BambooRegressionBugTest**.

Details

Test Class **com.atlassian.bamboo.acceptance.te**

Test **Cvs delete build files**

Last ran  **#43** (Scheduled build – 11 hours ago)

First ran **#4**

Failures **1**

Successes **25**

Passing since **#31** (Scheduled build – 2 weeks ago)

Test Statistics

96%
Successful

Successful Jobs: 25 / 26

Average Duration: 88 seconds

Recent Failures

Failed In	Fixed In	Time Taken To Fix
 #30 Scheduled build 2 weeks ago	 #31 Scheduled build 2 weeks ago	1 build 23 hours, 41 minutes

The following failure information is for last 25 builds:

- Average time to fix test when failed: **23 hours, 41 minutes**
- Average number of builds between fixes: **1 build(s)**

Viewing the code changes that triggered a build

If a build was triggered by a code change, the updated files will be listed in the build result.

When Atlassian's **FishEye** is connected to your Bamboo server, you can view the code changes that triggered a build. When a build fails due to a compilation error or failed test, you can explore the failed build in FishEye and jump directly into the changeset that broke the build. You can view the history of that changeset to see what the author was trying to fix, take advantage of the side-by-side diff view to analyze the change and then open the correct files in your IDE.

Related pages:

- [Integrating Bamboo with FishEye](#)
- [Specifying the source repository](#)
- [Triggering builds](#)

To view the code changes that triggered a particular build result:

- Navigate to the build results for the plan, as described in [Viewing a build result](#), and click the desired build result.
- Click the **Changes** tab. A list of updated files will be shown.
 - Click the link to the source file to view the changes.
 - Click the version number to view the entire file.
 - Click the **diffs** link to view the differences between the current and previous version of each file.

Links to individual source-code files will only be available if your Bamboo administrator has connected the plan to the source repository, as specified in the 'Advanced Options' on the 'Source Repositories' tab for the plan. For details, please see [Integrating Bamboo with FishEye](#).

Viewing a build's artifacts

After a build has run, you can view the artifacts that were produced by all of the jobs in the plan. You can also view the latest version of an artifact from the most recent build.

Artifacts are files created by a *job build* (e.g. JAR files). Artifact definitions are used to specify which artifacts to keep from a build and are configured for individual *jobs*.

Viewing the artifacts for a build

To view a build's artifacts:

- Go to the build result. See [Viewing a build result](#) for instructions.
- Click the **Artifacts** tab. The artifacts produced by the jobs in the plan will be displayed. The [artifact definitions](#) for a job determine

which artifacts are kept and which artifacts are shared with other jobs in the plan.

- The artifacts that are marked as [shared](#) (in the artifact definitions) are listed under 'Shared Artifacts'.
- The artifacts that are not marked as [shared](#) (in the artifact definitions) are listed under 'Job Artifacts'.

On this page:

- [Viewing the artifacts for a build](#)
- [Viewing the latest version of an artifact from the latest build](#)

Related pages:

- [Configuring a job's build artifacts](#)
- [Configuring artifact sharing between jobs](#)

Viewing the latest version of an artifact from the latest build

To view the latest version of an artifact from the most recent build, you can manually edit the build artifact URL to retrieve it.

To view the latest version of an artifact from the most recent build:

1. Copy the URL for the build artifact.
2. Paste the URL for the build artifact in your browser and replace the build number in the URL with '/latest'.
 - If you need to log in to view the artifacts, you can append `os_username` and `os_password` parameters to the URL to access the files.

For example, if the URL for your artifact is:

`http://server/bamboo/browse/MYBUILD-254/artifact/logs/sample-log.log`

You would replace '-254' with /latest:

`http://server/bamboo/browse/MYBUILD/latest/artifact/logs/sample-log.log`

Screenshot: *Build Artifacts*

The screenshot shows the 'Artifacts' tab in the Bamboo web interface. At the top, there are tabs for Summary, Tests, Changes, Artifacts (selected), Logs, Metadata, Issues, and Sandbox. An 'Actions' dropdown menu is on the right. Below the tabs, the title 'Artifacts' is displayed, followed by the text 'The following artifacts have been generated by the jobs in this plan.' and a 'Remove Artifacts' button. The page is divided into two main sections: 'Shared Artifacts' and 'Job Artifacts'. Each section contains a table with columns for Job, Artifacts, Size, and a 'Remove' link. In the 'Shared Artifacts' table, there is one entry for 'Build WAR Package' with the artifact 'Bamboo WAR' (82 MB). In the 'Job Artifacts' table, there are three entries for functional tests, each with multiple artifacts like 'Cargo Log' and 'Output'.

Job	Artifacts	Size	
✓ Build WAR Package	Bamboo WAR	82 MB	Remove

Job	Artifacts	Size	
✓ Build WAR Package	Cargo Log	0 bytes	Remove
	Output	0 bytes	Remove
✓ Func Test Misc Functional Test	Output	0 bytes	Remove
	Cargo Log	486 bytes	Remove
✓ Func Test Repository Functional Test	Output	0 bytes	Remove
	Cargo Log	487 bytes	Remove
✓ Func Test SVN Functional Test	Output	0 bytes	Remove
	Cargo Log	486 bytes	Remove

Viewing a build log

Every *build* has a *build log*. A *build log* is a permanent record of all the output generated by compiling the *job's* source-code and executing the tests.

Related pages:

- [Working with build results](#)

To view a build log:

1. Navigate to the build results for the plan or job, as described in [Viewing a build result](#), and click the desired build result.
2. Click the **Logstab**.
 - Click **View** for the desired log.
 - Click **Download** to download a text file of the log.

Screenshot: *Build Log*

Summary

Tests

Changes

Artifacts

Logs

Metadata

Issues

Actions

Build Log

The Build generated 2,021 lines of output. [View live logs](#) | [Download full Build log.](#)

16-Feb-2011 00:02:19

Build ASD-CI-WAR-43 started building on agent Elastic Agent on i-f398e29f

16-Feb-2011 00:02:20

Running preRetrieveSourceCode task...

16-Feb-2011 00:02:20

Updating source code to revision: 142521

16-Feb-2011 00:02:20

Working directory '/mnt/bamboo-working-dir/ASD-CI-WAR' is empty. Checking out SVN URL 'https://svn.atlassian.com/svn/private/atlassian/bamboo/trunk'

16-Feb-2011 00:02:57

Updated source code to revision: 142521

16-Feb-2011 00:02:57

Running postRetrieveSourceCode task...

16-Feb-2011 00:02:57

Executing build ASD-CI-WAR-43

16-Feb-2011 00:02:57

Running pre-build action: Build Number Stamper

16-Feb-2011 00:02:57

Running pre-build action: VCS Version Collector

16-Feb-2011 00:02:57

Running pre-build action: Pre Command Agent Runner

16-Feb-2011 00:02:57

Running pre-build action: Sandbox Environment Setup

16-Feb-2011 00:02:57

Running pre-build action: Clover Grails PreBuild Action

16-Feb-2011 00:02:57

Running pre-build action: Repository Isolation Enabler Action

16-Feb-2011 00:02:57

Running pre-build action: Maven Settings Prebuild Action

16-Feb-2011 00:02:58

Building started with Maven2Builder

16-Feb-2011 00:02:59

Starting to build 'Artifact Sharing Dogfooding - CI Tests - Build WAR'

... running command line: /opt/maven-2.1/bin/mvn -Djava.io.tmpdir=/tmp/ASD-CI-WAR clean install -Dmaven.test.skip=true

... in : /mnt/bamboo-working-dir/ASD-CI-WAR

... using java.home: /opt/jdk-5

16-Feb-2011 00:03:01

[INFO] Scanning for projects...

16-Feb-2011 00:03:02

[INFO] Reactor build order:

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo Components

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo Agent Bootstrap

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo Remote Agent

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo Remote Agent

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo Agent Class Path

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo API

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo Test Utils

16-Feb-2011 00:03:02

[INFO] Atlassian Bamboo Core

Viewing the metadata for a build result

Related pages:

To view the metadata for a build result:

Screenshot: Metadata for a Build Result

Summary
Tests
Changes
Artifacts
Logs
Metadata
Issues
Actions ▼

Metadata

This build has the following metadata. These are property key value pairs describing the build. You can specify your own metadata in the build process via plugins.

Key	Value
buildKey	BAM-FC
buildNumber	4108
buildResultKey	BAM-FC-4108
custom.brmp.name	3.1-SNAPSHOT
custom.brmp.qualifier	
custom.brmp.rawname	3.1
custom.brmp.rawversion	3.1
custom.brmp.type	
custom.brmp.version	3.1-SNAPSHOT

Variables

No parameters have been manually overridden.

Assigning responsibility for build failures

Bamboo automatically alerts the people who are assigned as responsible for a broken build, and lets other members of the team know that someone is looking at the problem. As you investigate the build failure, you can revise who is responsible, or claim all the responsibility for yourself!

People are assigned as being responsible for fixing a broken build in two ways:

- When a build fails, Bamboo automatically assigns all those who committed code to the failing build as responsible.
- You can manually assign people as being responsible.

Bamboo then sends notifications to whoever is assigned. Once the build is successful, Bamboo removes the responsible people from the build – they're off the hook!

Note that notifications need to have been configured first, using the 'Change of Responsibilities' **Event** and the 'Responsible User' **Recipient Type**. See [Configuring notifications for a plan and its jobs](#) for more information.

Related pages:

- [Working with build results](#)
- [Working with comments](#)
- [Notifications](#)

To assign responsibility for a broken build manually:

1. Go to the [Build Result Summary](#) for a plan.
2. Click **Assign responsibility** to make another member of your team responsible for fixing the build.
3. Click **Claim full responsibility** if you want to shoulder all the blame yourself.

People who are responsible for the broken build are displayed on the [Build Result Summary](#).

Broken builds that are assigned to you are displayed on your **My Bamboo** page, available from the Dashboard.

Build Result Summary

Details

Trigger This build was manually triggered by [Sarah Goff-Dupont](#).

Revisions

Bitbucket API	880f20a...
Bitbucket Core	880f20a...
Bitbucket Web App	a7141b9...
Git Submodules	c7f09cc...
Repo	

Responsible

 [James Dumay](#) 

 [Jens Schumacher](#) Assigned by [James Dumay](#) 

 [Sarah Goff-Dupont](#) Assigned by [James Dumay](#) 

Assign responsibility

Claim full responsibility

Configuring build results expiry for a plan

By enabling build expiry for just a plan (described below), you override the [global expiry settings](#) that affect all plans. If you disable build expiry for a plan, that plan's build result data will never be automatically deleted from your Bamboo server.

You can choose the build result data that will be kept for a plan and for how long this data will be kept (e.g. for [reporting](#) purposes), before Bamboo automatically deletes it.

You can also delete the results of a plan build manually — see [Deleting the results of a plan build](#).

On this page:

- [Configuring the expiry of build results for a plan](#)
- [Disabling the expiry of build results for a plan](#)

Related pages:

- [Editing a plan's configuration](#)
- [Configuring global build results expiry](#)

Configuring the expiry of build results for a plan

Before you begin:

- Ensure that you [back up](#) your build results data before its expiry date is reached.

To enable and configure the expiry of build result data for a particular plan:

1. Navigate to the configuration for the desired plan, as described on [Editing a plan's configuration](#).
2. Click **Miscellaneous** to display the plan's current build expiry settings.
3. Select the **Override global build expiry configuration** check box. The following fields will be displayed:

Setting	Result
Build results	All build results data (including artifacts and build logs) are deleted.
Artifacts	Only user-defined artifacts are deleted from the build results.
Build logs	Only build logs are deleted from the build results.
Expiry period	Specifies the period (days, weeks or months) for which you want to keep build results. E.g. specify '24 months' to keep all build results for the last two years.
Minimum builds to keep	Specifies the minimum number of build results you want to keep. E.g. specify '50' to keep the latest 50 build results, even if they are older than the period specified with Expiry period .
Labels to keep	Specifies the build labels (not plan labels or job labels) applied to builds for which you want to keep build results, regardless of the Expiry period and Minimum builds to keep settings. Note that builds can be labelled either manually or automatically .

4. Click **Save**.

 Note that the build expiry event is a global event that runs periodically, regardless of whether you disable or enable build expiry in your plans. When this event occurs, the build results for your plan will be expired according to the criteria specified in the settings above or globally. To configure the global event and global build expiry settings, please refer to [Configuring global build results expiry](#).

Screenshot: Enabling Build Expiry

Build Expiry

Bamboo will remove expired data when the global build expiry schedule is triggered. You can override the global criteria for expired data, for this plan, in the settings below.

- ☒ **Override global build expiry configuration**
This will enable and override build expiry for this build

What should be expired?

- ☒ **Build results**
The entire result will be removed (including artifacts)
- ☒ **Artifacts**
User defined artifacts will be expired
- ☒ **Build logs**
Build log will be expired

Expiry criteria

Expiry period months

Expire builds that completed before the above time period. Use 0 to ignore this option.

Minimum builds to keep

Minimum number of builds per plan to keep (e.g. 1 will keep the last build only), the rest will expire. Use 0 to ignore this option.

- ☐ Don't expire builds with certain labels

Disabling the expiry of build results for a plan

To disable expiry of the build result data for a particular plan:

1. Navigate to the configuration for the desired plan, as described on [Editing a plan's configuration](#).
2. Click **Miscellaneous**.
3. Select the **Override global build expiry configuration** check box.
4. Enter '9999 months' for the **Expiry Period**.

 This is a workaround for this Bamboo issue ([BAM-4270](#)). Please vote for this issue, if you would like to see it implemented in Bamboo.

5. Click **Save**.

Screenshot: Disabling Build Expiry

Build Expiry

Bamboo will remove expired data when the global build expiry schedule is triggered. You can override the global criteria for expired data, for this plan, in the settings below.

- ☒ **Override global build expiry configuration**
This will enable and override build expiry for this build

What should be expired?

- ☒ **Build results**
The entire result will be removed (including artifacts)
- ☒ **Artifacts**
User defined artifacts will be expired
- ☒ **Build logs**
Build log will be expired

Expiry criteria

Expiry period months

Expire builds that completed before the above time period. Use 0 to ignore this option.

Minimum builds to keep

Minimum number of builds per plan to keep (e.g. 1 will keep the last build only), the rest will expire. Use 0 to ignore this option.

- ☐ Don't expire builds with certain labels

Deleting the results of a plan build

If the results of a plan builds are no longer required, you can completely remove the them from your Bamboo system. The results include all the results of all job builds that were processed as part of an individual plan build (with a specific build number).

Note that you can also remove job build result data that reaches a particular age. See [Configuring global build results expiry](#) or [Configuring expiry of a plan's job build results](#) for more information.

Related pages:

- [Deleting a job's current working files](#)

Before you begin:

- The ['Admin' global permission](#) or ['Admin' plan permission](#) is required to delete plan build results.
- The result of a plan build cannot be deleted if that plan is currently being built. If you need to delete the result of a plan build, stop the plan's build first. Refer to [Stopping an active job build](#) for more information.

To delete the result of a plan build:

1. Click **Dashboard** and then the **All Plans** tab.
2. In the list of plans, click the name of the desired plan.
3. Click the **History** tab. A table of completed plan build results will be displayed, with the most recent builds at the top.
4. Locate the desired build result and click **Delete**. (see [screenshot below](#)).
5. Confirm the deletion. The plan build result and any [artifacts](#) generated as a result of the plan build's execution will be deleted.

Screenshot: Build history of a plan showing individual plan build results

Plan Summary Branches Recent Failures **History** Tracking Issues Configuration

Run Actions

Completed Plan Results

Status	Reason	Completed	Duration	Test Results
❗ #69	Manual build by Jens Schumacher	Wed, 25 Jan, 01:41 PM	39 seconds	Testless build Delete
❗ #68	Manual build by Jens Schumacher	Tue, 29 Nov, 03:50 PM	42 seconds	Testless build Delete
❗ #67	Manual build by Giancarlo Lionetti	Tue, 11 Oct, 10:22 AM	10 seconds	Testless build Delete
❗ #66	Manual build by Giancarlo Lionetti	Tue, 11 Oct, 10:19 AM	9 seconds	Testless build Delete
👍 #65	Manual build by Jens Schumacher	Mon, 26 Sep, 03:36 PM	6 seconds	Testless build Delete
👍 #58	Manual build by Giancarlo Lionetti	Thu, 15 Sep, 08:01 AM	1 minute	Testless build Delete

Working with comments

Comments are a useful way to record and share information about builds. There are two types of comments in Bamboo:

- Comments you make **when you commit code** — these comments are automatically copied into Bamboo from your source-code repository. See [Viewing code check-in comments](#).
- Comments you make **about a build result** — these are comments that you make ad-hoc about a particular build result. See [Commenting about a build result](#) and [Viewing comments about a build result](#).

Related topics

For information on working with comments, see the following topics:

- [Commenting about a build result](#)
- [Viewing comments about a build result](#)
- [Viewing code check-in comments](#)

Commenting about a build result

Bamboo allows you to record comments about a [build result](#). This is a convenient way to record relevant information for future reference, and to collaborate with colleagues.

You can include JIRA issue keys into your comments. Bamboo will automatically convert these into hyperlinks to the respective JIRA issues. Bamboo needs to have been configured for [JIRA integration](#), and the issue key must be of the default JIRA issue key format (that is, two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example BAM-123).

Before you begin:

- You must be logged in to Bamboo before you can comment on a build result.

Related pages:

- [Working with comments](#)
- [Viewing a build result](#)

To comment on a build result:

1. Navigate to the desired build result, as described on [Viewing a build result](#).
2. In the **'Build Result'** screen, click **Comment**.
3. Type your comment into the **Comment** box, then click **Add**.

SummaryTestsChangesArtifactsLogsMetadataIssuesActions ▾

Build Result Summary

Details

Trigger

Code has been updated by
Joseph Walton.

Revision

138589

Completed

17 Jan 2011, 5:58:30 PM – 1
month ago

Duration

8 minutes

Successful since

#8 (1 week before)

Labels

None Add

Code Changes

See full change details ▶

Joseph Walton
CORE-148 Merge forward check for
incomplete parsing.

JIRA Issues

See full issue details ▶

CORE-148 (Related)
Parse internationalised duration strings

Shared Artifacts

See all Artifacts

Core POM (102 bytes)

Core Target (604 KB)

XML POM (102 bytes)

1 more Shared Artifacts...

Comments

No comments on this Build.

Comment:

This is a comment.

Add

Cancel

Test Summary

1,011 tests in total

0

0

0

New Failures

Existing Failures

Fixed

Viewing comments about a build result

Bamboo allows you to record comments about a [build result](#). This is a convenient way to record relevant information for future reference, and to collaborate with colleagues. You can view comments recorded against build results by other users.

Viewing comments about a particular build result

To view comments about a particular build result:

1. Navigate to the desired build result, as described on [Viewing a build result](#).
2. A list of all comments about this build result will be displayed in the 'Summary' tab, including author and timestamp. Comments added to the job that produced the build result will also be displayed.

On this page:

- [Viewing comments about a particular build result](#)
- [Viewing comments on the Plan or Job Summary](#)

Related pages:

- [Working with comments](#)
- [Commenting about a build result](#)
- [Viewing a build result](#)

Screenshot: Comments about a build result and related job

Summary Tests Changes Artifacts Logs Issues Sandbox Actions

Build Result Summary

Details

Trigger This is the initial clean build.

Queued 21 Oct 2010, 3:56:35 PM

Status Reason for not building is unknown

Labels None Add

Code Changes

No changes found for this Build.

Comments


Andrew Lui 15 Feb 2011, 8:26:44 PM
Test Delete

Comments on Default Job


Andrew Lui 15 Feb 2011, 8:27:08 PM
Test Delete

Comment

Viewing comments on the Plan or Job Summary

To view comments about a build result a Plan summary or Job summary:

1. Navigate to the desired plan or job, as described on [Configuring plans](#) and [Jobs and tasks](#).
2. The plan or job's build results will be displayed in the 'Recent History' section of the Plan or Job Summary. The message icon () indicates that there are one or more comments about a particular build result. Hover your mouse over the icon to see the comments.

Screenshot: Viewing comments on a job summary

Job Summary Recent Failures History Tests Logs Configuration Showing Last 25 builds Actions

Job Summary

Current Activity

No Jobs are currently running.

Job Statistics

0%
Successful

Successful Jobs: 0 / 0
Average Duration: < 1 second

Recent History

 #1 Initial clean build No tests found

 Feed Andrew Lui on Default Job 2 days ago:
Test

Viewing code check-in comments

If a build was triggered by a code change, the commit comment (or check-in comment) will be shown in the build result.

To view the code check-in comments for a particular build result:

1. Navigate to the desired build result, as described on [Viewing a build result](#).
2. The build's commit comment will be shown to the right of the screen, under the heading 'Code Changes'.

Code Changes

[View all 3 code changes ▶](#)

 **Jesper Särnesjö**
Merge

24 May 2012, 12:09:12 PM

 **Jesper Särnesjö**
BAM-11045: Update plan variables description

23 May 2012, 3:25:00 PM

[1 more change...](#)

You can see more details on the **Changes** tab of the Build Result Summary page.

Note that you can include links to JIRA issues in a change comment simply by typing the key. Bamboo automatically converts the key to a link, as long as the key is of the default JIRA issue key format (that is, two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example BAM-123).

Related pages:

- [Working with comments](#)
- [Triggering builds](#)

Working with labels

About labels

A *label* is a convenient way to tag and group [build results](#) that are logically related to each other. Labels can also be used to define [RSS feeds](#) and to control [build expiry](#).

Labels can be applied to build results automatically, by specifying the label(s) in a plan (note that only Bamboo administrators can do this). Labels can also be [applied ad hoc](#) to build results by Bamboo users.

With Bamboo, you can label your [build results](#) in whatever way works best for your team. Labels are not restricted to a particular [plan](#), so you can apply the same label to build results from different plans.

For example, it might not be practical for your QA team to review every build, and you need to know which builds they have reviewed. By using labels such as "qa_passed" and "qa_failed", Bamboo allows them to simply indicate which builds have passed and failed QA.

Related pages

For information on working with labels, see the following topics:

- [Labelling a build result](#)
- [Removing a label from a build result](#)
- [Viewing labelled build results](#)
- [Viewing popular labels](#)
- [Labelling a plan](#)

Labelling a build result

With Bamboo, you can label your [build results](#) in whatever way works best for your team. Labels are not restricted to a particular [plan](#), so you can apply the same label to build results from different plans.

For example, it might not be practical for your QA team to review every build, and you need to know which builds they have reviewed. By using labels such as "qa_passed" and "qa_failed", Bamboo allows them to simply indicate which builds have passed and failed QA.

You can include JIRA issue key in the label, as long as the key is of the default JIRA issue key format (that is, two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example BAM-123).

Before you begin:

- You must be logged in to Bamboo before you can label a build result.

Related pages:

- [Working with labels](#)

To label a build result:

1. Navigate to the desired build result, as described on [Viewing a build result](#).
2. Click the pencil icon (), next to **Labels** in the 'Details' section.
3. Type the relevant label (or multiple labels, separated by commas or spaces). Note that the label will be saved in lowercase characters.
4. Click **Close**.

Note that:

- You can [view a list](#) of existing labels by clicking the **Labels** link.
- You can also label a build result using [Instant Messaging](#).

Removing a label from a build result

A *label* is a convenient way to tag and group *build results* that are logically related to each other. Labels can also be used to define [RSS feeds](#) and to control [build expiry](#).

Labels can be applied to build results automatically, by specifying the label(s) in a plan (note that only Bamboo administrators can do this). Labels can also be [applied ad hoc](#) to build results by Bamboo users.

Before you begin:

- You must be logged in to Bamboo before you can label a build result.

Related pages:

- [Working with labels](#)

To remove a label from a build result:

1. Navigate to the desired build result, as described on [Viewing a build result](#).
2. Click the pencil icon (✎), next to the **Labels** in the 'Details' section.
3. Click the 'x' at the right of the label you want to remove.
4. Click **Close**.

Viewing labelled build results

A *label* is a convenient way to tag and group *build results* that are logically related to each other. Labels can also be used to define [RSS feeds](#) and to control [build expiry](#).

Labels can be applied to build results automatically, by specifying the label(s) in a plan (note that only Bamboo administrators can do this). Labels can also be [applied ad hoc](#) to build results by Bamboo users.

Related pages:

- [Working with labels](#)

To view all build results which have a particular label:

1. Navigate to the **All Plans** tab of the Dashboard.
2. Click **Filter Plans** (or the label name).
3. Click the label of interest. The list of all build results which have that label will be displayed.

Viewing popular labels

A *label* is a convenient way to tag and group *build results* that are logically related to each other. Labels can also be used to define [RSS feeds](#) and to control [build expiry](#).

Labels can be applied to build results automatically, by specifying the label(s) in a plan (note that only Bamboo administrators can do this). Labels can also be [applied ad hoc](#) to build results by Bamboo users.

When [labelling a build result](#), it can be useful to see which labels are most popular, that is, most frequently used by your colleagues.

Related pages:

- [Working with labels](#)

To view the most popular labels:

1. Navigate to the desired build result, as described on [Viewing a build result](#).
2. Click the **Labels** link in the 'Details' section.
3. Click **See also labels in All Projects**.
4. Click **By Popularity**.

This will display a list all labels that are used in Bamboo, by popularity. You can click any label to see a list of all build results which have that label.

Quarantining failing tests

There may be times when you want to prevent a failing test from causing the whole build to fail.

Possible scenarios where this may be useful include:

- You want to build an artifact despite there being a failing test, but can't do this while the plan build is failing.
- In test-driven development (TDD), a test will fail until the functionality is implemented - you want to quarantine all but the relevant tests.
- A test may give unpredictable results, perhaps because of infrastructure issues or dependencies.
- You want to remove a test from a build, but don't want to alter or delete the test source code because doing so could affect another Bamboo plan.

In Bamboo, you can temporarily disconnect any test's results from the plan build results by quarantining the test. The test is still run whenever the plan is built, but the test's results do not affect the plan's build results.

You can always restore a test's results to the build results when required, for example if the test is now passing.

All the quarantined tests for a plan are displayed on the **Quarantined Tests** tab of the plan summary. The status bar for each test shows the

recent build history of the test.

On this page:

- To quarantine a failing test
- To restore a quarantined test to a build

Related pages:

- Working with builds
- Viewing a plan's build information
- Viewing test results for a build
- Viewing a build result
- Configuring plans

To quarantine a failing test

You need plan administrator permission to quarantine a test.

1. Choose **Dashboard** > **All Plans** > **#buildresult** to go to the build result where the test is failing.
2. Click **Quarantine** for the failing test (in the 'Build Result Summary' screen).

To restore a quarantined test to a build

You need plan administrator permission to restore a test.

1. Choose **Dashboard** and click on a plan to go to the plan's summary.
2. Click the **Quarantined Tests** tab.
3. Click **Unleash** for the test to be restored.

Screenshot: The quarantined tests for a plan, showing the Status bar.

Plan Summary Recent Failures History **Quarantined Tests** Configuration Run Plan Actions

Quarantined Tests

When tests are known to fail, or fail intermittently, you can temporarily make Bamboo ignore their result by putting them into quarantine.

Status [Green][Green][Green][Red][Green][Green]
The test status shows the recent results of a test, with the most recent on the right. If the status indicator is mostly green, it might be time to **unleash** the test back into the wild. This test status bar only shows the status of tests while they were in quarantine.

Test	Job	Status	Quarantined by	Quarantined at	
FailingTest testShouldFail ⓘ	Default Job	[Red][Red][Red][Red][Red][Red][Red]	Admin User	17 Jan 2012, 9:27:33 AM	Unleash
FailingTest testShouldFailSlowly ⓘ	Default Job	[Red][Green][Green][Red][Red][Red][Red]	Admin User	16 Jan 2012, 2:00:55 PM	Unleash

Setting up plan build dependencies

You may want to trigger a plan build when another plan's build has successfully completed. This ensures that changes to any job's source code associated with one plan does not break the build of another dependent plan (known in this context as a 'child' plan).

For example, there could be two plans in Bamboo:

1. **Acme – Core** — which contains the core code for an application.
2. **Acme – Plugin** — which contains code for a plugin to the application.

In this scenario, the **Acme – Plugin** plan is a *child* of **Acme – Core**. Any changes to source code associated with the **Acme – Core** plan should trigger a build of **Acme – Plugin**.

On this page:

- Triggering dependent plans
- Automatic dependency management with Maven 3
- Dependency blocking
- Notes

Triggering dependent plans

To trigger a child plan to build when this plan builds successfully:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list and click the edit icon to display the plan's Configuration pages.
3. Click the **Dependencies** tab
4. Under 'Child Plans', begin typing a plan name in **Search for plan** to select child plans to trigger. You can set multiple plans to be triggered.

5. Click **Save**.

Automatic dependency management with Maven 3

Automatic Dependency Management is a feature for users who use Maven 3 and wish for their parent and child dependencies to be set up according to the dependencies in the Maven pom.xml. Every time the plan is run, the Bamboo Automatic Dependencies are updated to reflect any additions or removals of Maven dependencies.

To setup automatic dependency management:

1. Click **Dashboard** and then the **All Plans** tab.
2. Locate the plan in the list and click the edit icon to display the plan's configuration pages.
3. Locate the job that contains the pom.xml you wish to use to automatically update plan dependencies by analysing a Maven pom file.
4. Choose **Actions > Configure Job**.
5. Click on the **Tasks** tab.
6. Click **Add Task** and add the **Maven Dependency Processor** task to the job. For best results, ensure that the task runs last by dragging it to the bottom of the task list. For more information on configuring tasks, see [Configuring tasks](#).

Setting	Notes
Override Project File	<i>Optional.</i> The location relative to the working directory or sub-working directory where the project file (pom.xml) is located.
Working Sub Directory	<i>Optional.</i> The sub directory from which the Task should look for the project file (pom.xml)
Alternate location of settings.xml	<i>Optional.</i> Specify an alternate settings.xml to be used if the Task needs to resolve dependencies from specific Maven repositories.
Path to Maven local repository	<i>Optional.</i> Specify a full path to a local Maven repository for the Task to use to resolve dependencies.

7. Click **Save**.
8. Use the Plan Navigator to return to the plan.
9. Click the **Dependencies** tab.
10. Select **Automatic Dependency Management**. You should see the name of the job for which you configured the **Maven Dependency Processor** appear.
11. Click **Save**.

Dependency blocking

Dependency blocking is an advanced feature of dependent build triggering that can be used to manage plan builds with parent build dependencies. This ensures that a "tree" of dependent builds always runs in tree hierarchy order, even if child plan builds are triggered independently of their parents. For more information, see [Dependency blocking strategies](#). Please note, dependency blocking only works when the plan build is triggered because of source repository code updates.

Notes

Build dependencies work together with the trigger configuration of plans to trigger builds of these plans. For example, you can set up Plan A to [poll its repository for changes](#) as well as to be dependent on a parent plan (Plan B). In this case, builds of Plan A will be triggered when code changes are detected in its repository *and also when* builds of Plan B complete successfully. If you want your builds to *only* be triggered by successful parent builds from your build dependencies, don't configure triggering for your child plans at all. See [Running a plan build manually](#).

- If the child build *uses the same source* as the parent build (for example, the Subversion URL is the same), the child build will be forced to check out the same revision of source code as the parent build. This ensures that builds are consistent when triggering one build from another.
- Take care not to create *circular dependencies*, where your child build triggers one of its parent builds. Otherwise your plans may build continuously. See [Running a plan build manually](#).

Dependency blocking strategies

Dependency blocking is an advanced feature of [dependent build triggering](#) that can be used to manage the builds of plans that have parent plans. This ensures that a 'tree' of dependent builds always runs in tree hierarchy order, even if child plan builds are triggered independently of their parents.

The three dependency blocking strategies are:

Do not block	When triggered by a source code update, the plan will always be built, regardless of any parent plan build dependencies.
Block build if parent builds are queued or in progress	When triggered by a source code update, the plan will <i>not</i> be built if its parent plans are building or are waiting in the build queue.

Block build if parent plans have unbuilt changes

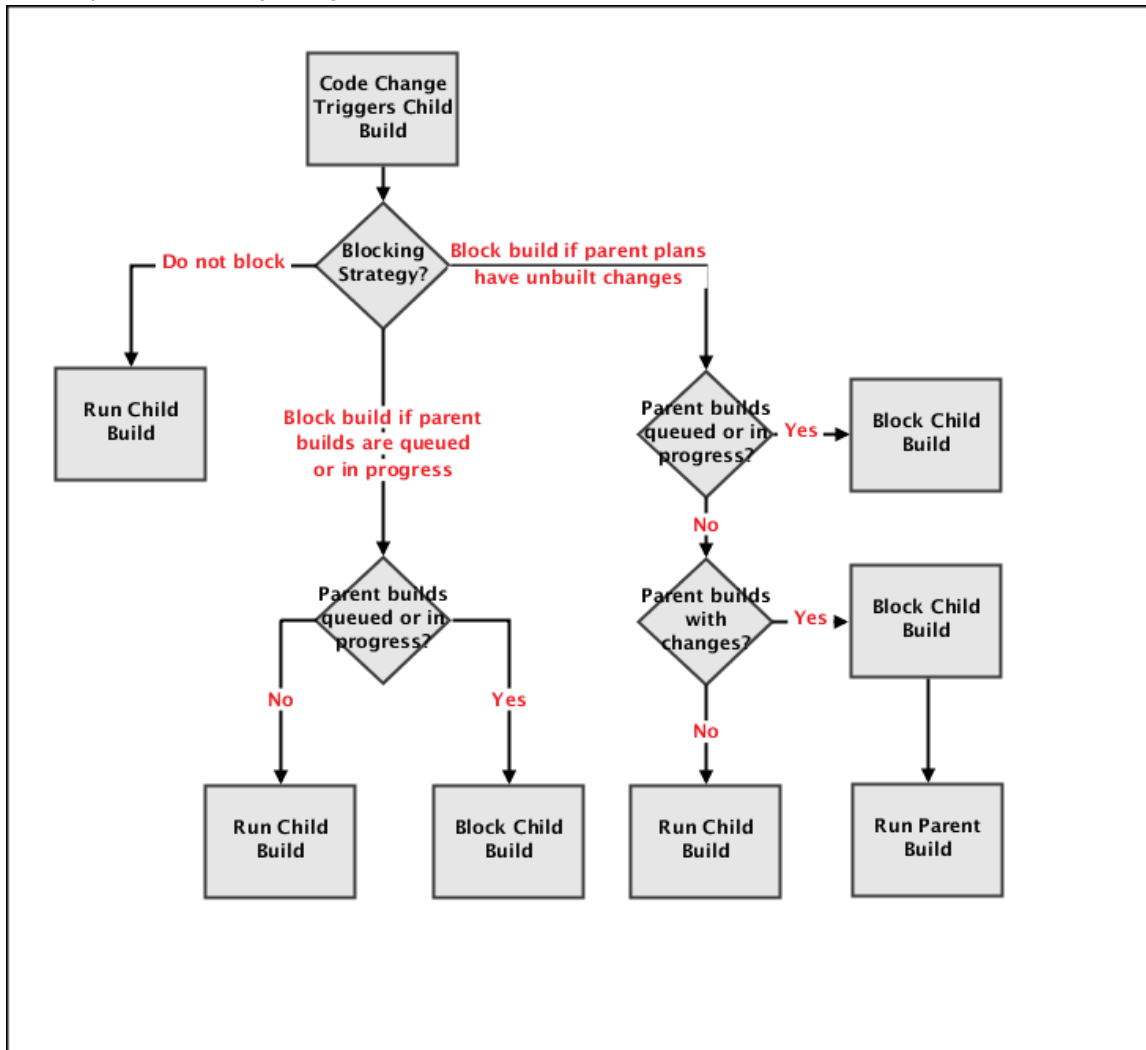
When triggered by a source code update, the plan will *not* be built if its parent plans are building, are waiting in the build queue, or have changes.

When Bamboo finds parent plans with source repository changes, those plans will be triggered and your plan will be blocked.

Note that for the *Block build if parent plans have unbuilt changes* option, only the repositories of parent plans that are specified by triggers (that is, by the [Polling the repository for changes](#) or [Repository triggers the build when changes are committed](#) trigger types) are scanned for unbuilt changes; if there are repository changes (for parent plans), then the parent plans are triggered and the current plan is blocked.

! *Dependency blocking only works when the plan uses a trigger configuration based on source code updates (i.e. [Polling the repository for changes](#) or [Repository triggers the build when changes are committed](#)). This feature will not work when a plan uses a trigger configuration based on a schedule or triggered via a parent build (when there are multiple parent plan builds in progress).*

These dependence blocking strategies are illustrated in the flowchart below:



Viewing test statistics for a job

Bamboo provides a summary of test results across all of a job's builds. This helps you to:

- **Troubleshoot** by identifying which tests fail most frequently, and which tests take longest to fix.
- **Manage your build duration** by identifying the plan's slowest running tests.
- **Ensure quality** by monitoring the number of tests over time: are your test cases growing with your code base?

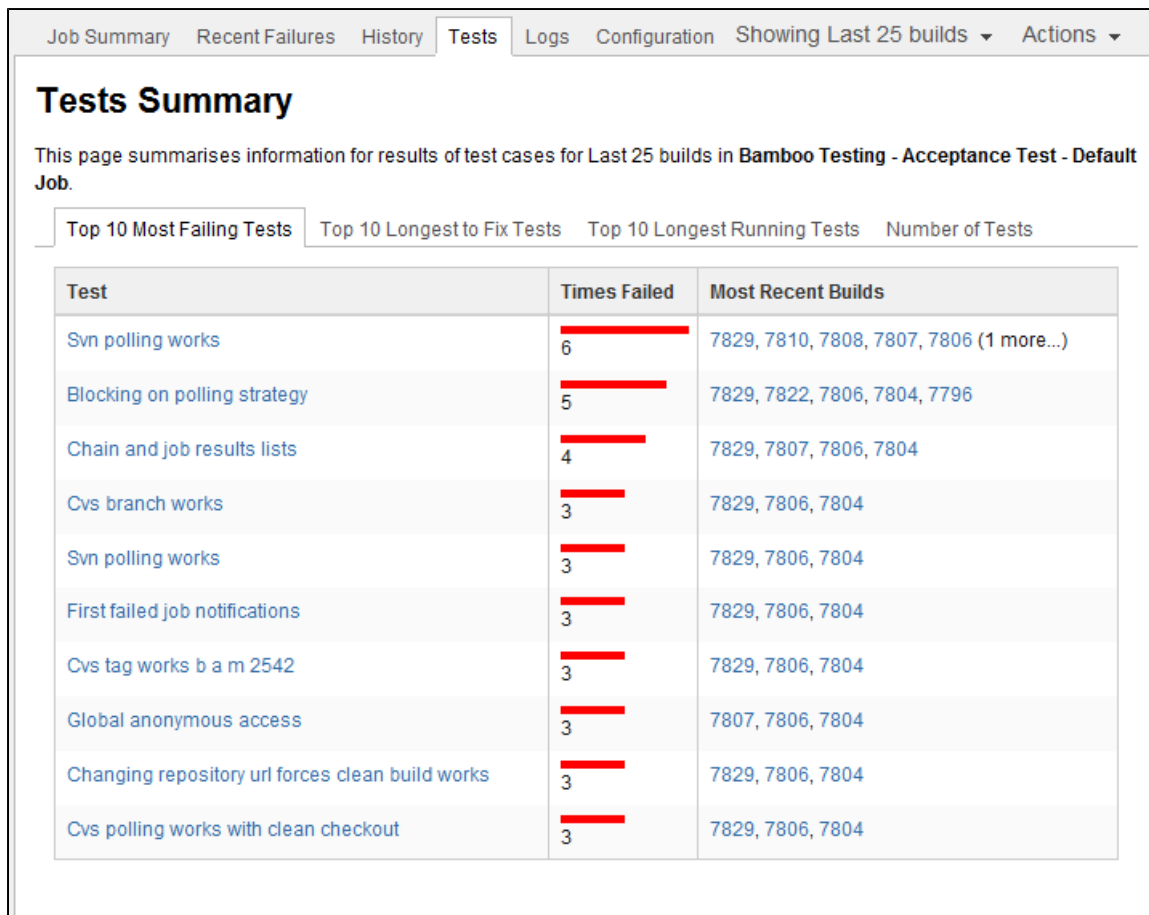
Related pages:

- [Reporting](#)

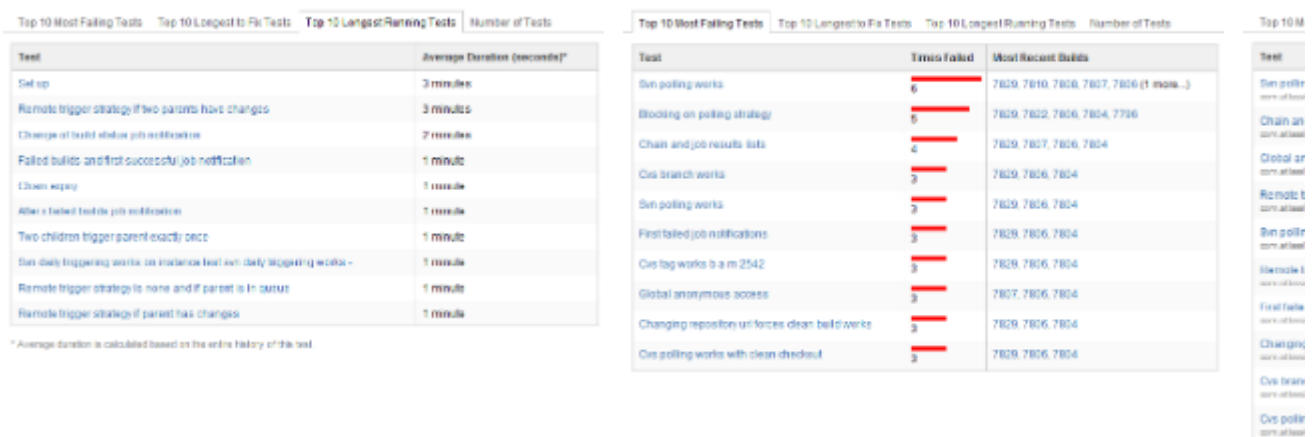
To view the test statistics for all of a job's builds:

1. Navigate to the desired build result page, as described in [Viewing a build result](#).
2. Click the **Tests** tab.
3. Click the sub-tabs to filter the test statistics (see screenshots below).
 - To view a test's [history](#), click the test name.

Screenshot: Test statistics for a job



Screenshots: Test statistics for a job - tabs



1. Most Failing

Reordering jobs in the build queue

Bamboo automatically assigns a plan's jobs to the **build queue** when the plan is **triggered** and no agents are available to run them. The build queue is displayed on the **Current Activity** tab of the **Dashboard**.

If you want to prioritise one job build over another in the build queue, you can manually reorder these jobs in the build queue. This will not force a job build to run immediately, but will promote it in the build queue. Your job build will still require an agent (which has the capabilities to meet the job's requirements) to become available. Similarly, you can demote a job build in the build queue if you do not need it to run urgently.

Bamboo administrators can reorder plans in the queue. To do this, use the  icon to move the plan to its new position in the queue.

Stopping an active build

The instructions on this page describe how to stop a plan or job build that is running.

Note that if your Bamboo server runs on Windows, it may only be possible to stop an active build by going to the Windows Task Manager and ending the relevant processes.

To start a building a plan manually, see [Running a plan build manually](#).

On this page:

- [Stopping an active plan build](#)
- [Stopping an active job build](#)

Related pages:

- [Running a plan build manually](#)
- [Disabling or deleting a plan](#)
- [Disabling or deleting a job](#)

Stopping an active plan build

To prevent Bamboo submitting a plan to the build queue, refer to [Disabling or deleting a plan](#).

To stop an active plan build:

1. Click **Dashboard** and then the **All Plans** tab.
2. Click the 'Stop' icon next to the active plan you want to stop.

Stopping an active job build

To prevent Bamboo submitting a job to the build queue, refer to [Disabling or deleting a job](#).

To stop an active job build:

1. Click **Dashboard** and then the **All Plans** tab.
2. Click the name of the plan.
3. Click the 'Stop' icon next to the active job you want to stop (in the 'Current Activity' section).

Getting feedback

Getting immediate feedback about build results is the essence of continuous integration. Furthermore, getting reports on activity of your development team can give you deep insights into your process efficiencies and schedule risks.

Notifications

Bamboo can send [notifications](#) to your team about the success or failure of their builds in a number of ways:

- [The Wallboard](#)
- [Email](#)
- [Instant messaging](#)
- [RSS feeds](#)

Reports

Bamboo provides various [reports](#) about the build activity of your development team:

- [Summary statistics for all users](#)
- [Build results for an author](#)
- [Comparison charts for authors](#)
- [Comparison charts for plans](#)
- [Clover code-coverage for a job](#)
- [Clover code-coverage for a build](#)

Notifications

Bamboo can send notifications about build results so that you can find out immediately about the success or failure of your builds.

You can get notifications in different ways:

Bamboo Wallboard	Show build results on a dedicated monitor. See Displaying the wallboard .
Email (e.g. GMail)	Get build results in your inbox. See Configuring notifications .
Instant messaging (e.g. HipChat, Google Talk)	Send notifications to your dev chat room. See Configuring notifications .

RSS feeds

Get aggregated key information about your builds.

See [Subscribing to RSS feeds](#).

See also [Changing your notification preferences](#).

Displaying the wallboard

A development team can benefit from setting up a dedicated monitor to display Bamboo's latest build results using the Bamboo wallboard.

The Bamboo wallboard can display the latest results for:

- all plans that you have permission to see.
- just your favourite plans.
- plans filtered by plan label.

The [branches wallboard](#) displays the status of all the branches and the plan that the branches belong to.

 Jason Acceptance Test Z Personal Branches 3 months ago Updated by Jason Berry	 Mercurial Chain Z Personal Branches 1 month ago Manual build by Jens Schumacher	 Acceptance Test JDK 1.6 Bamboo 12 hours ago Manual build by Przemyslaw Bruski	 CI Tests Bamboo 7 hours ago Manual build by Giancarlo Lionetti
 Deploy Bamboo 12 hours ago Scheduled build	 Remote Agent Functional Bamboo 4 hours ago Scheduled build	 Stable CI Tests Bamboo 43 minutes ago Updated by Brydie McCoy	 Atlassian Command Line Bamboo Plugins 7 hours ago Dependant of TRIGGER-TRUNK-2061
 Bamboo FindBugs Plugin Bamboo Plugins 7 hours ago Dependant of TRIGGER-TRUNK-2061	 Bamboo JMeter Aggregate Bamboo Plugins 6 hours ago Dependant of TRIGGER-TRUNK-2061	 Bamboo Sandbox Plugin Bamboo Plugins 6 hours ago Dependant of TRIGGER-TRUNK-2061	 Bamboo Sonar plugin Bamboo Plugins 6 hours ago Dependant of TRIGGER-TRUNK-2061

On this page:

- [How do I do that?](#)
- [Notes](#)

Related pages:

- [Getting feedback](#)
- [Using the Bamboo Dashboard](#)

How do I do that?

Log in to Bamboo. *This is optional when displaying all plans if your Bamboo administrator has allowed [anonymous access](#),*

Go to the Dashboard.

Task	Action	Notes
All plans	Wallboard > All Plans	Alternatively, use the following URL in your browser, replacing 'bambooserver' with the real name of your Bamboo server: <code>http://bambooserver:8080/bamboo/telemetry.action</code>
Favourite plans	Wallboard > Favourite Plans	Only users who have logged in to Bamboo can specify and access favourites . Alternatively, use the following URL in your browser, replacing 'bambooserver' with the real name of your Bamboo server: <code>http://bambooserver:8080/bamboo/telemetry.action?filter=favourites</code>
Filtered plans	Wallboard > Filtered Plans	You need to have set up a plan filter first. See Using the Bamboo Dashboard .

Notes

- You will only be able to display those plans that you have permission to see.
- Once you are viewing the wallboard in your browser window, set your browser to 'full screen' mode to make the wallboard fill your entire screen. (Use **F11** for common browsers on Windows and UNIX/Linux-based systems and **Shift+Cmd+F** for Firefox on Mac OS X.)
- If you are going to display the wallboard permanently, you may want to ask your Bamboo administrator to [create a user](#) who has only

a limited set of [permissions](#).

- If your wallboard is displayed on a touchscreen (such as an iPad) or its content can be accessed with a 'human interface device', such as a mouse, then touching or clicking a build result on the wallboard shows more information about that build.

Screenshot: More information from a build result on the wallboard



Configuring notifications for a plan and its jobs

Notifications in Bamboo are triggered by a range of events involving a plan and its jobs, including build completion, build outcomes and comments being posted against build results. You can configure whether notifications are sent for a particular event and who they are sent to. Users can choose whether to receive their notifications via [email](#), [IM](#), [both](#) or [neither](#).

For each plan or job, you can specify different recipients for each type of event notification. Also be aware that these recipients do not require [Bamboo user accounts](#).

Adding notifications for a plan or job

Before you begin:

On this page:

- [Adding notifications for a plan or job](#)
- [Notification events](#)
- [Removing notifications from a plan or job](#)

Related pages:

- [Notifications](#)
- [Editing a plan's configuration](#)
- [Changing your notification preferences](#)
- [Granting plan permissions in bulk](#)
- [Configuring Bamboo to send SMTP Email](#)
- [Configuring Bamboo to use Instant Messaging](#)

- You must have the ['Edit' permission for a plan](#), to add or remove notifications for it.
- You need to configure Bamboo's SMTP email and/or instant messaging capabilities before Bamboo can send notifications. If you have not configured either or both of these, a note will display on the page prompting you to set up the appropriate server(s):
 - To configure an email server for Bamboo, click **Add an Email Server** in the note and enter the email server details in the window that displays. See [Configuring Bamboo to send SMTP Email](#) for more information.
 - To configure an instant messaging server for Bamboo, click **Add an Instant Messaging Server** in the note and enter the instant messaging server details in the window that displays. See [Configuring Bamboo to use Instant Messaging](#) for more information.

To add a notification for a plan or its jobs:

1. Navigate to the configuration for the desired plan, as described on [Editing a plan's configuration](#).
2. Click the **Notifications** tab.
3. Set up a new notification in the 'Add Build Notification' section as follows:

Setting	Notes
Event	Select the event type you want to be notified about. Refer to the list of events (below) for details.
Recipient Type	User — Enter the username of the appropriate Bamboo user, or click the icon to select from a list of users.
	Hipchat — Enter the Hipchat API Token and Room Name . See Integrating Bamboo with HipChat for more information.
	Group — Enter the name of the appropriate Bamboo group(s) .

Email Address — You can use email to send notifications to a person who is not a Bamboo user. Type the appropriate email address. Note that:

- If you specify the email address of an existing Bamboo user, the user will receive notifications even if they have elected not to receive notifications in their user preferences.

IM Address — This is useful if you need to send Instant Messenger (IM) notifications to a person who is not a Bamboo user. Type the appropriate IM address. Note that:

- If you specify a broadcast address (eg. '[project-x@broadcast.chat.mycompany.com](#)'), Bamboo will not know the context of related IM responses.
- If you specify the IM address of an existing Bamboo user, the user will receive notifications even if they have elected not to receive notifications in their user preferences.

Responsible Users — The Bamboo users who have been assigned as being responsible for a broken build. See [Assigning responsibility for build failures](#).

Committers — The Bamboo users who have committed code to a particular build since build was last checked out by Bamboo.

Watchers — The Bamboo users who have marked this plan as one of their [favourites](#).

4. Click **Add**, then configure further notifications if required.
5. Click **Save** when you have finished.

Screenshot: Plan build notifications

Plan Details Source Repositories Branches Stages Dependencies Permissions **Notifications** Variables Miscellaneous Audit Log Run Actions

Notifications

There are currently no notifications set up for this Plan.

Add Build Notification

Select notification event and recipient

Event **All Builds Completed**
Notification sent for every fully completed build

Recipient Type **User**

User
The username of the user to notify

Add Cancel

Notification events

Plan Events	Description
All Builds Completed	Bamboo will send a notification whenever the plan build finishes, regardless of the plan build's result. This notification is recommended for any plans whose latest build activity is critical for people to be informed about. ✓ This is a good plan-based notification to use if you are new to Bamboo. You can change it to a less obtrusive notification option as you become more confident with continuous integration and Bamboo's build processes.
Change of Build Status	Bamboo will send a notification only when there has been a change in status of the plan's build activity over consecutive plan builds — for example, only whenever a plan's latest build changes from successful to failed or vice versa (i.e. 'fixed'). ✓ This notification option is less obtrusive than the other plan notifications mentioned above.

Failed Builds And First Successful	Bamboo will send a notification whenever: - a build of this plan fails. - the plan is 'fixed' (that is, the plan's latest build is successful <i>and</i> the previous plan build failed). ✔ This notification is generally suitable for the majority of plans.
After X Failed Builds	This notification allows you to specify the Number Of Failures (i.e. number of failed builds of this plan), after which Bamboo will send a notification. ✔ This notification option minimises the number of messages sent by Bamboo if the plan's builds fail on a frequent basis. You can also use this event to escalate plan build problems, for example, to notify a manager when a plan build fails five times.
Comment Added	Bamboo will send a notification whenever a comment is posted against a plan build result. The email notification will contain all comments against the plan build, whereas IM notifications will only contain the comment that triggered this notification event. ✔ This notification can help improve collaboration between team members. Be aware that you will not receive notifications for any comments which you post yourself.
Change of Responsibilities	Bamboo will send a notification whenever someone is added to, or removed from, the list of those responsible for a broken build. ✔ This notification can help improve collaboration between team members.
Job Events	
All Jobs Completed	Bamboo will send a notification whenever a job build of the plan finishes, regardless of the job build's result. This notification is recommended if the latest build activity of all jobs in this plan are critical for people to be informed about. ✔ This is a good job-based notification to use if you are new to Bamboo. You can change it to a less obtrusive notification option as you become more confident with continuous integration and Bamboo's build processes.
Change of Job Status	Bamboo will send a notification only when there has been a change in build activity status of the jobs within this plan over consecutive plan builds — for example, only whenever the latest build of any job in this plan changes from successful to failed or vice versa (i.e. 'fixed'). ✔ This notification option is less obtrusive than the other job notifications mentioned above.
Failed Jobs And First Successful	Bamboo will send a notification whenever: - a build of this job fails. - the job is 'fixed' (that is, the job's latest build is successful <i>and</i> the previous job build failed).
First Failed Job For plan	If multiple jobs fail in a plan, Bamboo will only send a notification for the first failing job detected by the Bamboo system. ✔ This is a less obtrusive notification option that informs about a failing job (and hence, plan) in the shortest possible time.
Job Error	Bamboo will send a notification whenever an error occurs in one of the plan's job build processes (i.e. the activities that Bamboo performs to run a job build). This event is not related to failures of the actual build itself (see the Failed Jobs And First Successful and Failed Builds And First Successful events above). For example, a notification will be sent if Bamboo encounters an error when connecting to the repository, or detecting changes.
Job Hung	Bamboo will send a notification whenever it determines that one of the plan's job builds has hung, according to the hung job build criteria (read more about configuring your hung job build settings). ✔ Use this notification to ensure that the relevant people are informed when a job build becomes unresponsive.

Job Queue Timeout	Bamboo will send a notification whenever one of the plan's job builds has been waiting in the queue for longer than the build queue timeout criteria (read more about configuring your job's Build Queue Timeout settings). ✔ Use this notification to ensure that the relevant people are informed when a job build is stuck in the build queue for too long.
Job Queued Without Capable Agents	Bamboo will send a notification whenever one of the plan's job builds is queued and there are no agents capable of building it. ✔ Use this notification to ensure that people are notified when changes to agents adversely affect your job's builds.

Removing notifications from a plan or job

Before you begin:

- You must have the 'Edit' permission for a plan, to add or remove notifications for it.

To remove notifications for a plan or its jobs:

- Navigate to the configuration for the desired Plan, as described on [Editing a plan's configuration](#).
- Click the **Notifications** tab.
- Click **Remove** for each of the notifications that you wish to remove.

Configuring Bamboo to send SMTP Email

Bamboo can send email notifications about its build results. There are two steps to setting this up:

- Configure Bamboo to send SMTP email (see below).
- Configure a plan to send SMTP email notifications about build results (see [Configuring notifications for a plan and its jobs](#)).

On this page:

- [Configuring Bamboo to send SMTP email](#)
- [Configuring email notifications for Gmail](#)
- [Notes](#)

Related pages:

- [Configuring notifications for a plan and its jobs](#)

Configuring Bamboo to send SMTP email

To configure Bamboo to send SMTP email:

- Click **Administration** in the top navigation bar.
- Click **Mail Server** in the left navigation column (under 'Communication'). This will display the 'Mail Server Details' page (see screenshot below).
- Edit the mail server settings as necessary:

Setting	Description
Name	A display-name for the email address e.g. 'SMTP Server'
From Address	The email address from which Bamboo notifications will be sent.
Subject Prefix	The text (if any) which will be added to the start of the email subject line. For example '[Bamboo]' will result in emails with subjects like: [Bamboo] TEST build 1,001 has FAILED (77 tests failed, no failures were new) : Change made by jsmith [Bamboo] TEST build 1,002 was SUCCESSFUL (with 77 tests) : Change made by jsmith
Email Settings	Choose either SMTP or JNDI . See the Notes about JNDI below.
SMTP Server	The address of the email server that Bamboo will use to send notifications e.g. 'mail.myserver.com'.
Username	The login name of the account that Bamboo will use to login to the SMTP server.
Password	The password of the account that Bamboo will use to login to the SMTP server.
JNDI Location	Depends on your application server, and on the location of the 'mail' resource within the JNDI tree you specify. E.g. 'java:comp/env/mail/BambooMailServer'.

4. Type a test email address in the **Test Recipient Address** box.
5. Click **Test**, and verify that a test email is received.
6. Click **Save**.

Screenshot: Email Server Details

Configure Mail Server Details

Configure Mail Server Details

What email settings should Bamboo use?. Please enter either a SMTP Server hostname, or the JNDI location of a javax.mail.Session object to use.

Name*
Provide a name for this email server (this will be displayed on notification emails).

From Address*
This is the email address Bamboo-generated emails are sent from.

Subject Prefix
This is the tag added to the start of the subject line to identify Bamboo-generated email. e.g. [Bamboo].

☐ Exclude email header "Precedence: bulk"

Email Settings
Choose either a SMTP server or a JNDI location

SMTP Server*
Specify the email server used to send out the build results. For example "mail.myserver.com".

SMTP Port
Specify the SMTP port (defaults to 25).

Username
☐ Change password?

Test Email Configuration

You can enter a test recipient email below and click Test. Bamboo will test whether this email setting is valid by sending a test email to the specified address(es).

Test Recipient Address (optional)
You can enter in one (or more, comma separated) email address to which Bamboo will send a test email.

[Cancel](#)

Configuring email notifications for Gmail

Gmail.com uses TLS (SSL). A JNDI connector needs to be configured. Unfortunately the Bamboo distribution does not yet support JNDI with TLS.

To enable Gmail as your mail server:

1. Install [Bamboo as war on Tomcat](#)
2. Add the following configuration to your apache-tomcat-xxx/conf/Catalina/localhost/server.xml file:

```
<Context path="/bamboo" docBase="${catalina.base}/bamboo.war"
reloadable="true">
    <Resource name="mail/GmailSmtpServer"
        auth="Container"
        type="javax.mail.Session"
        mail.smtp.host="smtp.gmail.com"
        mail.smtp.port="465"
        mail.smtp.auth="true"
        mail.smtp.user="<your-name>@gmail.com"
        password="<your-pw>"
        mail.smtp.starttls.enable="true"
        mail.smtp.socketFactory.class="javax.net.ssl.SSLSocketFactory"
    />
</Context>
```

3. Ensure that the files mail-X.X.jar and activation-X.X.jar exist only in the apache-tomcat-xxx/lib folder. You can move those installed at <Bamboo-Install>/WEB-INF/lib to apache-tomcat-xxx/lib if they don't exist there yet. If they are already there, you can delete those shipped with Bamboo.
4. Configure Bamboo to use a **JNDI Location** of java:comp/env/mail/GmailSmtpServer. Note that the JNDI Location is case sensitive and must match the resource name specified in server.xml.

Notes

You can use a mail session as an alternative to specifying mail details directly in Bamboo. You configure the mail session in your application server (e.g. in the server.xml file — see [Locating important directories and files](#)), and then use JNDI to look up the preconfigured mail session. JNDI has the following advantages:

- **Centralised management** - mail details are configured in the same place as database details, and may be configured through your application server administration tools.
- **Better security** - mail details are not available to Bamboo administrators through the Bamboo interface, and aren't stored in Bamboo backup files.
- **More SMTP options** - e.g. SSL. If you want to use SMTP over SSL you will need to use JNDI.

Configuring Bamboo to use Instant Messaging

Bamboo can send [Instant Messaging \(IM\) notifications](#) about its build results. There are two steps to setting this up:

1. Configure Bamboo to use Instant Messaging (see below).
2. Configure a plan to send IM notifications about its build results (see [Configuring notifications for a plan and its jobs](#)).

Please note, Bamboo supports [XMPP protocol](#) for messaging. This means Bamboo can be used with [Gtalk](#) or your enterprise XMPP server.

Related pages:

- [Configuring notifications for a plan and its jobs](#)
- [Configuring Bamboo to use Google Talk for Instant Messaging](#)

To configure Bamboo to use Instant Messaging:

1. Click **Administration** in the top menu bar.
2. Click **IM Server** in the left navigation panel (under 'Communication').
3. Click **Edit**.

Setting	Notes
Host	The address of your IM server (for example, 'chat.atlassian.com').
Port	The TCP port that your organisation uses for IM traffic (or leave this field blank to have Bamboo either perform a DNS lookup or use the default port).
Username	The login name of the IM account from which Bamboo notifications will be sent.
Change password	Select this if you have specified a username different from the currently logged-in user.
Resource	An identifying name for the connection if multiple clients use the same jabber account.

Requires a TLS/SSL connection	Select this if your IM server uses SSL.
Force legacy SSL	
Test Recipient Address	You can test this configuration by entering an address and clicking Test ,

- Click **Save**.

Screenshot: Instant Messaging server details

Update Instant Messaging Server Details

Update Instant Messaging Server Details

Update the details of the instant messaging Server in Bamboo, then click Save. Currently only XMPP (such as Jabber, Openfire) is supported.

Host*
For example "chat.myserver.com".

Port
If no port is specified, Bamboo will first perform a DNS SRV lookup or use the default port.

Username

☐ Change password?

Resource
Name of resource used to distinguish connections if multiple clients connect to the same jabber account.
For example "bamboo01.myserver.com".

☐ Requires an TLS/SSL connection

Test Instant Messaging Server Configuration

Enter recipient addresses below. Bamboo will test whether this instant messenger server setting is valid by sending a test message to the specified recipient(s).

Test Recipient Address (optional)
You can enter in one (or more, comma separated) instant messaging address to which Bamboo will send a test instant message.

[Cancel](#)

Configuring Bamboo to use Google Talk for Instant Messaging

If your Bamboo server has access to the internet, it can use [Google Talk](#) to send IM notifications about build results.

Related pages:

- [Configuring notifications for a plan and its jobs](#)
- [Working with Instant Messenger \(IM\) Notifications](#)

Before you begin:

- Google Talk does not allow IM messages to be received unless the receiver has approved the sender. Please ensure that the Gmail user specified below is approved by each Google Talk recipient. That is, ensure that the 'Host' and 'Username' have previously sent messages to each other via Google Talk.
- The Google Talk service is hosted at talk.google.com. The default port is 5222. (Note: be aware that your firewall might be blocking traffic to this port.)
- TLS is required.
- The only supported authentication mechanism is SASL PLAIN. For additional information, please see: http://code.google.com/apis/talk/open_communications.html

To configure Bamboo to use Google Talk for Instant Messaging:

- Click **Administration** in the top menu bar.
- Click **IM Server** in the left navigation panel (under 'Communication').
- Click **Edit**. Modify the settings as required.

Setting	Notes
Host	Type 'talk.google.com'. (If your IM Server uses an '@googlemail.com' account, type 'googlemail.com'.)
Port	Leave blank. Bamboo will perform a DNS lookup to figure out which port to use.
Username	Type the login name of the Google account from which IM notifications will be sent. Starting with Bamboo 3.4, you need to include the domain name, e.g. <i>atlassianbamboo@gmail.com</i> NOT <i>atlassianbamboo</i> .
Change password	Select this if you have specified a username different from the currently logged-in user.
Resource	An identifying name for the connection if multiple clients use the same jabber account.
Requires a TLS/SSL connection	Select this.
Force legacy SSL	
Test Recipient Address	You can test this configuration by entering an address and clicking Test ,

4. Click **Save**.

Modifying notification templates

If you want to customise the layout and content of your Bamboo notifications, you can customise the templates for each of the notification types (i.e. HTML email, text email, instant message) and events (e.g. Build Commented). The notification templates are written in [Freemarker](#).

Some content in notifications can also be configured via system properties, such as the number of log lines to include in email notifications that display log information.

 Changes to notification templates only take effect after a Bamboo restart.

On this page:

- [Modifying a notification template](#)
- [Configuring notifications content via system properties](#)
- [Notes](#)

Related pages:

- [Configuring Bamboo to use Instant Messaging](#)

Modifying a notification template

To modify a notification template:

1. Locate the default notification templates in your Bamboo distribution in **WEB-INF/classes/notification-templates/**
2. Copy the notification template that you wish to modify into the **templates/notification-templates** folder of your [Bamboo home directory](#), e.g. `HOME/templates/notification-templates`
 The filename will have formatted as: **<event name><notification type>.ftl**, e.g. `AfterXFailedHTMLEmail.ftl`
3. Modify the copied template, as desired. Please see the section on [Working with Freemarker](#) below for tips on updating templates.
4. Save your changes to the template. You need to restart your Bamboo server for the template changes to take effect.

Working with Freemarker

The Bamboo notification templates are written in [Freemarker](#). The Freemarker engine allows for dynamic content generation based on the Freemarker markup tags that are used in templates. This document does not describe the Freemarker language in detail. Please see the [Freemarker Online Manual](#) for full information on using this markup language.

Generating content via Freemarker involves merging a template (*.ftl file) with a context map. You can access any data in the context map from within the template using the Freemarker markup. For the notifications we have provided a specific subset of Bamboo objects that you can access. For example,

```
[#if buildSummary.successful]
${buildSummary.buildResultKey} was successful.
```

If you had a successful Bamboo build with build result, BAM-1234-1, the above markup would return the following text in your notification:
BAM-1234-1 was successful.

You can find more information on working with Freemarker, including Bamboo objects available from Freemarker templates, tips on writing

Freemarker templates and examples in the [Freemarker and notification templates](#) document.

Configuring notifications content via system properties

The following system properties can be configured to control some of the content that is included in notifications (e.g. the number of log lines to include in email notifications that display log information). For instructions on how to configure a system property, please refer to the [Configuring system properties](#) page.

Before you begin:

The system properties below do not add content to notifications. You still need to ensure that your notification templates contain the relevant entities to display the content. For example, changing the `bamboo.notifications.logLinesToInclude` property will not add log information to your notifications. It only modifies the number of log lines displayed in notification templates that already include logs.

System Property	Description	Default Value
<code>bamboo.notifications.logLinesToInclude</code>	Specifies the number of log lines to include in email notifications that display log information.	100

Notes

- **Bamboo does not validate notification templates.** If you have incorrectly formatted the markup text in the template, Bamboo will still use the template to send out notifications. If this happens, your users may receive notifications with unreadable or missing information, as well as error messages. Errors will also be posted to your logs.

Freemarker and notification templates

Notification templates in Bamboo can be modified to customise the format and content of your notifications. The templates are written in [Freemarker](#). This page is intended to complement the [Modifying notification templates](#) page and contains information on the Bamboo objects available from Freemarker templates, tips on writing Freemarker templates and examples.

 Changes to notification templates only take effect after a Bamboo restart.

On this page:

- [Accessing Bamboo data](#)
- [Special considerations when working with Freemarker](#)
- [Freemarker examples](#)

Related pages:

- [Configuring Bamboo to use Instant Messaging](#)
- [Modifying notification templates](#)

Accessing Bamboo data

Each individual notification has a different subset of data that can be accessed from the Freemarker templates. You can find information on the objects available in our javadocs below.

- [Build Completed Notification](#) ("All Completed Builds" and "Failed and First Success")
- [After X Failed Builds Notification](#)
- [Build Commented Notification](#)
- [Build Hung Notification](#)
- [Build Error Notification](#)

Special considerations when working with Freemarker

Never assume data exists

Unfortunately Freemarker is not very forgiving if data does not exist or is null. Hence, you will need to check whether information exists before adding it to a page. The sample code below shows how you can validate for non-existent data.

```
[#if issue.jiraIssueDetails.summary?has_content][/#if]
[#if issue.jiraIssueDetails.summary??][/#if]
${issue.jiraIssueDetails.summary?if_exists}
${issue.jiraIssueDetails.summary!}
```

Check the encoding of your information

Freemarker has built-in utilities for escaping special characters. These could be characters that you deliberately do not want to be interpreted as HTML, or data that could potentially contain malicious content. The sample code below shows how you can escape characters in Freemarker.

```
${commit.comment?html} // for data to be encoded to be displayed as html
${commit.author?url} // for data to be encoded for a url
```

You can find more information on these utilities in the [official Freemarker documentation](#).
Use white space carefully

When editing text email content and instant message content, you need to be very careful with spacing and line breaks. Any spaces and line breaks that you have entered in the Freemarker template will also exist in the evaluated content. Freemarker provides you with some utilities to remove white space, so that you can still retain some formatting in the templates.

More information can be found the [official Freemarker documentation](#).

Freemarker examples

Below are some raw examples of additional information that you can add to your emails.

Please note, these examples are intended to demonstrate the use of Freemarker and how to access Bamboo objects. You will need to modify these examples to include your desired formatting and make it work with your data.
List files in a commit

```
[#if buildSummary.commits.size() > 0]
  [#list buildSummary.commits as commit]
    [#if commit_index gte 3][#break][/#if] //only shows 3 commits
    Author:  <a href="[@ui.displayAuthorOrProfileLink commit.author/]">
              ${commit.author.fullName?html}
            </a>
            <br/>
    Comment:  ${commit.comment?html}
            <br/>
    [#if commit.guessChangeSetId()?has_content]
    Revision: ${commit.guessChangeSetId()?html}
    <br/>
  [/#if]

  [#if commit.files?has_content]
  Files:
    [#list commit.files as file]
      ${file.cleanName} [#if
file.revision?has_content](${file.revision})
      <br/>
    [/#list]
  [/#if]
[/#list]
[#else]
  This build does not have any commits.
[/#if]
```

Provide test error details

```
[#list buildResults.testResults.newFailedTests.values() as testResultClass]
  [#list testResultClass.testResults as testResult]
    <a href="${baseUrl}${fn.getViewTestClassResultUrl(build.key,
buildResults.buildNumber, testResultClass.name)}">
      ${testResultClass.shortName?html}
    </a> :
    <a
href="${baseUrl}${fn.getViewTestCaseHistoryUrl(buildSummary.buildResultKey,
testResult.className,
      testResult.actualMethodName)}">
      ${testResult.methodName?html}
    </a>
  <br/>
  [#if testResult.errors?has_content]
    [#list testResult.errors as error]
      <pre>${error.errorMessage}</pre> // a <pre/> tag is
required to reserve formatting of error
    [#list]
  [#if]
[#list]
[#list]
```

Working with Instant Messenger (IM) notifications

Bamboo can send you notifications about [build results](#) for a particular plan. Each plan's recipients are specified by a Bamboo administrator, but you can choose whether you would like to receive your Bamboo notifications via email and/or instant messenger (IM). See [Changing your notification preferences](#).

As well as receiving IM notifications, you can interact with Bamboo using IM, as described on this page.

On this page:

- [Labelling a build result using IM](#)
- [Commenting about a build result using IM](#)

Related pages:

- [Working with labels](#)
- [Configuring Bamboo to use Instant Messaging](#)
- [Getting feedback](#)

Labelling a build result using IM

To label a build result using IM:

In your Instant Messenger client, type your comment in the following format:

```
label [build key] <labels>
```

 Entering a build key is optional. If none is specified, Bamboo will look up the last time it corresponded with you and the build that was in context. The context gets updated when you specify a build key in your command, and when Bamboo sends you a notification about a particular build.

Commenting about a build result using IM

To comment on a build result using IM:

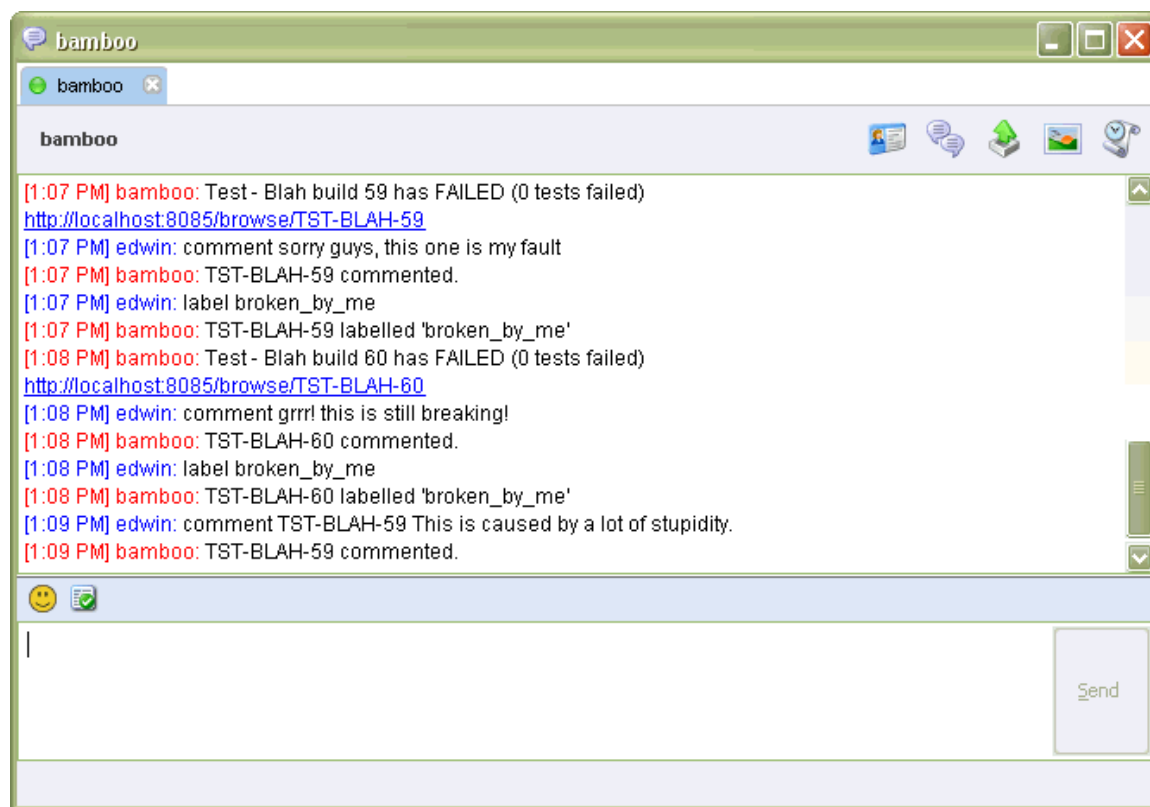
In your Instant Messenger client, type your comment in the following format:

```
comment [build key] <comment message>
```

 Entering a build key is optional. If none is specified, Bamboo will look up the last time it corresponded with you and the build that was in

context. The context gets updated when you specify a build key in your command, and when Bamboo sends you a notification about a particular build.

Screenshot: Interacting with Bamboo using IM



Subscribing to RSS feeds

Bamboo aggregates key information about your builds into RSS feeds. You can subscribe to these feeds using any feed reader.

RSS feed scope	Options	Set up
All plans	- All build results - Failed build results	- Go to the Dashboard's All Plans tab. - Locate the RSS icon at the bottom of the screen. - Right-click either all builds or all failed builds, and copy its URL. - Paste the URL into your RSS reader.
A specific plan	- All build results - Failed build results	- Go to the plan. - Locate the RSS icon at the bottom of the screen. - Right-click either all builds or all failed builds, and copy its URL. - Paste the URL into your RSS reader.
Build results with a particular label		- Go to the Dashboard. - Go to any plan that has a label (this may involve trial and error). - Click on any label, near the top of the screen. - Click All Labels. - Click the label of interest. - Locate the RSS icon at the bottom of the screen. - Right-click Feed for builds labelled a and copy its URL. - Paste the URL into your RSS reader.

Reporting

You are able to get reports about various kinds of activity in Bamboo:

Summary statistics for all users	A list of summary build statistics for all Bamboo users, showing such things as the number of builds triggered, broken and fixed. See Viewing build statistics for all users .
Build results for an author	Build results summaries for someone who has committed code to projects in Bamboo, including the last 10 builds, the last 10 broken and the last 10 fixed. See Viewing build results for an author .
Comparison charts for authors	Create comparison charts of build activity for selected authors. See Generating reports on selected authors
Comparison charts for plans	Create comparison charts of build results for selected plans. See Generating reports across multiple plans
Clover code-coverage reports	See Viewing the Clover code-coverage for a plan See Viewing the Clover code-coverage for a build

Viewing build statistics for all users

The build statistics summary gives you an overview of the activity of Bamboo users.

To view summary statistics for all users:

1. Click **Authors** in the top menu bar.
2. Click the **List Users** tab.

Related pages:

- [Reporting](#)
- [Viewing build results for an author](#)

Screenshot: Users Summary

User and Author Statistics

List UsersList AuthorsStatistics

Users

A listing of all authors who commit to projects in Bamboo. *Broken* means the build has failed but the previous build was successful. *Fixed* means that the build was successful but the previous build has failed. The *Score* is a difference of fixed and broken builds.

Name	Triggered	Failed	% Failed	Broken	Fixed	Score
Adrian Deccico	0	0	0%	0	0	0
Adrian Hempel	589	156	26%	60	56	-4
Agnes Ro	0	0	0%	0	0	0
Alex Wei	1	0	0%	0	0	0
Andrew Lynch	0	0	0%	0	0	0
Anna Buttfield	3	2	67%	0	0	0
Arun Bhalla	37	16	43%	5	2	-3
Belinda Teh	396	131	33%	29	21	-8
Ben Woskow	182	91	50%	17	7	-10

Viewing build results for an author

An *author* is any person who checks in code to a repository that is associated with a Bamboo [plan](#). An author need not be a Bamboo user. An author's source-code repository login must have been [associated with their Bamboo user profile](#) before you can see their build results in Bamboo.

To view build results for a particular author:

1. Click **Authors** in the top menu bar.
2. Click the **List Authors** tab.

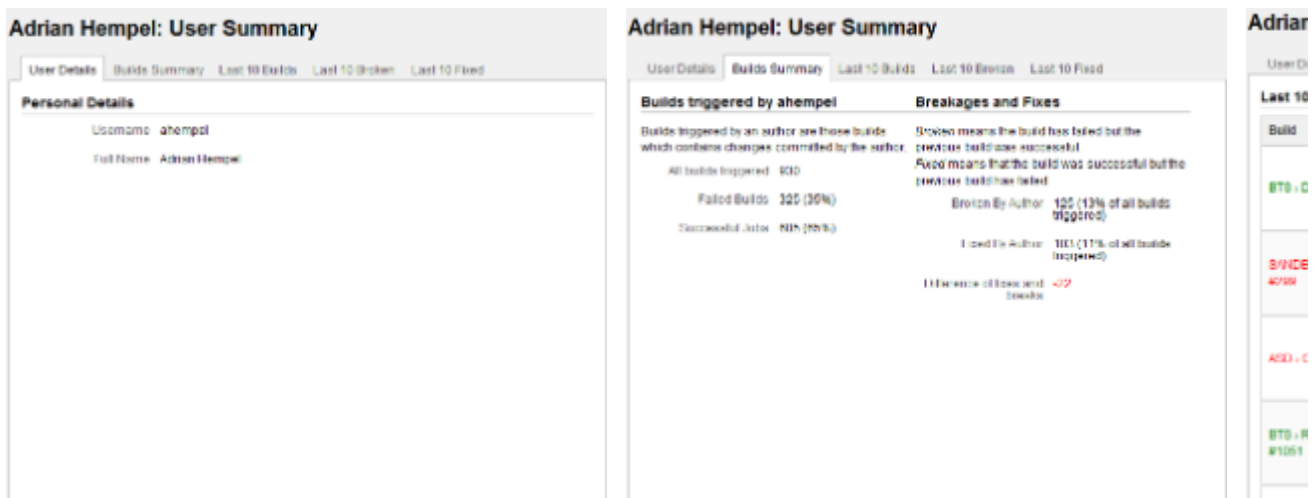
3. Click an author's name to see statistics and recent build results for the author:

User Details	The author's user details.
Builds Summary	A statistical summary of all the author's builds.
Last 10 Builds	The last 10 builds that were triggered by this author.
Last 10 Broken	The last 10 builds that were triggered by this author, where the build failed and the previous build for the same plan was successful.
Last 10 Fixed	The last 10 builds that were triggered by this author, where the build was successful and the previous build for the same plan failed.

Related pages:

- [Viewing build statistics for all users](#)

Screenshots: Viewing an author's build results (click to view larger images)



1. User Details

2. Build Summary

Generating reports on selected authors

An *author* is any person who checks in code to a repository that is associated with a Bamboo [plan](#). An author need not be a Bamboo user.

Generating a report on selected authors

To generate a report on selected authors:

1. Click **Authors** in the top menu bar.
2. Click the **Statistics** tab.
3. Set the report parameters:

Parameter	Notes
Report	Choose from the available reports, described below. Additional reports may have been configured by your Bamboo administrator.
Authors	Choose the authors on whom you want to report. Use the <Ctrl> key to select multiple authors.
Group By	Choose the time scale for the horizontal axis.

4. Click **Submit**.

On this page:

- [Generating a report on selected authors](#)
- [Selected author report types](#)

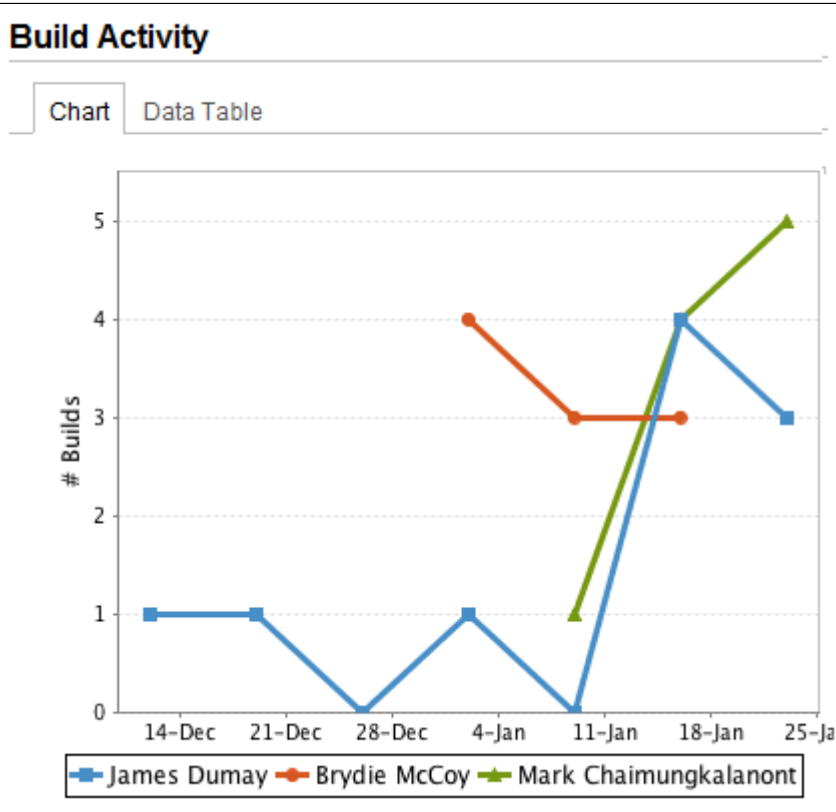
Related pages:

- [Viewing build results for an author](#)
- [Getting feedback](#)
- [Notifications](#)

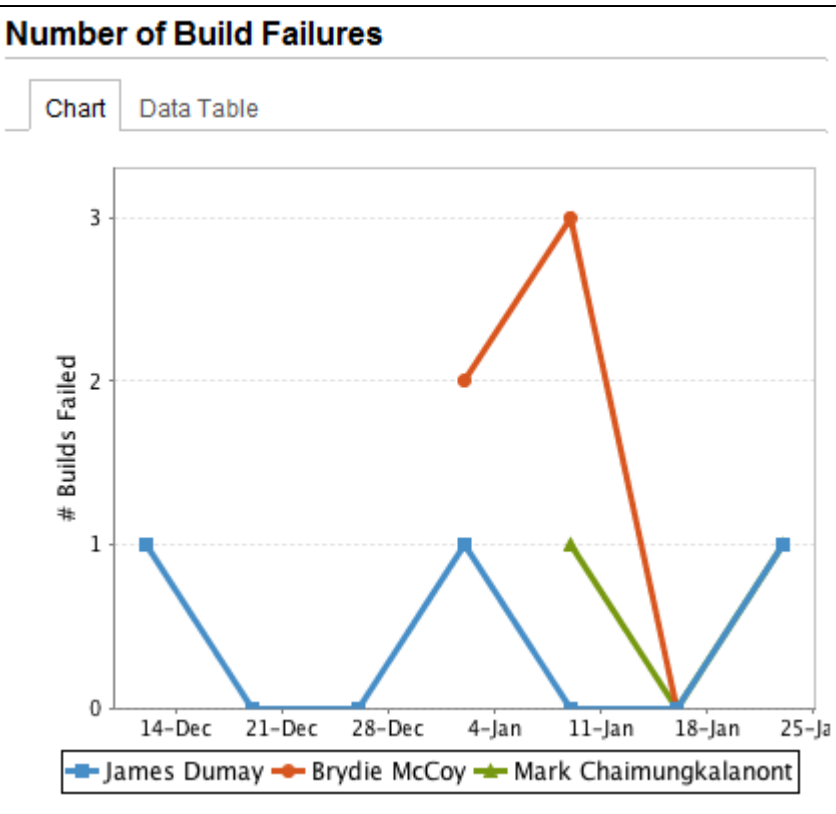
Selected author report types

The following standard report types are available.

Build activity



Number of build failures

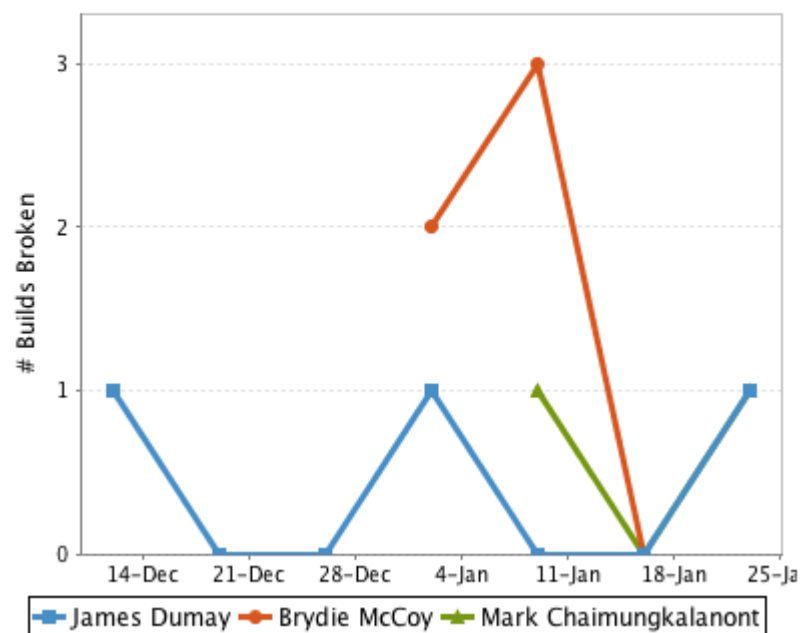


Number of builds broken

Number of Builds Broken

Chart

Data Table

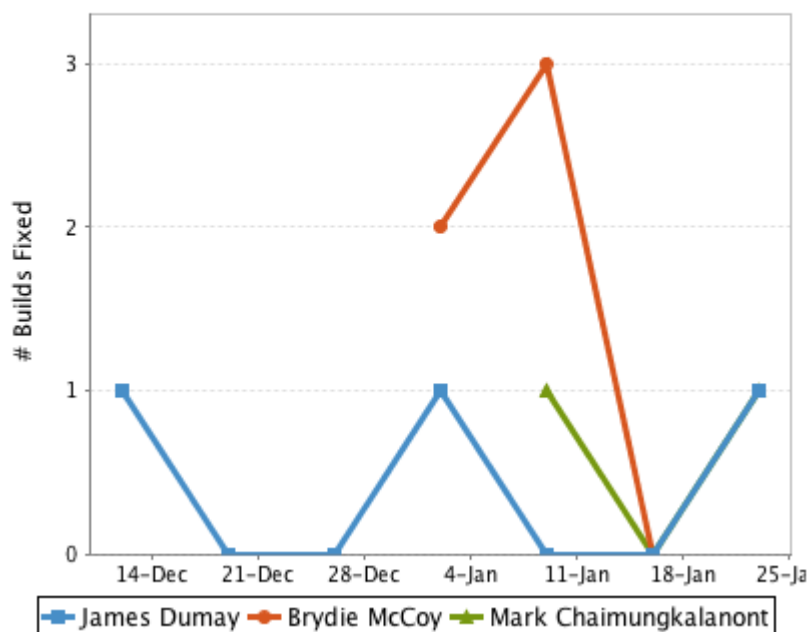


Number of builds fixed

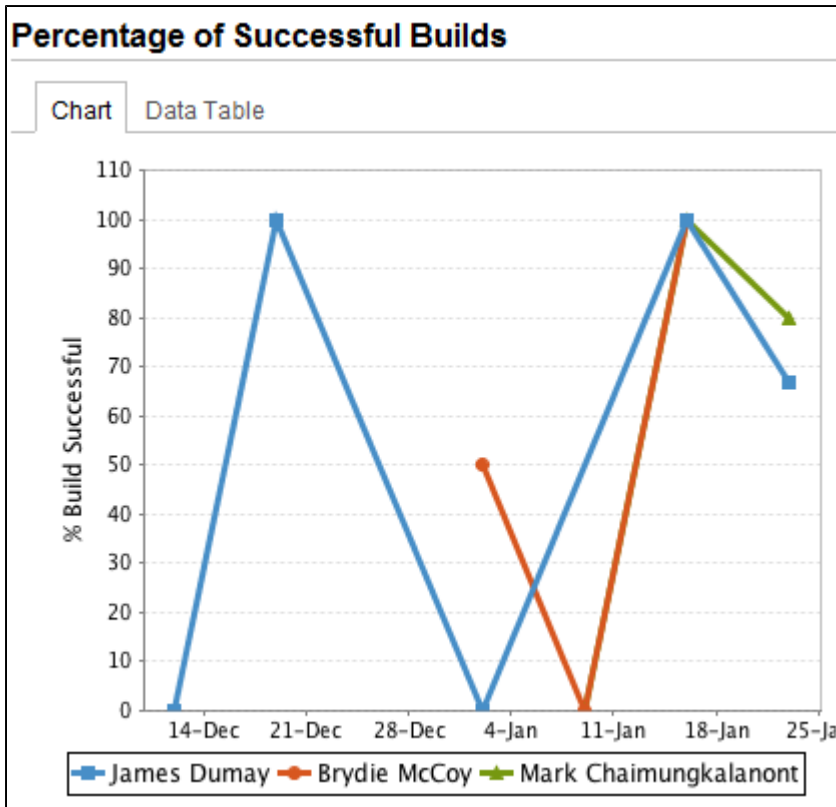
Number of Builds Fixed

Chart

Data Table



Percentage of successful builds



Generating reports across multiple plans

Bamboo provides a report generator that enables you to compare build statistics across one or more plans, using a variety of different metrics.

Generating plan reports

To report on build statistics per plan:

1. Click **Reports** in the top menu bar.
2. Set the report parameters:

Parameter	Notes
Report	Choose from the available reports, shown below. Additional reports may have been configured by your Bamboo administrator.
Build plans	Choose the plans on which you want to report. You can use the <Ctrl> key to select multiple plans.
Group By	Choose the time scale for the horizontal axis.
Date Filter	Choose the time period on which to report. Use Select Range to set a custom range.

3. Click **Submit**.

On this page:

- [Generating plan reports](#)
- [Plan report types](#)
 - [Build activity](#)
 - [Build duration](#)
 - [Percentage of successful builds](#)
 - [Time to fix](#)
 - [Number of tests](#)
 - [Number of build failures](#)
 - [Clover lines of code](#)
 - [Clover code coverage](#)

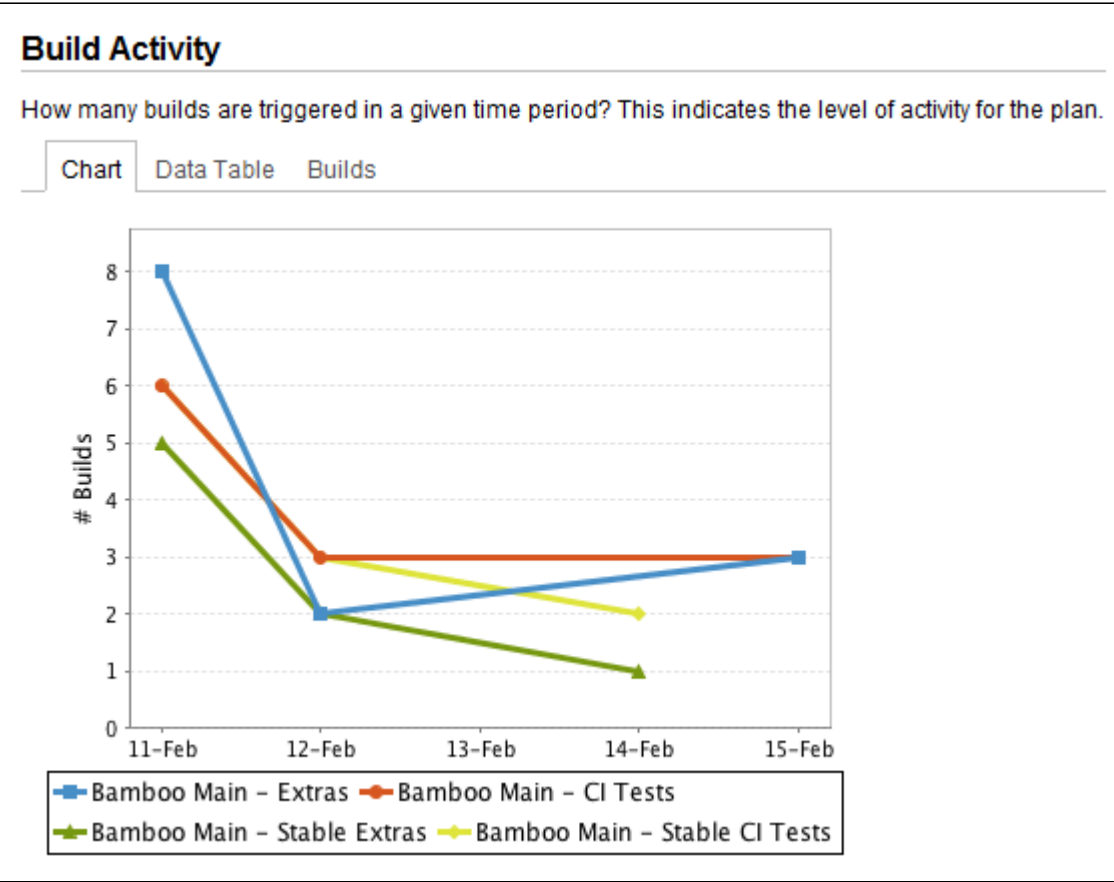
Related pages:

- [Generating reports on selected authors](#)
- [Viewing build results for an author](#)
- [Getting feedback](#)

Plan report types

Some of the standard plan report types are illustrated below.

Build activity



Build duration

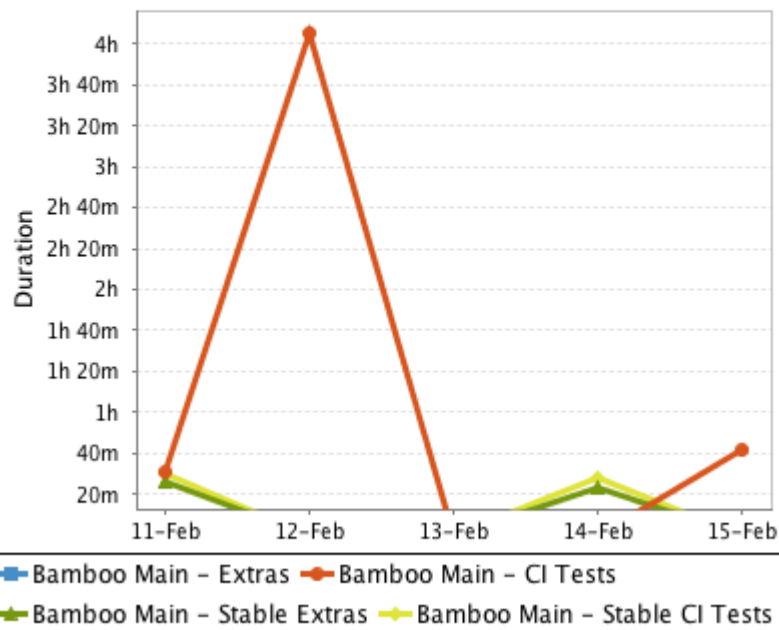
Build Duration

The report shows how long your build takes over time. Is it getting slower or faster?

Chart

Data Table

Builds



Percentage of successful builds

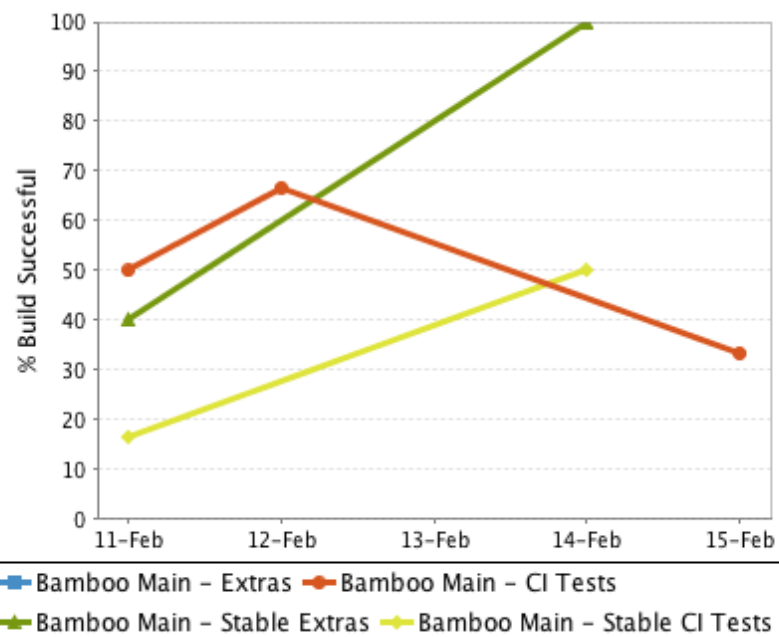
Percentage of Successful Builds

Comparing success percentages gives you an idea of how stable a plan is compared to one another. 100% means your plan is always rock solid. 0% means something is seriously wrong.

Chart

Data Table

Builds



Time to fix

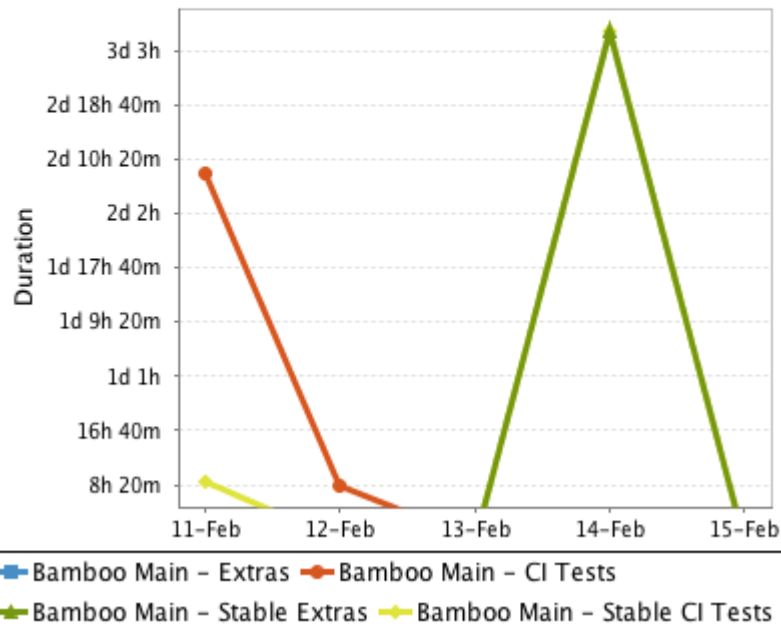
Time to Fix

How long does it take on average to fix problems? This provides an indication of how quickly breakages are resolved for the plan.

Chart

Data Table

Builds



Number of tests

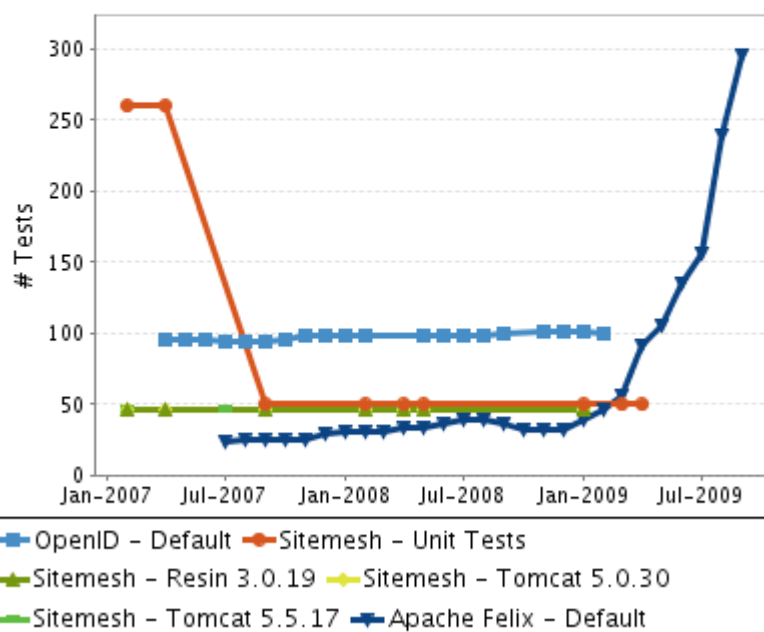
Number of Tests

How many tests does your build have? This provides a rough indication of the level of testing over time for the plan.

Chart

Data Table

Builds



Number of build failures

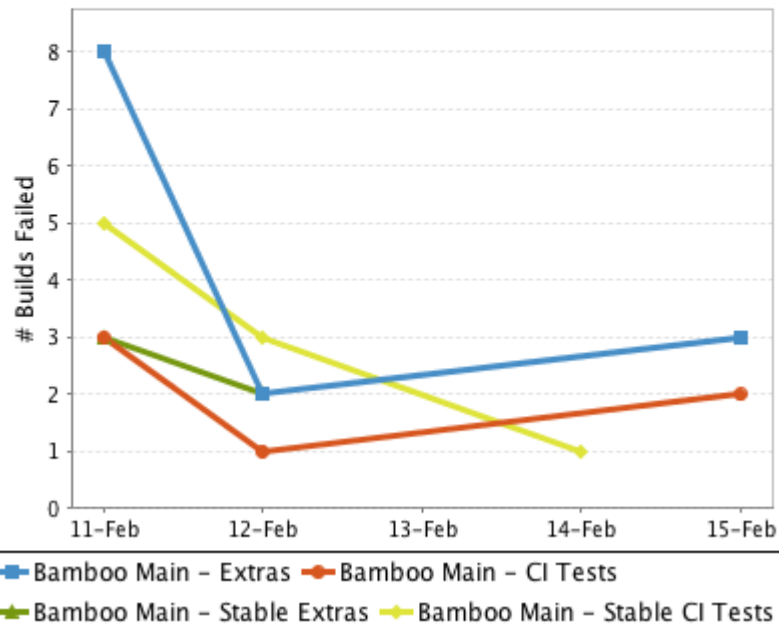
Number of Build Failures

How many builds are being broken? A high value indicates a relatively unstable plan that tends to be broken often.

Chart

Data Table

Builds



Clover lines of code

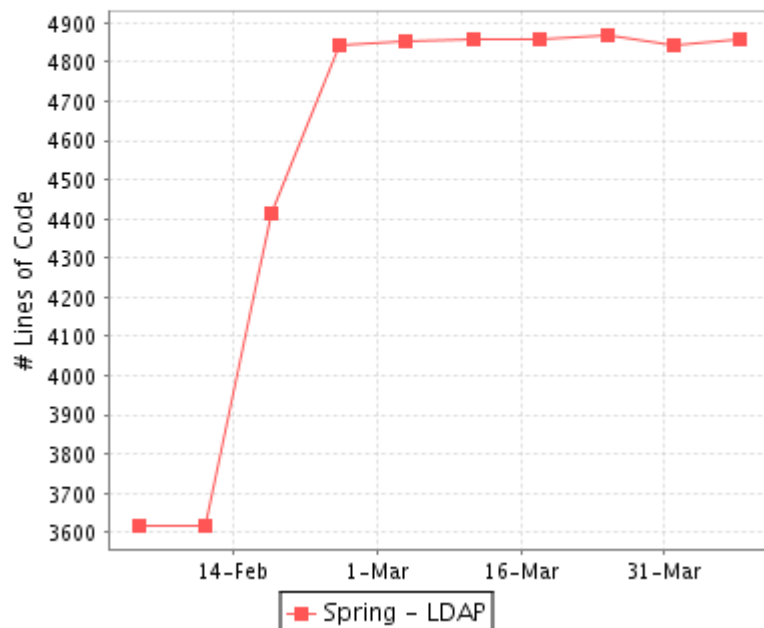
Clover Lines of Code

Provides an indication of the size of the code base for the build.

Chart

Data Table

Builds



Clover code coverage

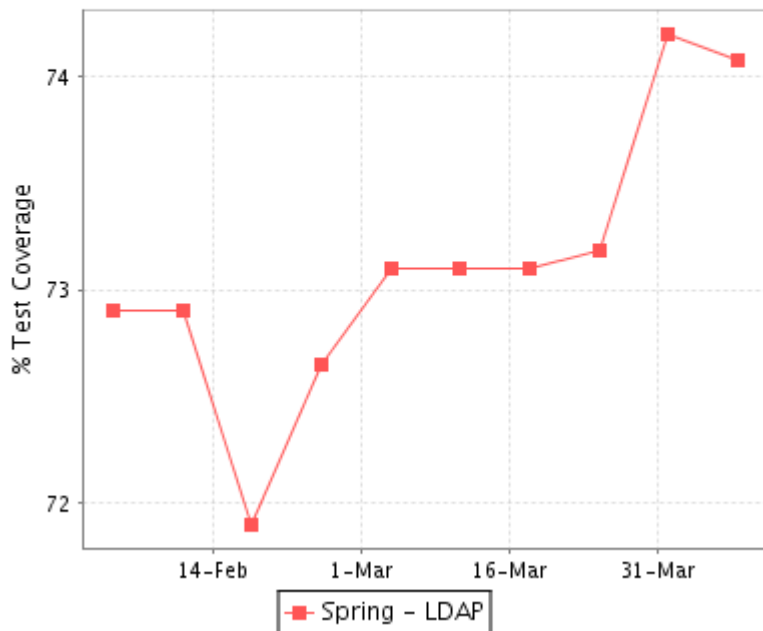
Clover Code Coverage

Comparing code coverage gives you an idea of how well the code base is tested. 100% coverage means that all code elements have been covered by your tests.

Chart

Data Table

Builds



Viewing the Clover code-coverage for a plan

If you use Atlassian's [Clover](#) and your job specifies a Clover directory (see [Enabling the Clover add-on](#)), you will be able to view the Clover coverage summary for the plan.

Related pages:

- [Enabling the Clover add-on](#)
- [Viewing the Clover code-coverage for a build](#)
- [Generating reports across multiple plans](#)

To view the Clover coverage summary for a plan:

1. Locate the plan summary on the Bamboo Dashboard
2. Click on the **Clover** tab.

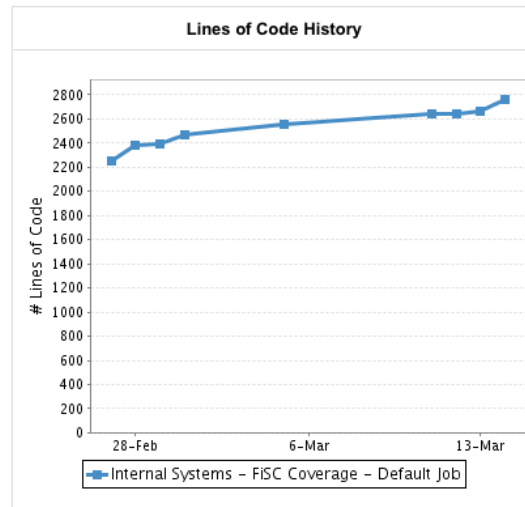
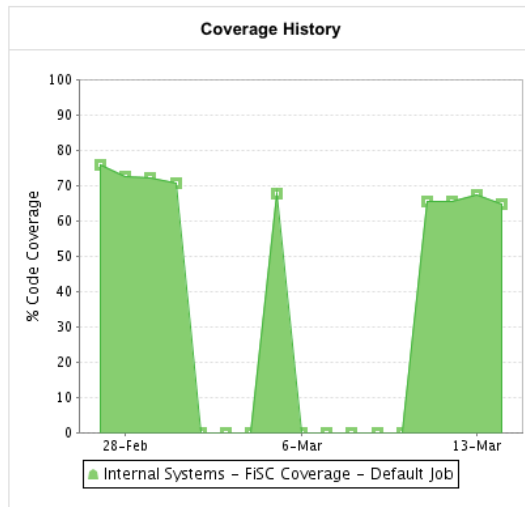
The Code Coverage Summary information will display. It will include:

- Latest coverage from the most recent build as a percentage and bar representation
- A link to a detailed HTML report
- Coverage History chart showing changes in [percentage Code Coverage](#) over time
- Lines of Code History chart showing changes in [LOC](#) over time

Code Coverage Summary

65%

Latest coverage

 [View latest Clover HTML report from #61 \(12:26 AM, Wed, 14 Mar 2012\)](#)**Note:**

1. Charts are only generated when build results from at least a 2-day span are available. Where shorter time spans are available, the user will receive a warning stating "Insufficient data in range to draw the chart."
2. Where your plan contains multiple jobs with Clover, then Code Coverage and Lines of Code values are aggregated from all these jobs.
3. If your plan contains multiple jobs with Clover, the **View latest Clover HTML report** link will point to the default job only. In order to see other reports, you must go to the specific job summary, as described in [Viewing the Clover code-coverage for a build](#).

Viewing the Clover code-coverage for a build

If your organisation uses the [Atlassian Clover](#) code-coverage tool, Bamboo can record code-coverage details (i.e. the percentage of code covered by tests) for each build result.

This is only available if the build's plan specifies a Clover directory (for details please refer to the [Enabling the Clover add-on](#)).

Bamboo also provides data on code-coverage trends for a plan over a period of time. For details see the *Related pages* at right.

Related pages:

- [Working with build results](#)
- [Enabling the Clover add-on](#)
- [Generating reports across multiple plans](#)

Clover HTML report for a job

Where Clover generates an HTML report (created by default in automatic integration), you can examine the report in the build job summary page. To view the report:

1. Go to the plan summary
2. Select the relevant build number
3. Select the appropriate job
4. Click on the **Clover** tab to open the report

Clover
Clover Maven Autointegration
#1

#1

Job: Default Job was successful

Stages & Jobs

Default Stage

Default Job

Job Summary

Tests

Changes

Artifacts

Logs

Clover

Metadata

Clover Code Coverage

Money Test 1.0-SNAPSHOT Clover Coverage Report

Dashboard
Coverage Reports
Coverage (Aggregate)
Test Code (Aggregate)
Test Results

Application Packages

com.cenqua.samples.money (93%)

Clover Coverage Report - Money Test 1.0-SNAPSHOT

Coverage timestamp: Pt sty 11 2013 08:53:28 CET

Overview
Package
File

FRAMES NO FRAMES SHOW HELP

Coverage 3 classes, 160 / 172 elements

93%

Test 100%

The **Clover** tab is not available on the Build summary page - you must navigate to the Job summary. This is because your build may contain multiple jobs, each of which may have its own Clover report.

Clover statistics report for a job

If your build generates a Clover XML report but not the HTML report, then the Clover Report artifact is not available on the **Artifacts** tab, however the build job summary page will contain a few code coverage statistics:

1. Go to the plan summary
2. Select the relevant build number
3. Select the appropriate job
4. Click on the **Clover** tab to open the report

Clover
Clover Maven Manual Integration
#7

#7

Job: Default Job was successful

Stages & Jobs

Default Stage

Default Job

Job Summary

Tests

Changes

Artifacts

Logs

Clover

Metadata

Clover Code Coverage

Test Coverage

Build Coverage 93,0% (0,0%)

The overall percentage of code base in this plan that is covered by tests. The second value in brackets is the change in coverage percentage since the previous successful build.

Statement Coverage 94,7%

The percentage of statements covered by tests

Methods 100,0%

The percentage of methods covered by tests.

Conditionals 84,8%

The percentage of conditional branches covered by tests.

Build Statistics

LOC 287

The lines of code in the build.

NCLOC 183

The non-comment lines of code in the build.

Files 3

Methods 31

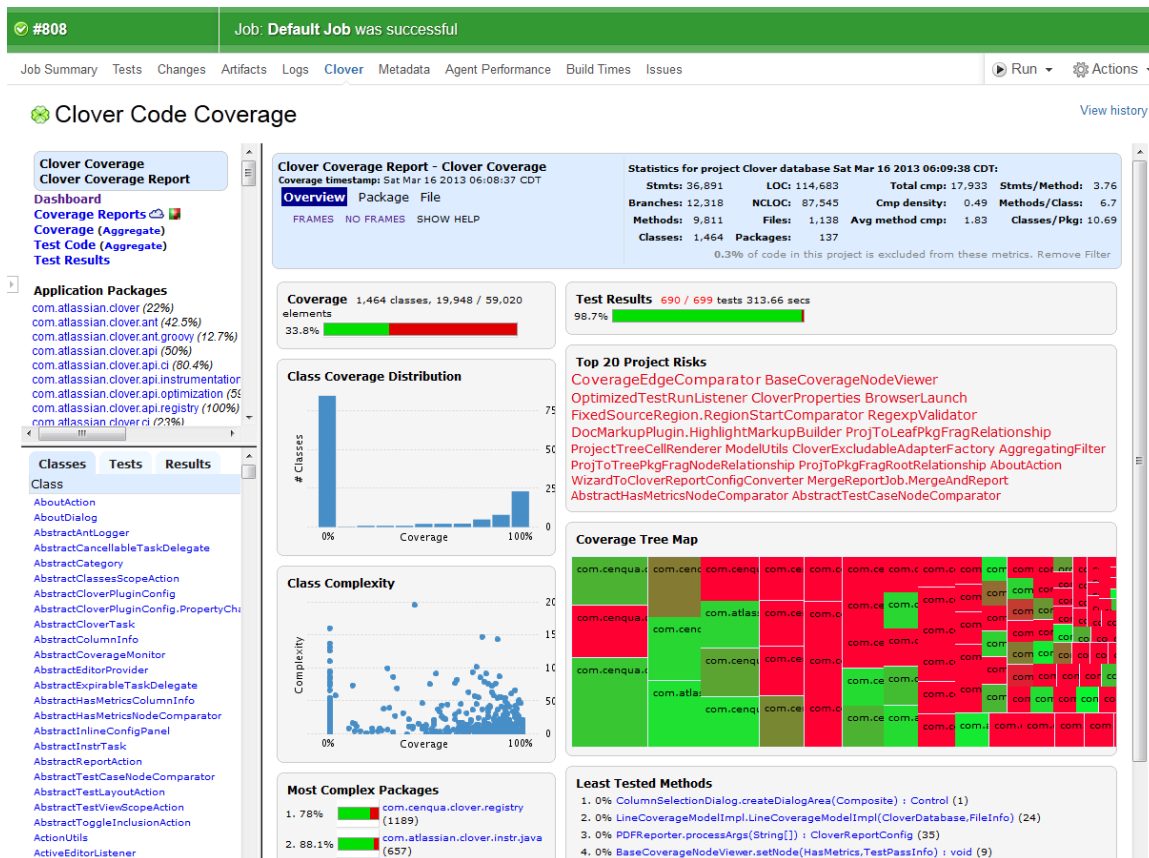
Classes 3

Packages 1

TIP: This usually happens for manual Clover integration. In case you want to see full Clover report, configure it as described on [Enabling the Clover add-on](#) page.

References

The content of the Clover HTML report is discussed in detail on the [Clover Documentation Home - 4. Understanding Reports](#) page. For completeness, an example Clover Code Coverage HTML report is shown below.



	When Bamboo is integrated with FishEye, you can: • view the code changes that triggered a build • explore a failed build in FishEye and jump directly into the changeset that broke the build • view the history of the changeset to see what the author was trying to fix • analyze the change using the side-by-side diff view • open the relevant files in your IDE.
	When Bamboo is integrated with Clover, you can: • View code-coverage details (i.e. the percentage of code covered by tests) for each build result • View code-coverage trends for a job over a period of time • View the code-coverage summary for the job.
	When Bamboo is integrated with HipChat, you can get notifications from Bamboo for things like: • when a build passes or fails • when you are assigned responsibility for a breaking build • when a build you are responsible for has been fixed • when a manual stage of a build is ready to be run ...and many other notification events.

See [The big list of Atlassian gadgets](#).

Integrating Bamboo with JIRA

Integrating Bamboo with Atlassian's [JIRA](#) combines Bamboo's continuous integration capabilities with your issue tracker to give you a unified view of your software development project.

Using JIRA and Bamboo together, you can:

- create actionable JIRA issues from a Bamboo build
- run a Bamboo build when [releasing a JIRA version](#)
- have Bamboo automatically [link a plan branch](#) with a JIRA issue
- view the JIRA issues linked to a [build result](#)
- view the Bamboo builds that relate to a [particular JIRA issue](#)
- view the Bamboo builds that relate to a [JIRA project](#) or [version](#)
- add Bamboo gadgets to a JIRA dashboard.

Configuring Bamboo and JIRA to work together simply requires you to set up an application link (two-way) between JIRA and Bamboo. By the way, application links have nothing to do with using JIRA as a user repository for Bamboo; these 2 configurations can exist separately.

On this page:

- [Before you begin](#)
- [Set up an application link](#)
- [Try your new configuration](#)
- [Notes](#)

Related pages:

- [Configuring Application Links](#) (JIRA documentation)

Before you begin

Version Requirements

Application	Requirement	Notes
Bamboo	Version 3.1 or later	If you are using an earlier version of Bamboo, please refer to these instructions in the corresponding older version of the Bamboo documentation.
JIRA	Version 4.4 or later	If you are using an earlier version of JIRA, please refer to the Bamboo 3.0 instructions: Integrating Bamboo with JIRA (Bamboo 3.0) .

If you are using an earlier version of Bamboo and/or JIRA, you can also download an older version of the JIRA Bamboo plugin from the [Atlassian Plugin Exchange](#). However, we strongly advise you to upgrade JIRA to version 4.4 or later and Bamboo to version 3.1 or later, if you wish to integrate Bamboo with JIRA.

Set up an application link

Before you begin:

- **Security Considerations** — The instructions below recommend setting up authentication for the application link between JIRA and Bamboo. Please ensure that you read the [Security Implications for each Authentication Type](#) (JIRA documentation). For example, if you use basic HTTP authentication for the JIRA to Bamboo link, you must specify a user that JIRA uses to log in to Bamboo. Hence, this user's Bamboo permissions will be used (not the Bamboo permissions of the user who is currently logged into JIRA), e.g. a user [viewing Bamboo information in JIRA](#) will see all builds available to the user specified in the JIRA-Bamboo setup, instead of the builds available under their own permissions.

Follow the [JIRA instructions](#) to configure the application link in JIRA.

- You will need to set up a two-way link, i.e. select the 'Create a link back to this server' option when adding the application link.
- You will need to configure either OAuth or Trusted Apps authentication for your application link. See [Configuring Authentication for an Application Link](#) for instructions.

Note that if you are running Bamboo behind a proxy, you may need to [configure the AJP connector](#).

Congratulations! You have successfully integrated Bamboo and JIRA.

Try your new configuration

You may wish to read about how to use these two applications together in the following pages:

- View the Bamboo builds that relate to a [particular JIRA issue](#). If you are able to see your JIRA issues from a Bamboo build, but cannot see your Bamboo builds in JIRA (on the **Builds** tab), please make sure that you have "View Version Control" permission enabled.
- View the Bamboo builds that relate to a [JIRA project](#) or [version](#).
- View the JIRA issues for a [build result](#).
- Add JIRA gadgets to [display the status of your builds](#) or a [graphical summary of each build plan](#) (please note, if you have added multiple Bamboo servers there will be one set of these Bamboo gadgets available for each server, e.g. 'Bamboo Status Gadget from <http://172.20.5.83:8085>').
- [Trigger Bamboo builds when releasing JIRA versions](#) (*JIRA documentation*).

Notes

What if the Bamboo gadgets do not appear in JIRA?

If the Bamboo gadgets do not appear in your JIRA gadget directory, you will need to subscribe to Bamboo's gadgets in JIRA.

To subscribe to Bamboo's gadgets in JIRA:

1. Go to your JIRA dashboard.
2. Click **Add Gadget**.
3. In the 'Gadget Directory' dialog, click **Gadget Subscriptions**.
4. In the 'Gadget Subscriptions' dialog, click **Add Subscription**.
5. In the 'Add Subscriptions' dialog, copy the base URL for your Bamboo site (e.g. <http://www.foobar.com:8085>) and paste it into the text box on the screen.
6. Click **Add Subscription**.
7. Click **Finished**.

Known issues

Deploying multiple Atlassian applications in a single Tomcat container is **not supported**. We do not test this configuration and upgrading any of the applications (even for point releases) is likely to break it. There are also a number of known issues with this configuration (see [this FAQ](#) for more information).

We also do not support deploying multiple Atlassian applications to a single Tomcat container for a number of practical reasons. Firstly, you must shut down Tomcat to upgrade any application and secondly, if one application crashes, the other applications running in that Tomcat container will be inaccessible.

Finally, we recommend not deploying *any other applications* to the same Tomcat container that runs Bamboo, especially if these other applications have large memory requirements or require additional libraries in Tomcat's `lib` subdirectory.

 *JIRA and Bamboo cannot run in the same Tomcat instance due to a known issue with the Bamboo plugin for JIRA (see [JRA-19662](#)).*

 *If integrating Bamboo with JIRA, you should not change the JIRA project key format from the default, as Bamboo only supports the default project key format.*

If you need further help, please raise a support request in our [support system](#), in the Bamboo project. You may also want to view articles in the [Bamboo Knowledge Base](#) and browse [our forums](#).

Viewing linked JIRA issues

If your organisation uses Atlassian's [JIRA](#) and your administrator has [integrated Bamboo with JIRA](#), you will be able to view the JIRA issues that have been linked to a build. This provides an easy way to jump to relevant issues in JIRA to see details about what the code is intended to achieve.

Linked JIRA issues can be viewed on:

- the **Issues** tab of the [Plan Summary](#) page, for all issues linked to the plan
- the [Build Result Summary](#) page, for just 2 of the issues linked to a build
- the **Issues** tab of the Build Result Summary page, for issues linked to a build.

Issue links can be created [automatically](#) by Bamboo when you specify an issue key in your build [comments](#), [labels](#) or [commit messages](#), or they can be added [manually](#).

On this page:

- [Viewing the JIRA issues linked to a plan's builds](#)
- [Viewing the JIRA issues for a build result](#)

Related pages:

- [Linking JIRA issues to a build](#)
- [Creating JIRA issues from a build](#)
- [Integrating Bamboo with JIRA](#)

Viewing the JIRA issues linked to a plan's builds

To view the JIRA issues linked to all builds for a plan:

1. Navigate to the desired plan, as described on [Configuring plans](#).
 2. Click the **Issues** tab. A list of all of the issues linked to builds for the plan are displayed, sorted by build date. You can constrain the list using the build filter (e.g. 'Showing last 25 builds') next to the tabs.
- Click the issue key to view the issue in JIRA.
 - Click the **N related builds** link (where N is a number of builds) to view the builds linked to that issue on the **Builds** tab in JIRA.

[Plan Summary](#) [Branches](#) [Recent Failures](#) [Quarantined Tests](#) [Tracking](#) **Issues** [Run](#) [Actions](#)

JIRA Issues

Showing Last 25 builds

This page lists the JIRA issues that have been linked to builds related to this plan, ordered by last built. This includes issues specified in commit messages, labels or comments for builds, as well as issues manually linked to builds.

JIRA Issues (15 issues)

Type	Key	Summary	Status	Assignee	Fix versions	Related Builds	Last Built
	BDEV-503	Plan Summary pie chart doesnt show in ie 8	To be reviewed	Jason Berry		21 related builds	Last built: 4 hours ago
	BAM-11045	Plan Variables - Descriptive text - Review if plan variables are visible to all builds or just jobs in plan	Resolved	Jesper Sarnesjo [Atlassian]	4.1.1	20 related builds	Last built: 5 hours ago
	BDEV-397	Redirect loop when clicking top-left nav bar	To be reviewed	Przemek Bruski		51 related builds	Last built: 10 hours ago
	BDEV-346	Duplicate authors in database export cause import to fail from HSQL to MySQL	In Progress	Krystian Brazulewicz		19 related builds	Last built: 17 hours ago

[Activity](#)

[All](#) [Comments](#) [Work Log](#) [History](#) [Activity](#) [Source](#) [Reviews](#) **Builds**

Tardigrade Server

Showing 25 of 29 builds.

+core+ Bamboo - CI Tests - atlassian_bamboo_4_1_branch #23
Relates to: [21 issues](#)
Code changed | Ran: 3 hours ago
Duration: 30 minutes | Tests: 2375 passed

+core+ Bamboo - Stable CI Tests #971
Relates to: [21 issues](#)
Code changed | Ran: 3 hours ago
Duration: 29 minutes | Tests: 2356 passed

+core+ Dogfooding - Bamboo Bitbucket - atlassian_bamboo_4_1_branch #23
Relates to: [21 issues](#)
Code changed | Ran: 4 hours ago
Duration: 4 minutes | Tests: No tests found

Viewing the JIRA issues for a build result

To view the JIRA issues linked to a particular build result:

1. Navigate to the build results for the plan, as described in [Viewing a build result](#).

2. Click the build number for the desired build result.

- **Build Summary** tab — the 'JIRA Issues' section displays up to two of the issues linked to the build.
- **Issues** tab — displays all of the JIRA issues linked to the build. Click **Add linked issue** to link this build to an issue in JIRA.

Screenshot: JIRA issues for a build result — Build Summary tab

Build Result Summary

Details

Trigger: Code has been updated by Ashwin Srinivasan.
Revision: b935f9e...
Completed: 25 May 2012, 6:21:29 PM – 2 days ago
Duration: 41 minutes
Successful since: #111 (12 hours before)
Labels: None

Code Changes [See full change details](#)

Ashwin Srinivasan 25 May 2012, 5:37:23 PM
BDEV-327, CR-BAM-4225: Changed wording of EC2 must be enabled warning message

JIRA Issues [See full issue details](#)

Issue	Description
BDEV-327	Home page pointing to EC2 config

Test Summary 4,020 tests in total

0 New Failures 0 Existing Failures 0 Fixed 3 Quarantined

Comments

No comments on this Build.

[Comment](#)

Screenshot: JIRA issues for a build result — Issues tab

JIRA Issues [Add linked issue](#)

This page shows the JIRA issues that have been linked to this build. This includes issues specified in commit messages, labels or comments, as well as issues manually linked to this build.

Related Issues (2 issues)

Type	Key	Summary	Status	Assignee	Fix versions	Related Builds	Operations
Issue	BDEV-484	mvn release new image files and redeploy all images to the prod account	Resolved	Przemek Bruski	Milestone Discoveries	36 related builds	Delete
Issue	BDEV-498	Error with the Streams completion service	Resolved	Przemek Bruski		81 related builds	Delete

Linking JIRA issues to a build

If your organisation uses Atlassian's [JIRA](#) and your administrator has [integrated Bamboo with JIRA](#):

- Bamboo will automatically link JIRA issues to builds.
- You can manually link an issue to a build.

Automatically linking issues to a build

Bamboo will automatically link an issue to a build if you specify a JIRA issue key in a Bamboo build [comment](#) or [label](#), or in a code [commit message](#).

The issue key must be of the default JIRA issue key format (that is, two or more uppercase letters ([A-Z] [A-Z]+), followed by a hyphen and the issue number, for example BAM-123).

On this page:

- [Automatically linking issues to a build](#)
- [Manually linking issues to a build](#)

Related pages:

- [Viewing linked JIRA issues](#)
- [Creating JIRA issues from a build](#)
- [Integrating Bamboo with JIRA](#)

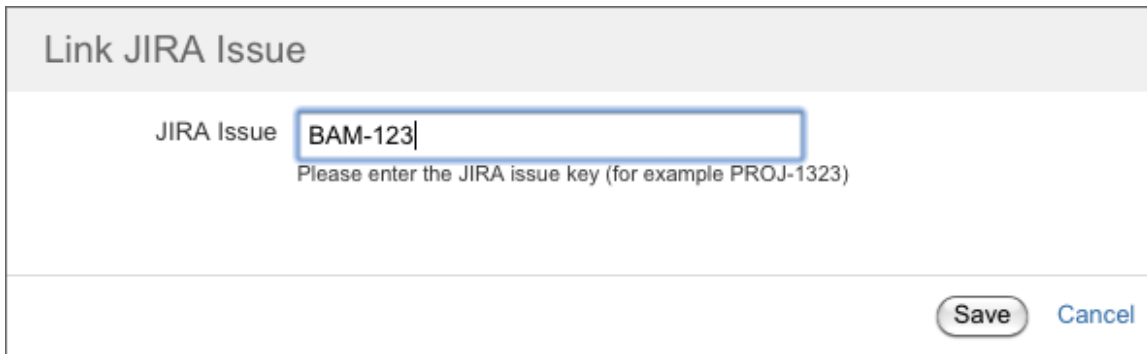
Manually linking issues to a build

If an issue has not been linked automatically to your build, you can manually create a link from that issue to your build.

To manually link a JIRA Issue to a build result:

1. Go to the [plan](#) in Bamboo.
2. Click on the build number for a build result.
3. Click the **Issues** tab in the 'Build Result Summary'. All of the JIRA issues linked to your build will be listed.
4. Click **Add linked issue**.
5. Enter the JIRA issue key of the issue you want to link to this build. Please note, the issue key must be of the default JIRA issue key format (that is, two or more uppercase letters ([A-Z] [A-Z]+), followed by a hyphen and the issue number, for example BAM-123).
6. Click **Save**.

Screenshot: Adding a new JIRA issue link to a build



Link JIRA Issue

JIRA Issue

Please enter the JIRA issue key (for example PROJ-1323)

Save Cancel

Creating JIRA issues from a build

When Bamboo is [integrated with JIRA](#), you can create new JIRA issues right from your Bamboo build result. This lets you easily:

- *Capture* critical infrastructure failures that are keeping your build from passing.
- *Request* that a successful build be deployed to the next environment.
- *Create* a searchable knowledge base of failure causes and solutions.
- *Log* time spent on build failures and use JIRA dashboard gadgets to discover trends over time.

When you create an issue from Bamboo, the issue in JIRA links back to the build result it was created from.

A link to the new issue is displayed in the 'JIRA Issues' section of the Build Result Summary, and on the Issues tab, in Bamboo.

To take advantage of JIRA issue creation in Bamboo:

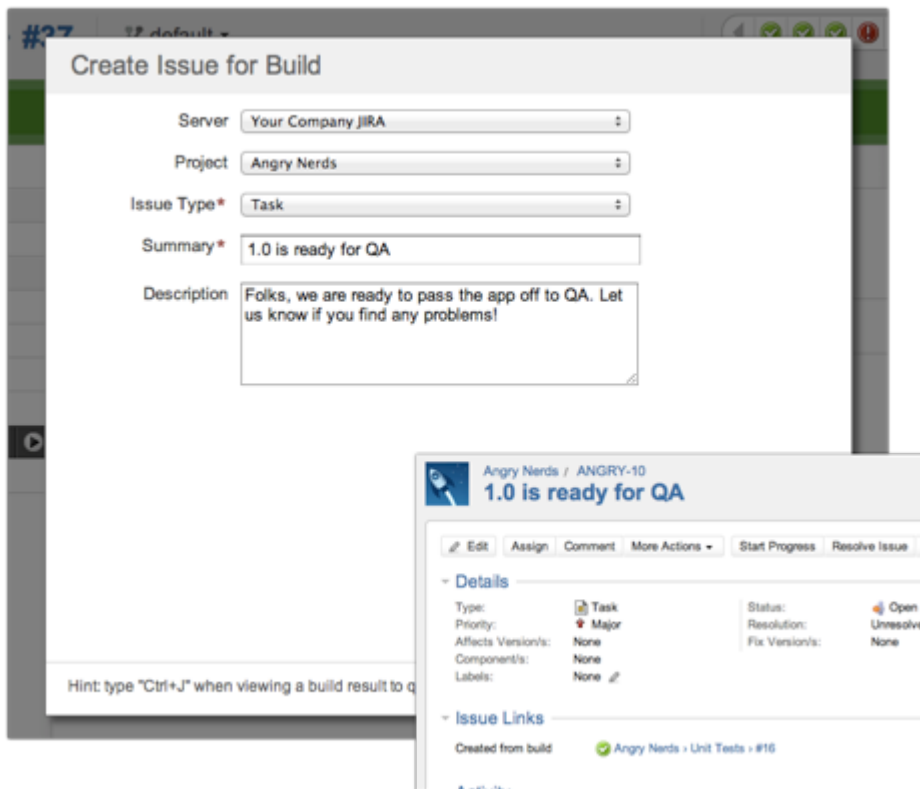
- You require JIRA 5.0, or higher.
- There must be an [application link](#) already set up between JIRA and Bamboo.
- Your JIRA administrator needs to have enabled [fully reciprocal issue linking](#) in JIRA.

Related pages:

- [Linking JIRA issues to a build](#)
- [Viewing linked JIRA issues](#)
- [Integrating Bamboo with JIRA](#)

To create a new JIRA issue from a Bamboo build:

1. On the [Build Result Summary](#), choose **Actions > Create Issue**.
2. Complete the form.
3. Click **Create**.



Integrating Bamboo with Confluence

Integrating Bamboo with Atlassian's [Confluence](#) combines Bamboo's continuous integration capabilities with your wiki to give you a unified view of your software development project.

When Bamboo is integrated with Confluence, you can add the following Bamboo gadgets to a Confluence wiki page:

- [Bamboo Charts](#)
- [Bamboo Plan Summary Chart](#)
- [Bamboo Plan Status](#)

Configuring Bamboo and Confluence to work together simply requires you to set up an application link (two-way) between Confluence and Bamboo.

On this page:

- [Before you begin](#)
- [Set up an application link](#)
- [Try your new configuration](#)
- [Notes](#)

Related pages:

- [Registering External Gadgets](#) (Confluence documentation)

Before you begin

Version Requirements

Application	Version Requirement
Bamboo	Version 3.2 or later
Confluence	Version 3.5.9 or later

Set up an application link

Before you begin:

- **Security Considerations** — The instructions below recommend setting up authentication for the application link between Confluence and Bamboo. Please ensure that you read the [Security Implications for each Authentication Type](#) (Confluence documentation). For example, if you use basic HTTP authentication for the Confluence to Bamboo link, you must specify a user that Confluence uses to log in to Bamboo. Hence, this user's Bamboo permissions will be used (not the Bamboo permissions of the user who is currently logged into Confluence).

Follow the [Confluence instructions](#) to configure the application link in Confluence.

- You will need to set up a two-way link, i.e. select the 'Create a link back to this server' option when adding the application link.
- You will need to configure either OAuth or Trusted Apps authentication for your application link. See [Configuring Authentication for an Application Link](#) for instructions.

Congratulations! You have successfully integrated Bamboo and Confluence.

Try your new configuration

You may wish to read about how to use these two applications together in the following pages:

- Add Bamboo gadgets to Confluence, see [Registering External Gadgets](#) (Confluence documentation).

Notes

If you need further help, please raise a support request in our [support system](#), in the Bamboo project. You may also want to view articles in the [Bamboo Knowledge Base](#) and browse [our forums](#).

Integrating Bamboo with Stash

When Bamboo is integrated with Atlassian's [Stash](#), you can:

- view, in Bamboo, the Stash [changesets](#) that are part of a build.
- click through to Stash to see the [changeset](#) diff for all files that are part of the changeset.
- see, in Stash, the latest [build status](#) associated with a commit when viewing any commit or pull request.

A user who has edit privileges for a plan can make links to source-code files available by connecting the plan to the source repository, as described below.

Related pages:

- [Integrating Bamboo with Atlassian applications](#)
- [Specifying the source repository](#)

To see Stash changesets for a build in Bamboo

1. Navigate to the 'Source Repositories' tab for the plan, as described in [Specifying the source repository](#).
2. Click on a repository name, and then click **Advanced Options**.
3. Choose **Web Repository** > **Stash**.
4. Specify the **Stash URL**, **Stash Project Key** and **Repository Name**.

Screenshot: Specifying a Stash project in Bamboo

The screenshot shows a configuration form for a 'Web Repository' in Bamboo. The 'Web Repository' dropdown is set to 'Stash'. Below it is a text box for 'Stash URL' containing 'https://stash.mycompany.com'. Below that is a text box for 'Stash Project Key' containing 'KEY'. At the bottom is a text box for 'Repository Name' containing 'reponame'. Each field has a descriptive label below it.

To see Bamboo build results in Stash

1. Set up an application link from Bamboo to Stash that uses **Basic Access** for outgoing authentication. See:
 - a. [Adding an Application Link](#)
 - b. [Configuring Basic HTTP Authentication for an Application Link](#)
2. Install the Stash Bamboo plugin (available from the [Atlassian Marketplace](#)) in Bamboo. The plugin allows Bamboo to publish build results, which Stash is able to display. See the [Universal Plugin Manager documentation](#).
3. Set up a build notification on the Bamboo plan. This can only be of the **All Builds Completed** type. See [Configuring notifications for a plan and its jobs](#).

See also [Bamboo integration](#) in the Stash documentation.

Integrating Bamboo with FishEye

When Bamboo is integrated with Atlassian's [FishEye](#), you can:

- view the code changes that triggered a build
- explore a failed build in FishEye and jump directly into the changeset that broke the build
- view the history of the changeset to see what the author was trying to fix
- analyze the change using the side-by-side diff view
- open the relevant files in your IDE.

A Bamboo administrator can make links to individual source-code files available by connecting the plan to the source repository, as described below.

Related pages:

- [Integrating Bamboo with Atlassian applications](#)
- [Specifying the source repository](#)

To integrate Bamboo with FishEye:

1. Navigate to the 'Source Repositories' tab for the plan, as described in [Specifying the source repository](#).
2. Click on a repository name, and then click **Advanced Options**.
3. Choose **Web Repository > FishEye**.
4. Specify the **FishEye URL**, **Repository Name** and **Repository Path**.

Screenshot: Specifying a FishEye project in Bamboo

Web Repository

FishEye

Select, if any, the browsable web repository associated with this build.

FishEye URL *

This is the URL of your FishEye instance e.g http://svn.atlassian.com/fisheye/

Repository Name *

The name of the repository in FishEye e.g myProject

Repository Path

The path from the root of the SCM to the FishEye Repository. This makes the difference between a file as defined by the SCM and as seen in FishEye e.g. /atlassian/myProject/trunk/

 In case links to Fisheye are broken in Bamboo builds, make sure that the Repository Path in Bamboo matches the Repository Path configured in Fisheye for the specified repository.

Integrating Bamboo with HipChat

When Bamboo is integrated with [HipChat](#), you can get notifications from Bamboo in your chat room for events like:

- when a build passes or fails
- when you are assigned responsibility for a breaking build
- when a build you are responsible for has been fixed
- when a manual stage of a build is ready to be run

...and many other [notification](#) events.

Related pages:

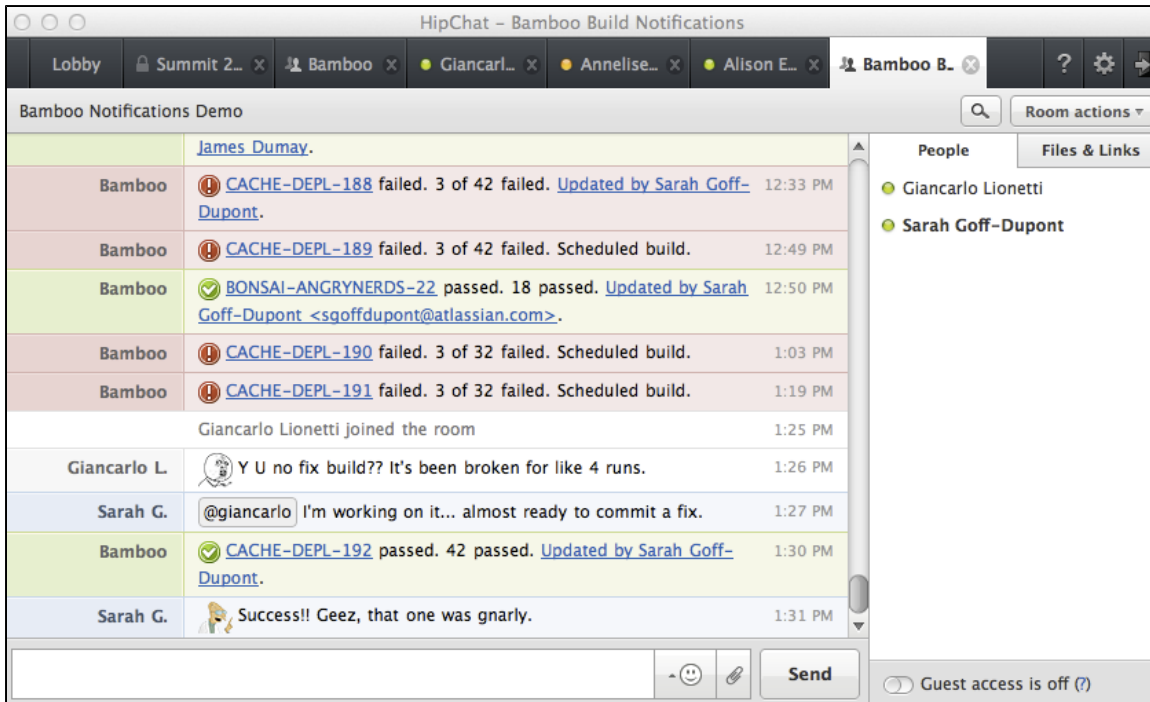
- [Notifications](#)
- [Working with Instant Messenger \(IM\) notifications](#)
- [Integrating Bamboo with Atlassian applications](#)

To integrate Bamboo with HipChat:



1. [Sign up](#) for a HipChat account.
2. [Set up plan notifications](#) in Bamboo that use the 'HipChat' **Recipient Type**.

Screenshot: Bamboo notifications in HipChat: failing builds are shown in red, successful builds in green.



Note

Bamboo does not support using HipChat as a [global IM server](#). HipChat should only be used in the plan notifications area as shown in this screenshot:

Edit Build Notification

Select notification event and recipient

Event **All Jobs Completed**

Notification sent for every job that finishes building

Recipient Type **Hipchat**

API Token*

Your Hipchat API token. You need a 'user' level token. Generate one at https://yourdomain.hipchat.com/group_admin/api.

Room Name*

Name of the Hipchat room (or ID)

☐ Notify

Check this box to notify room participants (ie via Growl) of the new message.

Save

Cancel

Managing your user profile

You can manage your user details, password, notifications preferences and other preferences using your user profile.

To change your personal details:

1. Go to **your name** (the 'Profile' menu) at the top of the page and choose **Profile**.
2. Click **Edit Profile**.
3. Update your personal details as required.

Note that if your user profile is managed using a single sign-on application, like [Atlassian's Crowd](#), you will only be able to edit your **Instant Messaging Address** and **Source Repository Alias**.

Related pages:

- [Changing your password](#)
- [Changing your notification preferences](#)
- [Associating your author name with your user profile](#)

Changing your password

To change your Bamboo password:

1. Go to **your name** (the 'Profile' menu) at the top of the page and choose **Profile**.

2. Click **Change Password**.
3. Complete the form.

i If your password is managed via a single sign-on application, like [Atlassian's Crowd](#), this function will not be available.

Changing your notification preferences

Notifications in Bamboo are triggered by a range of events for a [plan](#), including build completion, build outcomes and comments being posted against build results. You can configure whether notifications are sent for a particular event and who they are sent to. Users can choose whether to receive their notifications via [email](#), [IM](#), [both](#) or [neither](#).

You can see which notifications are currently applicable to you, in your user profile: go to **your avatar** (the 'Profile' menu) at the top of the page, choose **Profile**, and then click the **Notifications** tab.

Before you begin:

- You must have the 'Edit' permission for a plan to [add or remove notifications](#) for it.

Related pages:

- [Managing your user profile](#)
- [Configuring notifications for a plan and its jobs](#)

To change your notification preferences:

1. Go to **your name** (the 'Profile' menu) at the top of the page and choose **Profile**.
2. Click the **Notifications** tab.
3. Click **Edit Notification Preferences**, on the right.
4. Choose an option from **How would you like Bamboo to send you notifications**. If you choose one of the IM options, you also need to specify an **Instant Messaging Address** on the **Personal Details** tab.
5. Choose an **Email Format** option, if required.
6. Click **Save**.

Screenshot: User Profile

User Profile: Andrew Lui

Personal Details Notifications

Edit User Profile

How would you like Bamboo to send you notifications: Send email and instant message

Email Format:

☐ Text Email

☒ Multipart Email (Text and HTML)

Please specify the format you want your emails sent in.

Save Cancel

Associating your author name with your user profile

An *author* is any person who checks in code to a repository that is associated with a Bamboo [plan](#). An author need not be a Bamboo user. Your *Author Name* is your login name for the source-code repository. This is the identity that the SCM associates with tasks you perform on the repository. However, if this is not the login you use for Bamboo, then Bamboo may not be able to make the connection between your SCM login and your Bamboo login. See also [Managing authors](#).

When your Bamboo user profile is associated with your author name, then Bamboo is able to:

- match your SCM activity with your Bamboo activity.
- show information about your recent builds on your 'My Bamboo' tab.
- show a **User Details** tab in your 'Author' information.

To associate your author name with your user profile:

1. Go to **your name** (the 'Profile' menu) at the top of the page and choose **Profile**.
2. Click **Edit Profile**.
3. Select your author name from the **Source Repository Aliases** list. If your name does not appear in the list, click **Add Alias**. Note that your author name (alias) need not be identical to your user name.
4. Click **Save**.

You can link more than one author name to a Bamboo user name.

Related pages:

- [Managing your user profile](#)
- [Managing authors](#)

Bamboo variables

Variables can be used to set static values that are used when building plans in Bamboo.

- **Global variables** are defined across your entire Bamboo instance, and have the same (static) value for every plan that is built by Bamboo.
- **Plan variables** are similar to global variables, but are defined for specific plans. Plan variables override global variables with the same name. You can also override a plan variable for a build, if you have triggered the build manually.
- **Build-specific variables** are evaluated by Bamboo dynamically at build time. The source of a build-specific variable can either be a Bamboo property or one of the [default plugins](#) (assuming they have been enabled).
- **Deployment variables** are a number of standard reserved variables that are available during deployment executions.
- **System variables** also apply across your entire Bamboo instance and inherit their values from system or environment variables of the same name.

You can generate variables of your own, using a similar format, however you cannot create a variable that is already in use by Bamboo. See [Defining global variables](#) and [Defining plan variables](#).

How to use variables

Using variables

Variables can be used in all fields of a Task, with the exception of password variables. Variables can also be used with deployments.

`${bamboo.variableName}`

You can override a plan variable for a build, if you have triggered the build manually. For details, see [Triggering a plan build manually](#).

Defining variables

See [Defining global variables](#) for information on how to define your own global variables.

See [Defining plan variables](#) for information on how to define your own plan variables.

See [Defining deployment environment variables](#) for information on how to define your own plan variables.

On this page:

- [How to use variables](#)
 - [Using variables](#)
 - [Defining variables](#)
 - [Build-specific variables](#)
- [Deployment variables](#)
 - [Variables associated with releases](#)
- [System variables](#)
 - [Using variables in bash](#)
- [JIRA variables](#)
- [Examples](#)
 - [Using variables in bash](#)
 - [Maven examples](#)
 - [Ant examples](#)
- [Specifying capabilities as variables](#)
 - [Using capabilities](#)

Related pages:

- [Defining global variables](#)
- [Defining plan variables](#)
- [Running a plan build manually](#)
- [Configuring plugins](#)
- [Variables for deployment environments](#)

Build-specific variables

The following build-specific variables are available by default:

Build-specific variable	Description
bamboo.buildKey	The job key for the current job, in the form PROJECT-PLAN-JOB, e.g. BAM-MAIN-JOBX
bamboo.buildResultKey	The result key when this job executes, in the form PROJECT-PLAN-JOB e.g. BAM-B00-J0B1

bamboo.buildResultsUrl	The URL of the result in Bamboo once the job has finished executing.
bamboo.buildNumber	The Bamboo build number, e.g. 123
bamboo.buildPlanName	The Bamboo plan name e.g. Some Project name - Some plan name
bamboo.buildTimeStamp	The time when build was started in ISO 8601 format e.g. 2010-01-01T01:00:00.000+01:00
bamboo.buildForceCleanCheckout	Whether the "Force Clean Build" option was used, values:true/false
bamboo.build.working.directory	The working directory that the build is being executed on
bamboo.ManualBuildTriggerReason.userName	The user who triggered the manual build
Generic repository variables	
bamboo.repository.revision.number	The revision number
bamboo.repository.branch.name	The repository branch name (for Bamboo version 4.2 or later)
bamboo.repository.previous.revision.number	The previous revision number (might not exist if for example is initial build)
Subversion	
bamboo.custom.svn.revision.number	The revision number
bamboo.custom.svn.lastchange.revision.number	The last changed revision number
bamboo.custom.svn.username	User name used for repository authentication
bamboo.repository.svn.repositoryUrl	The repository url
CVS	
bamboo.custom.cvs.last.update.time	The last updated timestamp
bamboo.custom.cvs.last.update.time.label	The last updated timestamp to be used as a label for post build result labelling. The spaces in the cvs version string are replaced with '_'
Perforce	
bamboo.custom.p4.revision.number	The change set number
bamboo.custom.p4.username	User name used for repository authentication
bamboo.custom.p4.port	Port used for repository communication
bamboo.custom.p4.client	Client used for repository communication
Git	
bamboo.repository.git.branch	The branch
bamboo.repository.git.repositoryUrl	The repository url
Mercurial	
bamboo.repository.hg.repositoryUrl	The repository url
bamboo.repository.hg.branch	The branch
bamboo.repository.hg.username	User name used for repository authentication

- *System variables* also apply across your entire Bamboo instance and inherit their values from system or environment variables of the same name.

Deployment variables

Bamboo manages a number of standard reserved variables that are available during deployment executions.

Variables later in the following list override the previous ones in case of repeating names:

- global variables
- release variables as defined below
- user variables defined at environment level
- autogenerated variables:

Variable	Description
bamboo.agentID	The id of the agent that the deployment is executed on.
bamboo.agentWorkingDirectory	The path to the working directory on the agent. This is not the same as the Bamboo working directory.
bamboo.build.working.directory	The path to the working directory for Bamboo. This is used by both the build plan and the deployment project.
bamboo.deploy.environment	The name of the environment that the release is to be deployed to.
bamboo.deploy.project	The name of the deployment project.
bamboo.deploy.rollback	True if the release being deployed is older than the release being replaced.
bamboo.deploy.version	The name of the release that is being deployed.
bamboo.deploy.version.previous	The name of the release that is being replaced (if available).
bamboo.resultsUrl	The URL to the screen in Bamboo that displays build results.

You can generate variables of your own, using a similar format, however you cannot create a variable that is already in use by Bamboo. See [Variables for deployment environments](#) for more information on configuring your deployment environment variables.

Variables associated with releases

Bamboo makes the following types of variables available for deployment releases:

- Snapshots of values for [global variables](#).
- Snapshots of values for [plan variables](#).
- The autogenerated release variables in the following table:

Variable	Description
bamboo.buildNumber	The build result from which the release is created.
bamboo.buildResultKey	The key of the build result from which the release is created e.g. KUNG-FOO-JOB1-35
bamboo.planKey	The key of the plan related to the release e.g. KUNG-FOO
bamboo.planName	The name of the plan related to the release e.g. Kung - Foo
Repository	
bamboo.planRepository.<position>.branchName	The name of the branch in the repository (depends on availability from the VCS used) e.g. default
bamboo.planRepository.<position>.name	The name of of the repository, as shown in the repository for the plan e.g. Mercurial
bamboo.planRepository.<position>.revision	The revision use to build this release. Format depends on the VCS used.
bamboo.planRepository.<position>.type	The type of the repository, as defined by a repository plugin e.g. hg, svn, git

In the variable names in the table above, <position> is the repository in the plan's repository list. It can be ignored for the first repository on the list; that is planRepository.name is equivalent to planRepository.1.name.

Note that several of the variables in the above table are actually those associated with the build plan.

System variables

The usage format for all system variables is:

```
${system.<variable>}
```

For example, if you have a *system variable* MYPATH=C:\MyPath; you can use a *Bamboo system variable* system.MYPATH which will inherit the same value as the system variable.

 In older Bamboo versions using 'PATH' in the Environment Variables field (of a Script task) doesn't set the windows PATH variable, whereas using 'Path' sets Path and PATH in cmd shell.

Using variables in bash

Bamboo variables are exported as bash shell variables. All full stops (periods) are converted to underscores. For example, the variable `bamboo.my.variable` is `$bamboo_my_variable` in bash.

JIRA variables

Note that these JIRA variables can be accessed from a Bamboo build only when that build was triggered by [releasing a version in JIRA](#).

JIRA variable	Description
<code>\${bamboo.jira.baseUrl}</code>	The URL of your JIRA server.
<code>\${bamboo.jira.projectKey}</code>	The key of the triggering JIRA project.
<code>\${bamboo.jira.projectName}</code>	The name of the triggering JIRA project.
<code>\${bamboo.jira.version}</code>	The release version of the triggering JIRA project.
<code>\${bamboo.jira.username}</code>	The username of the user who triggered the release build.

Examples

Using variables in bash

Bamboo variables are exported as bash shell variables. All full stops (periods) are converted to underscores. For example, the variable `bamboo.my.variable` is `$bamboo_my_variable` in bash.

Maven examples

For example, you may want your Maven 2 version to be determined by Bamboo. In Maven 2 `pom.xml` you may have:

```
...
<groupId>com.atlassian.boo</groupId>
<artifactId>boo-test</artifactId>
<packaging>jar</packaging>
<version>1.1.${bambooBuildNumber}-SNAPSHOT</version>
...
```

You can then specify the following in the **Goal** field of your build plan:

```
clean package -DbambooBuildNumber=${bamboo.buildNumber}
```

When the command runs, Bamboo will replace the `buildNumber` with the actual number (e.g. 1102), which will be passed to the underlying Maven build to use. The command will then produce a jar that looks like this: `boo-test-1.1.1102-SNAPSHOT.jar`.

Ant examples

You can then specify the following in the **Target** field of your build plan:

```
-f build.xml -DbambooBuildNumber=${bamboo.buildNumber}
```

When the command runs, Bamboo will replace the `buildNumber` with the actual number (e.g. 1102), which will be passed to the underlying Ant build to use.

Specifying capabilities as variables

You can also specify a capability to be used in a similar way to a global variable.

The format of the capability should be as follows:

```
${bamboo.capability.<capability_key>}
```

For example,

- Custom

`${bamboo.capability.<capability_key>}`
- JDK

`${bamboo.capability.system.jdk.<jdk_label>}`
- Builder

`${bamboo.capability.system.builder.<builder_type>.<builder_label>}`
e.g. `${bamboo.capability.system.builder.maven.Maven1}`
- Perforce

`${bamboo.capability.system.p4Executable}`

If you click on a capability, the specific capability key will be contained in the URL.

Please note, the space characters in the URL will be replaced with '+' characters. We recommend that you do not use capability labels with space characters, if you wish to use them as variables. A possible solution for space characters is to format them with '{}' symbols, however, this does not work in all cases.

Using capabilities

Global and Build-Specific Variables can be used in a specific fields of your build plan, as [specified above](#). For capabilities,

- **System Capabilities** are available to all of these fields, (i.e. global and build-specific).
- **Agent Capabilities** (i.e. agent-specific and shared/server capabilities) are available only to the build-specific fields. (i.e. not available to **Repository URL**, **CVS Root** or **Branch name**.)

For example,

If you wanted to specify a system variable, but have it set to different values on each agent, do the following:

1. Set the following as a system environment variable field on the **Builder** tab:

`${bamboo.capability.thatsystemvariable}`
2. Specify the system environment variable as a custom capability on each of your agents, and set to the capability to the different values, as desired.

Defining global variables

When configuring a plan, you may want to specify variables to be used in the build process. For details on how variables are used, see [Bamboo variables](#).

Global variables are one type of variable that is available to you. Global variables are defined across your entire Bamboo instance, and have the same value for every plan that is built by Bamboo. If you want to define a variable for a specific plan rather than across all plans, define a plan variable as described in [Defining plan variables](#).

Global variables can be accessed by using `${bamboo.globalVarName}`. Global variables can also be overridden at runtime when running a manual build. For more information, see [Running a plan build manually](#).

Related pages:

- [Bamboo variables](#)
- [Defining plan variables](#)
- [Running a plan build manually](#)

To access the global variables page:

1. Click the **Administration** link in the top navigation bar.

2. Click **Global Variables**, in the left navigation column under the 'Build Resources' section.
3. Add, update or delete the global variables, as desired:
 - Click **Add** to add a new variable once you have entered the key and value for it.
 - Updates to existing rows will be saved as you move between cells in the table.
 - Click the trash can to delete a variable.

Screenshot: Global variables

Global Variables

You can use this page to view, add, edit and delete global variables. Global variables are available on every build run in Bamboo and can be accessed using `${bamboo.globalVarName}`

Key	Value	
		Add
confluence.current.stable.branch	confluence-project-3.5-stable	
currentStableVersion	3.0	
finalBranchName	atlassian_bamboo_3_0_branch	
mavenOptions	MAVEN_OPTS="-Xmx1024m -Xms128m -XX:MaxPermSize=128M"	
nextMilestone	rc1	
nextReleaseVersion	3.1	
nextStableVersion	3.0.4	
stableBranch	atlassian_bamboo_3_0_branch	
trunkVersion	3.1-SNAPSHOT	

Defining plan variables

When configuring a plan, you may want to specify variables to be used in the build process. For details on how variables are used, see [Bamboo variables](#).

Plan variables are one type of variable that is available to you. A plan variable is defined for one specific plan, and has the same value every time that plan is built. If you want to define a variable across all plans rather than a single plan, define a global variable as described in [Defining global variables](#).

Plan variables can be accessed by using `${bamboo.varName}`. Plan variables can also be overridden at runtime when running a manual build. For more information, see [Running a plan build manually](#).

Related pages:

- [Defining global variables](#)
- [Bamboo variables](#)
- [Running a plan build manually](#)

Before you begin:

- Note that plan variables override global variables with the same name.

To define a plan variable:

1. Click **Dashboard**, then the **All Plans** tab, then the name of the plan in the list, to get to the plan you want to edit.
2. Choose **Actions > Configure Plan**.
3. Click the **Variables** tab.
4. Add, update or delete plan variables, as desired:
 - Click **Add** to add a new variable once you have entered the key and value for it.
 - Updates to existing rows will be saved as you move between cells in the table.
 - Click the trash can to delete a variable.

Screenshot: Adding a plan variable

Plan Details
Source Repositories
Dependencies
Permissions
Notifications
Variables
Miscellaneous
Audit Log
Run
Actions

Variables

You can use this page to view, add, edit and delete Plan variables. Plan variables are available on every build run in Bamboo and can be accessed using `${bamboo.varName}`

Key	Value	
		Add
foo	bar	
test	test	

💡 If the key contains the phrase "password", the value will be masked with "*****" in the Build Logs. E.g. if the key is password, repoPassword or userPassword the value will be masked.

Bamboo permissions

Bamboo provides the following types of permissions to allow fully customisable control of access to the continuous delivery workflow:

- Global permissions
- Build plan permissions
- Deployment permissions
 - Deployment project permissions
 - Deployment environment permissions

This page uses the following key for Bamboo permissions:

Permissions key:

- ★ - Permission is set by default
- ☆ - Permission is available as an option
- ✶ - Permission not available, even as an option

On this page:

- [Global permissions](#)
- [Build plan permissions](#)
- [Deployment permissions](#)
 - [Deployment projects](#)
 - [Deployment environments](#)

Global permissions

Global permissions level control the ability to view the system, create a new build plan and use administration tools. Global application permissions are accessed from the Global permissions page within the Bamboo administration pages.

	User	Access	Create Plan	Restricted Admin	Admin
General users	Anonymous	★	✶	✶	✶
	Logged-in	★	★	☆	☆
Groups	Bamboo-admin	★	★	☆	★
Users	Administrator	★	★	☆	★
	Bamboo Evaluator*	★	★	☆	★

*Only enabled on the evaluation licence

Key:

- **Access** - User can access Bamboo
- **Create Plan** - User can create a new plan in Bamboo
- **Restricted Admin** - User can perform some administration operations and view all plans in Bamboo
- **Admin** - User can perform all operations and view all plans in Bamboo

Build plan permissions

Build plan permissions allow a user to control access to the functions of the build plan. These include viewing, editing, building, cloning and administering a build plan. Build plan level permissions are accessed from the build plan configuration page.

User	View	Edit	Build	Clone	Admin
Anonymous	★	★	★	★	★
Logged-in	★	★	★	★	★
Administrator	★	★	★	★	★

Key:

- **View** - User can view the plan and builds
- **Edit** - User can view and edit the configuration of the plan and it's jobs, not including permissions or stages
- **Build** - User can trigger a manual build, or suspend and resume the plan
- **Clone** - User can clone the plan
- **Admin** - User can administrate all components of this plan including the stages and the plan's permissions

Deployment permissions

Bamboo's deployments features allow you to control permissions for both deployment projects and deployment environments.

Deployment projects

User	View	Edit
Anonymous	★	★
Logged-in	★	★
Administrator	★	★

Key:

- **View** - Who can see this project and its associated environments
- **Edit** - Who can change the details of the project, related plan, any of the environment configuration and create releases

Deployment environments

User	View	Edit	Deploy
Anonymous	★	★	★
Logged-in	★	★	★
Administrator	★	★	★

Key:

- **View** - Who can see this project and its associated environments. Users must also have view permission on the deployment project
- **Edit** - Who can edit this environment
- **Deploy** - Controls who can deploy releases to this environment and create releases for this project.

Administering Bamboo

Bamboo is a [continuous integration](#) (CI) server. Bamboo assists software development teams by providing:

- automated building and testing of software source-code status.
- updates on successful/failed builds.
- reporting tools for statistical analysis.

This administration guide has information about managing the Bamboo server itself. Please see [Using Bamboo](#) for help with setting up CI builds.

Administering

[System settings](#)
Configuring the Bamboo server.

[Agents and capabilities](#)
Setting up services, including Elastic Bamboo, to perform builds.

[Users and permissions](#)
Managing users, groups and their permissions.

[Add-ons](#)
Extending Bamboo.

[Data and backups](#)
Managing databases, data and backups.

[Security](#)
Managing security for agents and Elastic Bamboo.

Installing

[Bamboo installation guide](#)

[Bamboo EAR-WAR installation guide](#)

[Bamboo installation guide for Linux](#)

[Bamboo installation guide for Mac](#)

[Bamboo installation guide for Windows](#)

[Connecting Bamboo to an external database](#)

[Bamboo remote agent installation guide](#)

[Supported platforms](#)

See also

[Getting started](#)

[Using Bamboo](#)

[Release notes](#)

[Bamboo security advisories](#)

[Bamboo documentation downloads](#)

System settings

For information on configuring system settings, see the following topics:

- [Viewing Bamboo's system information](#)
- [Updating your Bamboo license details](#)
- [Specifying Bamboo's title](#)
- [Specifying Bamboo's URL](#)
- [Logging in Bamboo](#)
- [Enabling GZIP compression](#)
- [Enabling Bamboo's Remote API](#)
- [Configuring system properties](#)
- [Finding Your Bamboo Support Entitlement Number \(SEN\)](#)
- [Configuring Gravatar support](#)

Viewing Bamboo's system information

When you [installed](#) Bamboo, you provided information about how the system should be configured. You can view the system information from your administration console in Bamboo.

The system information contains useful data for you to send to Atlassian when requesting [support](#).

Related pages:

- [Locating important directories and files](#)

To view your **Bamboo system information**:

1. Click **Administration** in the top navigation bar.
2. Click **System Information** (under 'System') in the left navigation panel.

Screenshot: Bamboo system information (cropped)

System Information

System Properties

System Date	Tuesday, 15 Feb 2011
System Time	15:49:22
Up Time	1 day, 31 minutes, 14 seconds (since Mon Feb 14 15:18:08 EST 2011)
Username	panda
User Timezone	Australia/Sydney
User Locale	English (United States)
System Encoding	MacRoman
Operating System	Mac OS X 10.6.5
Operating System Architecture	x86_64
Available Processors	8
Application Server Container	Apache Tomcat/6.0.18

Java / JVM Information

Java Version	1.6.0_17
Java Vendor	Apple Inc.
JVM Spec. Version	1.0
JVM Spec. Vendor	Sun Microsystems Inc.
JVM Version	14.3-b01-101
JVM Vendor	Apple Inc.
JVM Name	Java HotSpot(TM) 64-Bit Server VM
JRE Version	1.6.0_17-b04-248
JRE Name	Java(TM) SE Runtime Environment

Network

Host Name	panda.sydney.atlassian.com
IP Address	172.20.6.108

Memory Statistics

Total Memory	292 MB
Free Memory	63 MB
Used Memory	229 MB

Bamboo Version Information

Version	3.0
Build Number	2301
Build Date	14/02/11

Updating your Bamboo license details

When you upgrade or renew your Bamboo license, you will receive a new license key. You will need to update your Bamboo server with the new license key.

Please see the [Licensing FAQ](#) if you have questions to do with licensing.

Related pages:

- [System settings](#)

To update your Bamboo license key:

1. Click **Administration** in the top menu bar.
2. Click **License Details** (under 'System') in the left navigation panel. This will display your existing Bamboo license details.
3. Paste your new license into **License Key**.
4. Click **Save New License**.

Specifying Bamboo's title

Bamboo's *name* is the displayed title of this installation of Bamboo. It will appear throughout Bamboo (e.g. on the [Dashboard](#)), and in the window title of your users' browsers.

Related pages:

- [System settings](#)

To specify Bamboo's title:

1. Click **Administration** in the top menu bar.
2. Click **General Configuration** (under 'System') in the left navigation column.
3. Type the display title for your Bamboo server (e.g. "MyCompany's Bamboo") into the **Name** field.
4. Click **Save**.

Specifying Bamboo's URL

This is the base URL of this installation of Bamboo. All links created (for links in Bamboo email notifications etc.) will be prefixed by this URL.

To specify Bamboo's URL:

1. Click **Administration** in the top menu bar.
2. Click **General Configuration** (under 'System'), in the left navigation panel.
3. In the **Base URL** field, type the URL address of your Bamboo server (for example, "<http://keg:8080/bamboo>").
4. Click **Save**.

Related pages:

- [System settings](#)

Notes

- **Accessing Bamboo from Outside a Firewall** — When accessing Bamboo through a web browser, most Bamboo URL links (which provide navigation throughout the product) will use the base URL that was originally entered into your browser's URL field. For example, to access Bamboo through a web browser on the same machine running Bamboo itself, you may have entered the base URL:

```
http://localhost:8085/...
```

into your browser's URL field. Consequently, most Bamboo URL links will use the base URL:

```
http://localhost:8085/...
```

However, URL links to a Bamboo instance that are provided in Bamboo email notifications and by some Bamboo plugins, will use the base URL set on this 'General Configuration' page. Hence, if you configure the **Base URL** field above to one that can only be accessed internally, behind a firewall, then you may have problems accessing this Bamboo instance externally.

Logging in Bamboo

The information on this page relates to the *Bamboo server (atlassian-bamboo) logs only*. You cannot change the logging configuration for the build logs.

Bamboo generates two distinct sets of logs:

- **Build logs:** The build logs are generated each time a plan is executed. All information specific to the build is stored in these logs, which can be downloaded as an artifact (see [Viewing a build's artifacts](#)). The build logs are located in the <Bamboo-Home>/xml-data/builds/ sub-directories.

On this page:

- [Configuring the level of logging on the Bamboo server](#)
- [Configuring the level of logging on remote agents](#)
- [Configuring the location of the atlassian-bamboo logs](#)

Related pages:

- [System settings](#)
- [Viewing a build's artifacts](#)
- [Locating important directories and files](#)

- **atlassian-bamboo logs:**

- **atlassian-bamboo logs for the Bamboo server** — Bamboo records all server activity in the `atlassian-bamboo.log`. The location of the `atlassian-bamboo.log` file can be viewed in Bamboo's [System Information](#) under the 'Bamboo Paths' section.

In case of a Tomcat webapp deployment, the logs are piped out to `catalina.out` file.

 Please note, this log is different from the `bamboo.log` found in the `<Bamboo-Install>/logs` directory, which is the log written by the Java Service wrapper.

- **atlassian-bamboo logs for elastic agents** — Elastic agent activity is logged inside the elastic instance where the elastic agent runs. To access the elastic agent logs (`atlassian-bamboo.log` and `bamboo-elastic-agent.out`) use ssh to log in to your elastic instance as described in [Viewing an elastic instance](#) and retrieve the logs.
- **atlassian-bamboo logs for remote agents** — All agent activity is recorded in `atlassian-bamboo-agent.log` file stored on the agent machine. These are generated in the running directory of the agent. The running directory can be viewed in the [remote agent's system properties](#) under the 'Bamboo Paths' section.

See [Locating important directories and files](#) for information on where to find other important files in Bamboo.

Configuring the level of logging on the Bamboo server

Bamboo uses the [log4j library](#) for logging during runtime. The logging levels can be changed by editing the `<Bamboo-Install>/webapp/WEB-INF/classes/log4j.properties` file. There are five logging levels available: 'DEBUG', 'INFO', 'WARN', 'ERROR' and 'FATAL'. Each logging level provides more logging information than the level after it:

DEBUG > INFO > WARN > ERROR > FATAL

i.e. DEBUG provides the most verbose logging and FATAL provides the least verbose logging.

You can adjust the logging levels for the different Bamboo packages on the fly, using the runtime log4j configuration tool in the Bamboo administration console. The default log settings are still stored in the `log4j.properties` file. When you view the log settings page for the first time you will see the default log settings as defined in `log4j.properties`. All changes to the log settings via the runtime log4j configuration tool *will not be persisted* and are valid during Bamboo runtime only.

Before you begin:

- Note, you do not need to restart your Bamboo server for any logging changes to take effect.

To change the level of logging on your Bamboo server:

1. Click **Administration** in the top navigation bar.
2. Click **Log Settings** (under 'System') in the left navigation panel. The 'Bamboo Log Settings' page will display showing the Bamboo packages being logged (see screenshot below).
 - To change the logging level of a package that is already being logged, locate the Bamboo package, select the desired logging level from the list next to it and click **Save**.
 - To start monitoring a package in the Bamboo logs, enter the class name in the text box at the top of the page, select the desired logging level from the list next to it and click **Add**.
 - To stop logging a package, locate the Bamboo package and click **Delete** next to it.

Screenshot: Bamboo log settings

Bamboo Log Settings

On the fly switch between Production and Debug Logging. The settings are predefined in log4j.properties

Add a Class to Log4j

Add a new class and level to be included in the bamboo log

	DEBUG ▼
Add	

Edit current Levels

Change the log level

Package	Current Level	New Level
com.atlassian.bamboo.filter.AccessLogFilter	DEBUG	ALL ▼ Save Delete
com.atlassian.bamboo.user.BambooUserManagerImpl	WARN	ALL ▼ Save Delete
com.atlassian.user.impl.cache.CachingGroupManager	WARN	ALL ▼ Save Delete
com.opensymphony.xwork.util.LocalizedTextUtil	ERROR	ALL ▼ Save Delete
com.opensymphony.xwork.util.OgnlValueStack	ERROR	ALL ▼ Save Delete
net.sf.hibernate	WARN	ALL ▼ Save Delete
net.sf.hibernate.impl.SessionImpl	ERROR	ALL ▼ Save Delete
root	INFO	ALL ▼ Save Delete

Configuring the level of logging on remote agents

The runtime log4j configuration tool in the Bamboo administration console can only be used to modify the logging levels for the Bamboo server. To configure the logging levels for your remote agents, you will need to update the log4j.properties file manually.

You can control the logging for each of remote agents separately from the Bamboo server. To do this, simply repeat the process described below for multiple remote agents, so that each remote agent has a log4j.properties file that overrides the log4j.properties file on the Bamboo server.

To change the level of logging on your remote agent:

1. Configure a log4j.properties file for your remote agent. This can be any log4j.properties file. If you do not already have a log4j.properties file, you can take a copy of the log4j.properties file from the server, copy it to your remote agent and configure it as desired:
 - The rootLogger property in the log4j.properties file controls the verbosity of logs being generated at the top level. By default, the root level logging is set to 'INFO'. To change the root level logging, find the following lines in <Bamboo-Install>/webapp/WEB-INF/classes/log4j.properties file and update the value of log4j.rootLogger to the desired logging level:

```
#  
# Change the following line to configure the bamboo logging levels (one  
# of INFO, DEBUG, ERROR, FATAL)  
#  
log4j.rootLogger=INFO, console, filelog
```

- Modify the logging level for any of the individual packages in the `log4j.properties` as desired, e.g. `log4j.category.webwork=WARN`
2. Save changes to the file.
 3. Update the `log4j.configuration` system property on your remote agent to point to the `log4j.properties` file. To do this, add the following line to the `<bamboo-agent-home>/conf/wrapper.conf` file:

```
wrapper.java.additional.3=-Dlog4j.configuration=/full/path/to/log4j.properties
```

where **/full/path/to/log4j.properties** is the **absolute path** of your `log4j.properties` file.

4. Restart your remote agent.

Configuring the location of the atlassian-bamboo logs

To change the directory that the atlassian-bamboo logs are generated to, you must set the environment variable for the target location of the logs, as seen below:

```
log4j.appender.fileLog.file=/my/path/to/atlassian-bamboo.log
```

Note that the new log file location applies to both the server and remote agents. If using an absolute path this may result in aggregated logs.

Enabling GZIP compression

You can enable GZIP compression in order to reduce the size of Bamboo's web pages. This is useful if Bamboo is being run over slow networks. There is a slight performance penalty, and note that GZIP may not work for languages other than English.

Related pages:

- [System settings](#)

To enable GZIP compression:

1. Click **Administration** in the top navigation bar.
2. Click **General Configuration** (under 'System') in the left navigation panel.
3. Select **Apply gzip compression to reduce the size of Bamboo's web pages?**
4. Click **Save**.

Enabling Bamboo's Remote API



Please note, the Bamboo Remote API *has been deprecated* in favour of the new [Bamboo REST API](#).

You can access Bamboo's data from an external program by using Bamboo's REST-style remote API.

Configuring system properties

The default settings on a number of Bamboo functions can be configured by setting the appropriate system properties. This page provides general instructions on how to set a system property in Bamboo.

Bamboo on UNIX-based operating systems (such as Solaris, Linux or Mac OS X) can be started by either executing the `bamboo.sh` script or using the Java Service Wrapper packaged with Bamboo.

Bamboo on Windows-based operating systems can be started by running the `startup.bat` file from the command line (which is the same as running the 'Start in Console' option from the Windows Start menu) or as a Windows Service. Both approaches start Bamboo using the Java Service Wrapper.

On this page:

- [Configuring a Bamboo system property \(UNIX\)](#)
- [Configuring a Bamboo system property \(Windows\)](#)
- [Configuring Bamboo runtime parameters for bamboo.war](#)

Related pages:

- [System settings](#)
- [Configuring Bamboo on start-up](#)

Configuring a Bamboo system property (UNIX)

Before you begin:

- Bamboo must be shut down before modifying any of its system properties. Once you have modified one or more system properties, they will come into effect when Bamboo is restarted.
- If you have any [elastic agents](#) running, ensure that they are [shut down](#) before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become [orphaned from your Bamboo server](#).

To configure a system property via the `bamboo.sh` file:

1. Open the Bamboo start-up script `bamboo.sh` in a text editor. (This is usually located at the root of your Bamboo installation directory.)
2. Locate the variable `RUN_CMD` in `bamboo.sh` and add the system property as a parameter to the `java` command string value of `RUN_CMD`, by adding the `'-D'` prefix to the system property.
For example, if you wanted to set the `bamboo.agent.heartbeatInterval` system property to 10 (seconds), you would add the parameter `-Dbamboo.agent.heartbeatInterval=10` to the `java` command string value of `RUN_CMD` such that the `RUN_CMD` variable assignment in `bamboo.sh` might look like:

```
RUN_CMD="java -server -Xms256m -Xmx512m -XX:MaxPermSize=256m
-Dbamboo.agent.heartbeatInterval=10 -Djava.awt.headless=true -classpath
$CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085
com.atlassian.bamboo.server.Server 8085 ./webapp /"
```

3. Save your changes to the `bamboo.sh` file and start Bamboo.

Configuring a Bamboo system property (Windows)

Before you begin:

- Bamboo must be shut down before modifying any of its system properties. Once you have modified one or more system properties, they will come into effect when Bamboo is restarted.
- If you have any [elastic agents](#) running, ensure that they are [shut down](#) before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become [orphaned from your Bamboo server](#).

To configure a system property via the Java Service Wrapper `wrapper.conf` configuration file:

1. Open the Bamboo Wrapper configuration file `wrapper.conf` in a text editor. (This is usually located in the `conf` subdirectory of your Bamboo installation directory.)
2. Locate the set of variables beginning `wrapper.java.additional.X`, where `X` is a series of consecutive numbers starting from '1'. After the final `wrapper.java.additional.X` variable in this set, add a new variable `wrapper.java.additional.Y`, where `Y` is the next consecutive number in this set of variables.
3. Add the entire system property with the Java `'-D'` prefix and assign it to the value of `wrapper.java.additional.Y`.
For example, if you wanted to set the `bamboo.agent.heartbeatInterval` system property to 10 (seconds), you would add a new variable `wrapper.java.additional.4` to `wrapper.conf` and assign it the value `-Dbamboo.agent.heartbeatInterval=10`, such that this section of the `wrapper.conf` file might look like:

```
wrapper.java.additional.1=-Dorg.mortbay.xml.XmlParser.NotValidating
=true
wrapper.java.additional.2=-XX:MaxPermSize=256m
wrapper.java.additional.3=-Djava.awt.headless=true
# And now for the new variable:
wrapper.java.additional.4=-Dbamboo.agent.heartbeatInterval=10
```

4. Save your changes to the `wrapper.conf` file and start Bamboo.

Configuring Bamboo runtime parameters for `bamboo.war`

The application container that deploys bamboo has to be configured with the additional java parameter.

Example Tomcat:

```
./bin.setenv.sh
```

```
...  
JAVA_OPTS="-server -XX:MaxPermSize=256m -Dbamboo.home=/path/to/bamboo-tomcat-home  
-Xmx512m -Djava.awt.headless=true -D<your-parameter>=<value> $JAVA_OPTS"  
export JAVA_OPTS  
...
```

Finding Your Bamboo Support Entitlement Number (SEN)

Your Support Entitlement Number (SEN) is required when raising a support request in our support system: <http://support.atlassian.com>.

See [Finding Your Support Entitlement Number](#) in the support space for more general information about how Atlassian Support uses this number.

The three ways of finding you SEN are described below.

On this page:

- [Method 1 — Check the Bamboo Administration Interface](#)
- [Method 2 — Check my.atlassian.com](#)
- [Method 3 — Check your Atlassian Invoice](#)

Method 1 — Check the Bamboo Administration Interface

To find your SEN via the Bamboo administration interface:

1. Click **Administration** in the top menu bar of Bamboo.
2. Click **License Details** in the left navigation panel (under 'System'). The SEN is shown, as in the screenshot below.

Screenshot: SEN in the Bamboo administration console

License Key Details

Existing License Key Details

You may view your licensing details or use the Update License form to update the license Bamboo is running with.

Organisation Atlassian

Date Purchased 26 June 2010

License Type Bamboo Enterprise: Developer

Number of local agents supported Unlimited

Number of remote agents supported 25

Support Period Lasts Until 27 June 2011

Server ID [REDACTED]

Support Entitlement Number (SEN) [REDACTED]

Update License

License Key

Please enter your Bamboo license key above - either commercial or evaluation.
Please contact [Atlassian](#) if you require a license key.

Save New License

Method 2 — Check my.atlassian.com

To find your SEN via my.atlassian.com:

1. Log into my.atlassian.com as the Account Holder or Technical Contact for your Bamboo product.
2. The SEN will be shown, as per the screenshot below.

Screenshot: SEN in my.atlassian.com



[Atlassian Home](#)
[Documentation](#)

[Licenses](#)
[Order Centre](#)
[My Profile](#)
[Dow](#)

Welcome

Hey Andrew Lui, welcome to the Atlassian customer portal. For questions relating to managing your account, [purchasing and licensing FAQ](#) or [contact a customer service representative](#). To request technical support, pl

Licenses

[New Evaluation License](#)

You do not have any product licenses.

Evaluations

[New Evaluation License](#) | [Expand All](#) | [Collapse All](#)

Product	Name
☐ Bamboo 25 Remote Agents: Evaluation	Atlassian
Server ID: XXXXXXXXXX SEN  : SEN- XXXXXXXXXX	License: AAABEw00 D9bte211 /i995K7+ 2ga/reYc IuvV4PaR 3BGA1aJ/

Method 3 — Check your Atlassian Invoice

Your Support Entitlement Number (SEN) appears on the third page of your Atlassian Invoice.

Configuring Gravatar support

Bamboo is configured to support Gravatars by default. This means that Bamboo will attempt to use user's emails to retrieve profile pictures from the [Gravatar service](#). The profile pictures will be displayed against user activity, e.g. comments, in Bamboo.

Related pages:

- [System settings](#)

To enable (or disable) Gravatar support:

1. Click **Administration** in the top navigation bar.
2. Click **General Configuration** in the left navigation panel.
3. Select (or clear) the **Enable Gravatar Support** checkbox, as required.
4. Click **Save**.

Agents and capabilities

A Bamboo *agent* is a service that can run [job builds](#). There are two types of Bamboo agents:

- *local agents* run as part of the Bamboo server.
- *remote agents* run on computers, other than the Bamboo server, that run the [remote agent](#) tool. An [elastic agent](#) is a remote agent that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#).

 Local agents run in the Bamboo server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.

A *capability* is a feature of an [agent](#). A capability can be defined on an agent for:

- an executable (e.g. Maven)
- a JDK
- a Version Control System client application (e.g. Git)
- a custom capability. This is a key-value property which defines a particular characteristic of an agent (e.g. 'operating.system=WindowsXP' or 'fast.builds=true').

Capabilities typically define the path to an executable that has already been installed, and must be defined in Bamboo before Bamboo or its agents can make use of those.

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

See [Configuring capabilities](#) for more information.

Bamboo matches job requirements against agent requirements when allocating jobs to agents.

Viewing an agent

To view an agent, including the agent properties, capabilities and the plans that an agent can build, see [Viewing a Bamboo agent's details](#).

Viewing the status of all agents

To view the status of all of your agents, see [Monitoring agent status](#).

Related pages:

- [Configuring agents](#)
- [Configuring capabilities](#)
- [Remote agents](#)

Viewing the agents and plans related to a capability

To view the agents and plans related to a capability, see [Viewing a capability's agents and jobs](#).

Configuring agents

A Bamboo *agent* is a service that can run [job builds](#). There are two types of Bamboo agents:

- *local agents* run as part of the Bamboo server.
- *remote agents* run on computers, other than the Bamboo server, that run the [remote agent](#) tool. An [elastic agent](#) is a remote agent that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#).

 Local agents run in the Bamboo server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.

If you are looking for information on **elastic agents**, please refer to the documentation on [Working with Elastic Bamboo](#).

On this page:

- [Creating a new agent](#)
- [Editing an agent's details](#)
- [Configuring an agent's capabilities](#)
- [Disabling or deleting an agent](#)
- [Notes](#)

Creating a new agent

To create a new agent, see:

- [Creating a local agent, or](#)
- [Creating a remote agent.](#)

Editing an agent's details

To edit an existing agent's details, see [Editing an agent's details](#).

Configuring an agent's capabilities

To configure an existing agent's capabilities, see:

- [Configuring capabilities](#)
- [Configuring remote agent capabilities](#)

Disabling or deleting an agent

To disable or delete an agent, see [Disabling or deleting an agent](#).

Notes

- A *capability* is a feature of an [agent](#). A capability can be defined on an agent for:
 - an executable (e.g. Maven)
 - a JDK
 - a Version Control System client application (e.g. Git)
 - a custom capability. This is a key-value property which defines a particular characteristic of an agent (e.g. 'operating.system=WindowsXP' or 'fast.builds=true').

Capabilities typically define the path to an executable that has already been installed, and must be defined in Bamboo before Bamboo or its agents can make use of those.

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

Viewing a Bamboo agent's details

A Bamboo *agent* is a service that can run [job builds](#). There are two types of Bamboo agents:

- *local agents* run as part of the Bamboo server.
- *remote agents* run on computers, other than the Bamboo server, that run the [remote agent](#) tool. An [elastic agent](#) is a remote agent that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#).

 Local agents run in the Bamboo server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.

On this page:

- [Viewing an agent's details](#)
- [Viewing the agents that can build jobs](#)

Viewing an agent's details

To view an agent's details:

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left panel to display the 'Agents' page, which lists all local and remote agents that currently exist in your Bamboo system.
3. Click the name of the desired agent. The agent's page will be displayed.
4. Click one of the following tabs to see corresponding details for the agent:

Tab	Description	
Capabilities	Displays a list of all 'Agent-Specific Capabilities' and 'Shared Capabilities' (see screenshot below). The capabilities in each of those sections are grouped into the following subsections: • Custom — custom capabilities. • Executable — executable capabilities. • JDK — JDK capabilities. • Perforce, Mercurial, Git — VCS capability.  You'll only see a subsection if a capability of that type is defined in Bamboo. To define a new capability, see Configuring capabilities.	
Executable Jobs	Displays a list of jobs, arranged by plan, that the agent can build.	
System Properties	Displays information about the agent.	
Audit Logs	Displays a record of changes that have been made to the agent.	

Agents › Local Agent (local)

Information

73%
5457 / 7464
Successful

Current status Idle

[View Recent Activity](#) | [Edit Details](#) | [Disable](#) | [Request Agent Stop](#) | [Stop Immediately](#) | [Delete](#)

[Capabilities](#) | [Executable Jobs](#) | [System Properties](#) | [Audit Logs](#)

Tabs

Agent-Specific Capabilities

[Add Capability](#)

A capability is a feature of an agent. There are 3 types of capabilities: executables, JDKs and custom.

No agent-specific capabilities currently exist.

Shared Capabilities

[Edit](#)

The following 'shared' capabilities are inherited by this agent. Note that the value of a shared capability is overridden by the value of an agent-specific capability with the same key/label (if one exists).

Custom

'Custom' capabilities are key-value pairs that define particular characteristics of an agent (e.g. 'operating.system=WindowsXP', 'fast.builds=true'). For an agent to be able to build a job, both the 'Key' and 'Value' must match the jobs's requirements.

Key	Value	Operations
\${first.second}	boo	View Edit
crowd.functionalTests	true	View Edit
first.second	hoo	View Edit
hasClover	true	View Edit
local	true	View Edit
release	true	View Edit

Executable

'Executable' capabilities define the executables which are available to your build plans.

Executable Label	Path	Operations
A label to uniquely identify this executable	Please enter the path to your executable	
Ant (Ant)	/usr/share/java/ant-1.8.2	View Edit
Bash (Command)	/bin/bash	View Edit

Viewing the agents that can build jobs

To view which agents can build which jobs:

1. Click **Administration** in the top navigation bar.
2. Click **Agent Matrix** in the left navigation panel.
3. If an agent is not capable of building a particular job, hover your mouse over the cross to see the job requirements that are not met.

Creating a local agent

A Bamboo *agent* is a service that can run [job builds](#). There are two types of Bamboo agents:

- *local agents* run as part of the Bamboo server.
- *remote agents* run on computers, other than the Bamboo server, that run the [remote agent](#) tool. An [elastic agent](#) is a remote agent that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#).

Local agents run in the Bamboo server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.

Note that one local agent, with the default name of 'Default Agent', is automatically created after installing Bamboo.

Related pages:

- [Configuring agents](#)
- [Bamboo remote agent installation guide](#)

To create a new local agent:

1. Click **Administration** in the menu bar.
2. Click **Agents** in the left panel (under 'Build Resources') to display a list of all local and remote agents that currently exist in your Bamboo system.
3. Click **Add Local Agent**.
4. Enter details for the agent. The name is displayed on the [dashboard](#). The description is only visible to administrators.
5. Click **Add**.

Note that your new local agent:

- will be enabled by default.
- will inherit all [local server capabilities](#) that are defined in your Bamboo system.
- will be able to run builds for all jobs whose requirements are met by the agent's capabilities (see [Configuring a job's requirements](#)).

Screenshot: Creating a local agent

Add Local Agent

Enter a new unique name and a description for this local agent.

Information

Name

Description

Editing an agent's details

A Bamboo *agent* is a service that can run [job builds](#). There are two types of Bamboo agents:

- *local agents* run as part of the Bamboo server.
- *remote agents* run on computers, other than the Bamboo server, that run the [remote agent](#) tool. An [elastic agent](#) is a remote agent that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#).

i Local agents run in the Bamboo server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.

Each agent has a defined set of [capabilities](#) and can only run builds for jobs whose [requirements](#) match the agent's capabilities.

On this page:

- [Editing an agent's name or description](#)
- [Editing an agent's capabilities](#)

Related pages:

- [Configuring capabilities](#)
- [Defining a new executable capability](#)
- [Configuring an agent-specific JDK capability](#)
- [Defining a new custom capability](#)

Editing an agent's name or description

To edit an agent's name or description:

1. Navigate to the desired agent, as described on [Viewing a Bamboo agent's details](#).
2. Click **Edit Details**.
3. Update the details for the agent.
4. Click **Save**.

Editing an agent's capabilities

i To edit an agent's capabilities, see:

- [Defining a new executable capability](#)
- [Configuring an agent-specific JDK capability](#)
- [Defining a new custom capability](#)

Screenshot: Editing the details of an agent

Edit Details - Test Agent

Update the details of this agent, then click Update.

Information

Name

Description

[Cancel](#)

Disabling or deleting an agent

Bamboo allows you to disable or delete an agent, to prevent that agent from running any further builds.

- **Disabling an agent** lets you keep the agent in Bamboo, but stops it from running builds.

 If you need to prevent Bamboo from building any plans at all (e.g. while you re-index Bamboo), you can disable all agents. By doing so, all builds will wait in the queue until you re-enable the agents.

- **Deleting an agent** removes it from Bamboo altogether. If you need to use the agent again in future, you will need to recreate it (see [Creating a local agent](#) and [Creating a remote agent](#) for more information).

Note that you can also delete/disable individual plans and/or their jobs. This prevents the plans and/or their jobs from being submitted to the build queue. See [Disabling or deleting a plan](#) and [Disabling or deleting a job](#).

Related pages:

- [Disabling or deleting a plan](#)
- [Disabling or deleting a job](#)
- [Creating a local agent](#)
- [Creating a remote agent](#)

To disable (or delete) an agent:

1. Click **Administration** in the menu bar.
2. Click **Agents** in the left panel to display the 'Agents' screen, which lists all agents that currently exist in your Bamboo system. The 'Status' column indicates which agents are currently enabled or disabled.
3. Select the check box for the agent (or agents) you wish to disable or delete.
4. Click the **Disable** (or **Delete**) button above the table.

Screenshot: Agent — Delete or Disable

Local Agents

[Server Capabilities](#)

Local agents run on the Bamboo server.

Select: [All](#), [None](#), [Idle](#), [Disabled](#) Action:

	Agent	Status	Operations
☐	Default Agent	 Idle	View Edit
☐	Test Agent	 Idle	View Edit

[Add Local Agent](#)

Monitoring agent status

You can monitor your agents' status to check that all agents are functioning as expected.

Online versus Offline agents:

- An 'Online' agent is an agent which is currently available for use by Bamboo. Local agents are always online, although remote agents may be either online or offline.
- An 'Offline' agent is a [remote agent](#) which has been registered with the Bamboo server, was online, but is now unavailable for builds because:
 - The Bamboo remote agent process (running on the remote hardware) was stopped.
 - The Bamboo server (for whatever reason) cannot communicate with the remote hardware that is running the Bamboo remote agent process.

Bamboo administrators can manually 'disable' an online agent to prevent it from being used in build generation. The agent will still be online and it can be 'enabled' at a later point in time. It is not possible to disable offline agents.

Related pages:

- [Creating a local agent](#)
- [Bamboo remote agent installation guide](#)

To monitor the status of your agents:

1. Click **Administration** in the menu bar.
2. Click **Agents** in the left panel. This will display the 'Agents' screen, showing lists of all local agents and all remote agents that currently exist in your Bamboo system (see screenshot below).
Agents can have one on the following statuses:

Status	Description
Idle	Available to execute builds.
Building	Currently executing a build.
Cancelling	Currently cancelling a Job build
Disabled	Not available to execute builds (see Disabling or deleting an agent)
Disabled - Building	Currently executing a build, but disabled so cannot execute further builds.
Disabled - Cancelling	Currently cancelling a build, and disabled so cannot execute further builds.

Note that to see the jobs that are currently being built, look at the **Current Activity** tab on the [dashboard](#).

Screenshot: Viewing the status of your agents

Agents

An agent is a service that executes Bamboo builds. You can use this page to view, add and delete agents. You can also use this [matrix](#) to determine which agents can execute which build plans.

Local Agents

[Server Capabilities](#)

Local agents run on the Bamboo server.

Select: [All](#), [None](#), [Idle](#), [Disabled](#) Action: [Delete](#) [Disable](#) [Enable](#)

	Agent	Status	Operations
☐	Local Agent	 Building - SANDBOX-RFIN-JOB1-22	View Edit
☐	Reporting agent	 Idle (Disabled)	View Edit

[Add Local Agent](#)

Remote Agents

[Shared Remote Capabilities](#)

Remote agents run on computers other than the Bamboo server.

[Online Remote Agents](#) [Offline Remote Agents](#)

There are currently 3 remote agents online (3 elastic). Start [elastic agents here](#). A maximum of 25 agents are [supported](#) by your license.

Select: [All](#), [None](#), [Idle](#), [Disabled](#) Action: [Delete](#) [Disable](#) [Enable](#)

	Agent	Status	Operations
☐	 Elastic Agent on i-01bcc86d	 Idle	View Edit
☐	 Elastic Agent on i-03bcc86f	 Idle	View Edit
☐	 Elastic Agent on i-1fbcc873	 Building - BTS-FUNC-JOB1-7816	View Edit

```
Feb 11, 2011 10:13:43 AM A remote agent is loading on ip-10-204-186-247.ec2.internal (172.20.6.108).
Feb 11, 2011 10:13:44 AM A remote agent is loading on ip-10-102-61-209.ec2.internal (172.20.6.108).
Feb 11, 2011 10:13:45 AM A remote agent is loading on ip-10-102-61-80.ec2.internal (172.20.6.108).
Feb 11, 2011 10:13:49 AM A remote agent is loading on ip-10-102-61-238.ec2.internal (172.20.6.108).
Feb 11, 2011 10:13:50 AM A remote agent is loading on ip-10-204-185-205.ec2.internal (172.20.6.108).
Feb 11, 2011 10:14:00 AM A remote agent is loading on ip-10-204-185-229.ec2.internal (172.20.6.108).
Feb 11, 2011 10:15:40 AM Remote agent "Elastic Agent on i-01bcc86d" has registered.
Feb 11, 2011 10:15:43 AM Remote agent "Elastic Agent on i-03bcc86f" has registered.
Feb 11, 2011 10:15:50 AM Remote agent "Elastic Agent on i-1fbcc873" has registered.
```

These are live logs that are refreshed every 10 seconds, you will need to refresh the page to see any other updates.

[Install Remote Agent](#) | [Disable Remote Agent Support](#)

Configuring capabilities

A *capability* is a feature of an [agent](#). A capability can be defined on an agent for:

- an executable (e.g. Maven)
- a JDK
- a Version Control System client application (e.g. Git)

- a custom capability. This is a key-value property which defines a particular characteristic of an agent (e.g. 'operating.system=WindowsXP' or 'fast.builds=true').

Capabilities typically define the path to an executable that has already been installed, and must be defined in Bamboo before Bamboo or its agents can make use of those.

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

On this page:

- [Defining a new capability](#)
- [Editing a capability](#)
- [Renaming a capability](#)
- [Deleting a capability](#)
- [Notes](#)

Related pages:

- [Configuring agents](#)

Defining a new capability

To define a new capability, see:

- [Defining a new executable capability](#)
- [Defining a new JDK capability](#)
- [Defining a new custom capability](#)
- [Defining a new version control capability](#)

Editing a capability

To edit an existing capability, see [Modifying and deleting capabilities](#).

Renaming a capability

To rename an existing capability, see [Renaming a capability](#).

Deleting a capability

To delete a capability, see [Modifying and deleting capabilities](#).

Notes

- A *requirement* is specified in a [job](#) or a [task](#). A requirement specifies a [capability](#) that an [agent](#) must have for it to [build](#) that job or task. A job inherits all of the requirements specified in its tasks.

Together, capabilities and requirements control which agents can execute builds for particular [jobs](#). Each job can only be built by agents whose capabilities match the job's requirements. See [Configuring a job's requirements](#) for more information.

About capabilities and requirements

A *capability* is a feature of an [agent](#). A capability can be defined on an agent for:

- an executable (e.g. Maven)
- a JDK
- a Version Control System client application (e.g. Git)
- a custom capability. This is a key-value property which defines a particular characteristic of an agent (e.g. 'operating.system=WindowsXP' or 'fast.builds=true').

Capabilities typically define the path to an executable that has already been installed, and must be defined in Bamboo before Bamboo or its agents can make use of those.

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

See [Configuring capabilities](#) for more information.

On this page:

- [How do capabilities work with requirements?](#)
- [How are builds distributed to agents?](#)
- [How do capabilities affect the distribution of builds to agents?](#)

Related pages:

- [Configuring capabilities](#)
- [Configuring agents](#)
- [Remote agents](#)
- [Working with Elastic Bamboo](#)

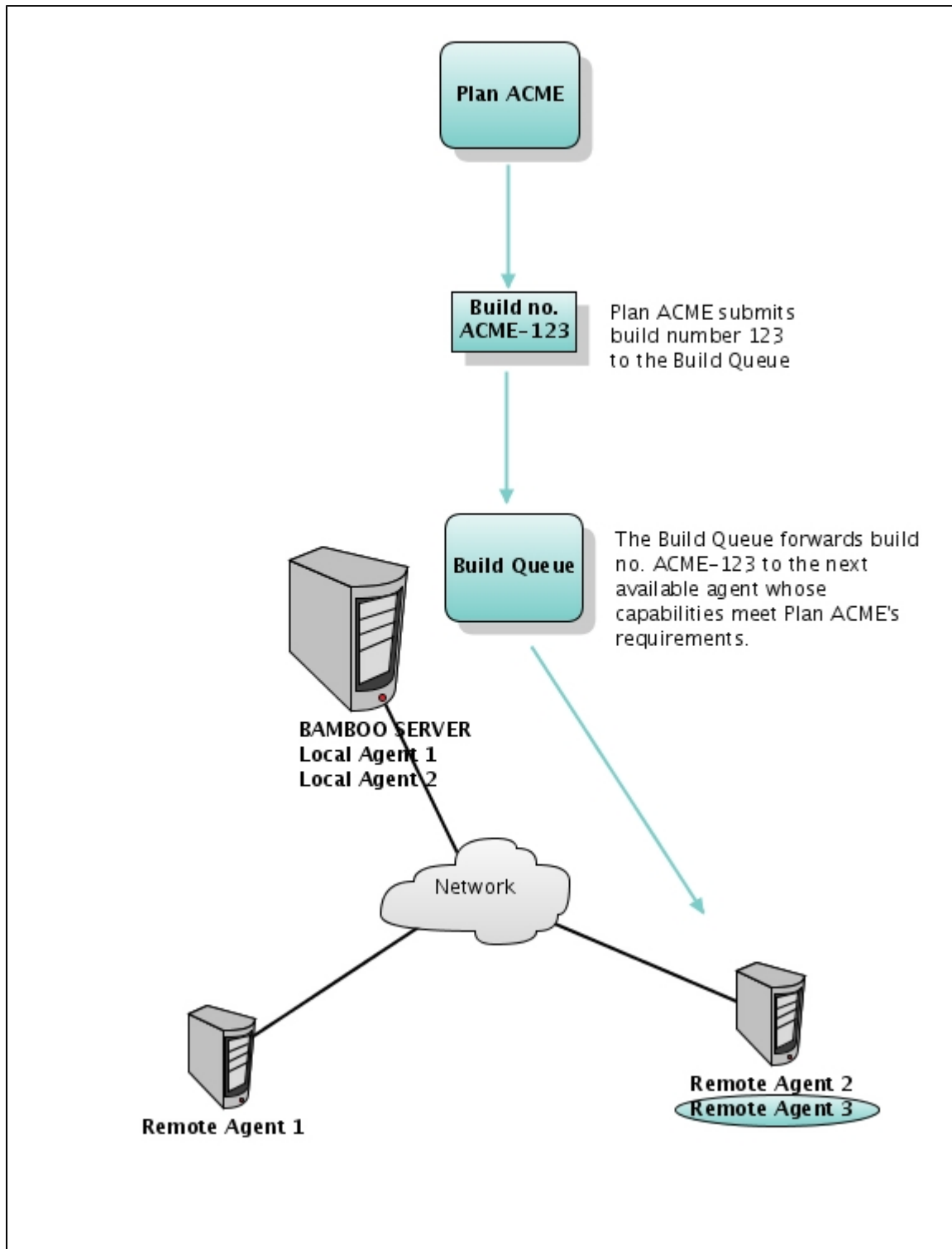
How do capabilities work with requirements?

A *requirement* is specified in a [job](#) or a [task](#). A requirement specifies a [capability](#) that an [agent](#) must have for it to [build](#) that job or task. A job inherits all of the requirements specified in its tasks.

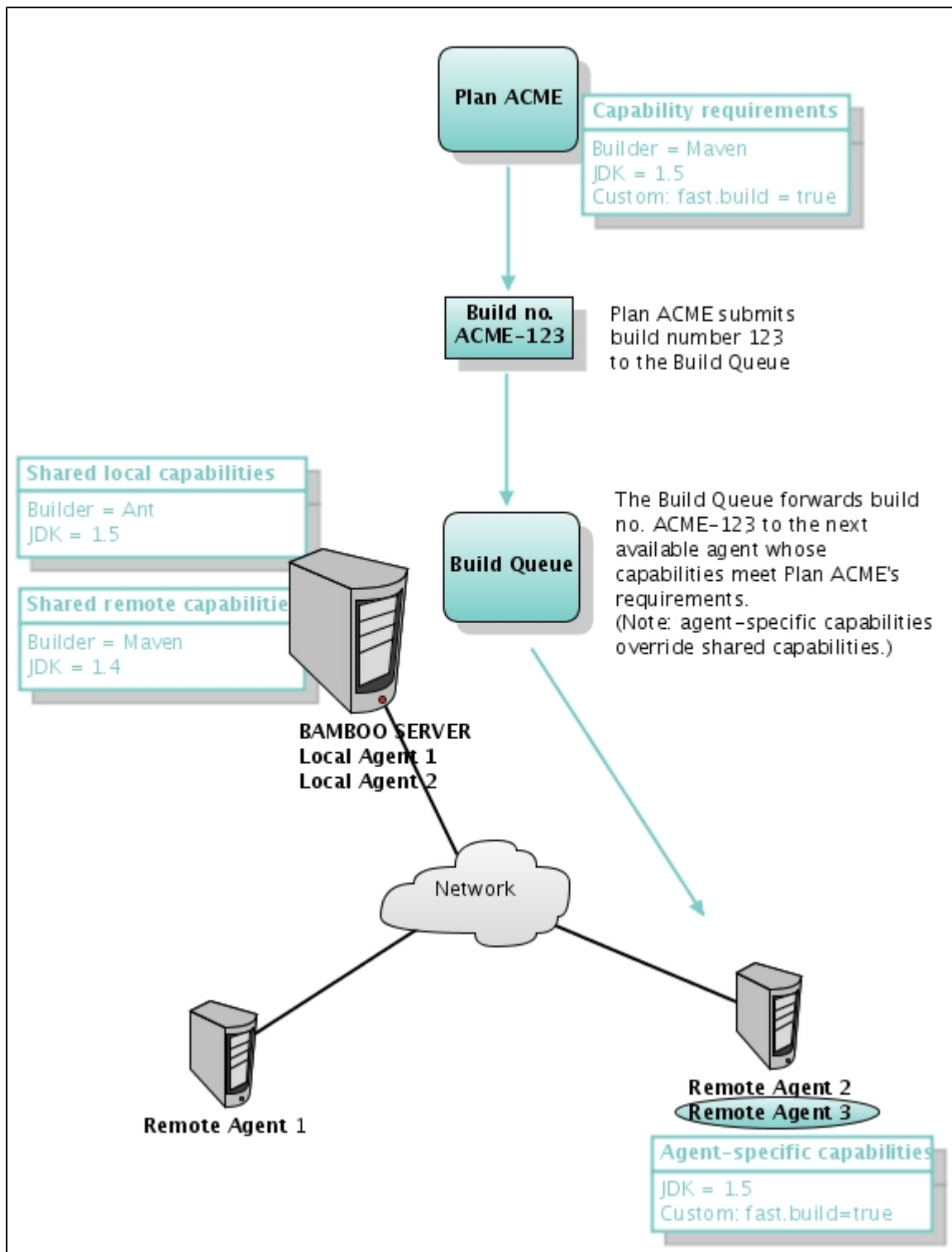
Together, capabilities and requirements control which agents can execute builds for particular [jobs](#). Each job can only be built by agents whose capabilities match the job's requirements. See [Configuring a job's requirements](#) for more information.

How are builds distributed to agents?

An agent will consume a single job at a time and will block any other Bamboo jobs from being processed until that job build is complete. If you would like to build multiple jobs simultaneously on the Bamboo server, then simply [set up multiple local agents](#). If the agents are remote, then you will need to [install that number of agent instances](#) on the machine. Separate installations are required because each remote agent will need its own home and log directories.



How do capabilities affect the distribution of builds to agents?



Viewing a capability's agents and jobs

You can view a capability to see the following information about it:

- which [agents](#) have/inherit the capability. Click one of the listed agents to show further information about that agent:
 - **Executable Jobs** tab — all the jobs whose requirements match the capabilities of this agent
 - **Capabilities** tab — the capabilities of the agent itself
 - **System Properties** tab — system information about this agent
 - **Recent Activity** link — recent builds for the agent
- which [jobs](#) have the capability specified as a [requirement](#).
- which elastic images have this capability and the Bamboo plans that rely on this capability. See also [Viewing an elastic image](#).

On this page:

- [Viewing an agent-specific capability](#)
- [Viewing a local server capability](#)
- [Viewing a shared remote capability](#)

Related pages:

- [Configuring capabilities](#)
- [Renaming a capability](#)
- [Modifying and deleting capabilities](#)

Viewing an agent-specific capability

To view an agent-specific capability:

1. Navigate to the desired agent, as described on [Viewing a Bamboo agent's details](#).
2. Click the **Capabilities** tab.
3. Click **View** for the capability you wish to view.

Viewing a local server capability

To view a local server capability:

1. Click **Administration** in the top navigation bar.
2. Click **Server Capabilities** in the left navigation panel.
3. Click **View** for the capability you wish to view.

Viewing a shared remote capability

Before you begin:

- Shared remote capabilities are **not shared** with [elastic agents](#).

To view a shared remote capability:

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left navigation column.
3. Click **Shared Remote Capabilities** in the top right of the 'Remote Agents' section.
4. Click **View** for the capability you wish to view.

Defining a new executable capability

An **executable** is an external program that Bamboo uses during the build process. Bamboo supports the following executables:

- Ant
- Maven
- Grails
- NAnt
- devenv.com
- msbuild.exe
- PHPUnit
- Custom command (e.g. 'make')
- Script

On this page:

- [Defining an agent-specific executable capability](#)
- [Defining a local server executable capability](#)
- [Defining a shared remote executable capability](#)
- [Notes](#)

Executables must be defined as capabilities (that is, 'registered') in Bamboo before they can be used by tasks in a Bamboo job. At least one capability was automatically defined when you installed Bamboo, but you can define additional capabilities for other executables.

You can define an executable capability that is:

- for a [specific](#) local or remote agent
- shared by all [local](#) agents
- shared by all [remote](#) agents.

Once you have defined a new executable capability in your Bamboo system, its label (e.g. 'Ant') will appear in the **Executable** list when you use the executable in a task (see [Configuring tasks](#)). The executable you select will be used every time the task is run during a build. That is, the task can only be run by agents which have a capability that matches the executable specified in the task's **Executable** list.

Note that agent-specific capabilities override any shared capability of the same name.

Defining an agent-specific executable capability

An *agent-specific capability* applies to one agent only. Note that the value of an agent-specific capability will override the value of a *shared capability* of the same name (if one exists).

To define a new agent-specific executable capability:

1. Navigate to the desired agent, as described on [Viewing a Bamboo agent's details](#).
2. In the 'Agent-Specific Capabilities' section of the **Capabilities** tab, click **Add Capability**. The 'Add Capability' page is displayed.
3. Choose **Capability Type > Executable**.
4. Select the appropriate executable from the **Type** list.
5. In the **Executable Label**, type a name/label for the executable. Bamboo uses this name in the **Executables** list whenever a [task's executable is configured](#).
6. In the **Path** field, type the path to the installed executable. This will vary depending on the **Type** you selected in the previous step.
 For Ant and Maven, Bamboo requires the path to be the location of the executable installation folder.
7. Click **Add**. This will verify whether the executable and path you have specified are valid.

Defining a local server executable capability

Local server capabilities are inherited by all local agents. This means that local agents can all make use of the executables installed on the Bamboo server machine.

Before you begin:

- If you want to run multiple Maven agents on your local server, you will need to configure repository isolation for your Maven executables. See [Configuring repository isolation for Maven executables](#) for details.

To define a new local server executable capability:

1. Click **Administration** in the top navigation bar.
2. Click **Server Capabilities** in the left navigation panel.
3. Choose **Capability Type > Executable** in the 'Add Capability' section at the end of the page (see [screenshot below](#)).
4. Select the appropriate type of executable from the **Type** list.
5. In the **Executable Label** field, type a name/label for the executable, which Bamboo presents in the **Executable** list whenever a [Task's executable is configured](#).
6. In the **Path** field, type the appropriate path. This will depend on the **Type** you selected in the previous step.
 For Ant and Maven, Bamboo requires the path to be the location of the executable installation folder.
7. Click **Add**.

Defining a shared remote executable capability

Shared remote capabilities are inherited by all remote agents. However, Bamboo remote agents inherit only the paths of the shared executable capabilities, not the actual executable files. This means that every time you define a capability for an agent, you must make sure that the executable (for example, Ant or Maven) has actually been installed in that location on the remote server on which the remote agent will run.

Note that the value of a shared capability will be overridden by the value of an *agent-specific capability* of the same name (if one exists).

Shared remote executable capabilities are **not shared** with [elastic agents](#).

To define a shared remote executable capability:

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left navigation panel.
3. In the 'Remote Agents' section, click **Shared Remote Capabilities** at the right.
4. Choose **Capability Type > Executable** in the 'Add Capability' section (see the [screenshot below](#)).
5. Select the appropriate type of executable from the **Type** list.
6. In the **Executable Label** field, type a name/label to help you identify this executable.
7. In the **Path** field, type the appropriate path. This will depend on the **Type** you selected in the previous step.
 For Ant and Maven, Bamboo requires the path to be the location of the executable installation folder.
8. Click **Add**.

Notes

- **Pre-defined executables** — The executable that was automatically defined when you installed Bamboo depends on the system environment variables (e.g. 'ANT_HOME=/opt/java/ant') that were present on the machine that Bamboo was installed on.
 - On the Bamboo server, environment variables that were present during installation were saved as *local server capabilities* in Bamboo.
 - On remote agents, environment variables that were present during installation were saved as *agent-specific capabilities* in Bamboo.
- **Using other executables** — If you need to use an executable that is not natively supported by Bamboo, a number of [third-party plugin modules](#) are available (e.g. NoseXUnit). You can also create your own executable plugin (see the [Bamboo Plugin Guide](#) for details).
- **msbuild.exe** — You will need to install the .NET framework SDK and reference the default path for msbuild.exe, (e.g. C:\Windows\Microsoft.NET\Framework*64*\v2.0.50727), to use this executable.
- **PHPUnit** — You will need to [install PHPUnit](#) and reference the path to your PHP command-line interpreter, (e.g. /usr/bin/phpunit on Ubuntu), to use this executable.

Viewing your executable capabilities

You can view all of the executable capabilities that have been defined in Bamboo on the 'Executables' page. These include local server capabilities, local agent-specific capabilities and remote agent-specific capabilities.

An *executable* is an external program that Bamboo uses during the build process. Generally, executables compile source code to generate compiled executable files (referred to as *artifacts* in Bamboo). *Ant*, *Maven*, *MSBuild* or *PHPUnit* are just some examples of executables that can be used as part of your build process.

New executables can be defined as *capabilities* in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a task.

On this page:

- [Viewing and configuring executable capabilities](#)
- [Notes](#)

Related pages:

- [Configuring capabilities](#)

Viewing and configuring executable capabilities

To view and configure the executable capabilities defined in Bamboo:

1. Click **Administration** in the top navigation bar.
2. Click **Executables** in the left navigation panel.
3. Click a specific executable's tab to see the agents and jobs related to this executable capability.
 - [View more details about an agent with this executable capability](#) — click the linked name of the agent in the 'Agent' column. This will show you the complete list of capabilities and jobs associated with that agent.
 - Edit the executable path of an agent with this capability — click **Edit** in the 'Operations' column for the agent you wish to configure. See [Defining a new executable capability](#).
 - Remove this executable capability from an agent — click **Delete** in the 'Operations' column for the agent that currently possesses this executable capability.
 Be aware that you can only remove a executable capability from *all* local agents, not from individual local agents. See [the note below](#) for more information.
 - [View details about \(and configure\) an elastic image with this executable capability](#) — click the linked name of the elastic image in the 'Elastic Image Configuration' column.
 - [Configure a job that relies on or requires this executable capability](#) — click the linked name of the job in the 'Plan' column.
 - If you are currently viewing a Maven (2.x or later) executable capability, you can configure repository isolation for it by clicking **Edit Capability Configuration**. Please refer to [Configuring repository isolation for Maven executables](#) for more information.
 - To [add a new executable as a local server capability](#), click **Add executable to server capabilities** to navigate to the 'Server Capabilities' page.

Screenshot: Executables

Executable

The following table lists the executables that have been defined as capabilities in Bamboo. You can [add a executable as a server capability](#) on this page (shared by all local agents), or delete any of the existing executable capabilities listed. A executable can only be added as an agent-specific capability on that agent's page.

Ant (Ant)

Maven 2.1 (Maven 2.x)

Maven 2 (Maven 2.x)

Bash (Command)

Maven 1 (Maven 1.x)

Script (Script)

Ant

The screen shows the summary of a capability. You can see which jobs have a requirement on this capability and which agents have the capability.

Agents with capability

The following agents have this capability.

Agent	Path	Operations
All local agents	/Volumes/Pharlap/opt/java/tools/ant	Edit Delete
kozlak.atlassian.pl	/usr/share/ant	Edit Delete
tardigrade.sydney.atlassian.com	/opt/dev/tools/apache-ant-1.7.0	Edit Delete

Elastic Image Configurations with capability

2 elastic image configurations have this capability.

Elastic Image Configuration	Path
Default	/opt/apache-ant-1.7.1
Clean instance	/opt/apache-ant-1.7.1

Jobs with requirement

2 jobs rely on this capability.

Plan	Path	Operations
Personal Branches > Jess's test build > HGTest	exists	
Personal Branches > New Plan > Default Job	exists	

[Rename Capability](#)

Notes

- Bamboo's automatic detection of executables** — When you install the Bamboo server application or the Bamboo Remote Agent application on another machine, either of these applications will automatically look for existing executables installed on the same machine (based on a combination of the machine's environment variables and other conditions). A 'executable capability' will be created for each executable that either of these Bamboo applications find.

The environment variables and conditions that Bamboo uses to automatically detect and create executable capabilities are listed below. With the exception of the 'Command' executable, the paths for each automatically detected executable are based on the path 'string' values found within these environment variables.

- Ant — the ANT_HOME environment variable
- Maven — the MAVEN_HOME environment variable (Maven 1), M2_HOME or MAVEN2_HOME environment variable (Maven 2.x)
- Grails — GRAILS_HOME environment variable
- Command — the existence of the /bin/bash file
- PHPUnit — the existence of the phpunit file anywhere within the machine's PATH environment variable value
- Local agents and executable capabilities** — Since Bamboo automatically looks for executables installed on the same machine and creates a 'executable capability' for each executable installation it finds, all existing and subsequent local agents that you create will possess these executable capabilities. Hence, when you access the 'Executables' page and view these executable capabilities, all local agents will be grouped together in the 'All local agents' category and you will only be able to remove these executable capabilities from *all* local agents, not from individual local agents.

Configuring repository isolation for Maven executables

Bamboo allows you to isolate Maven (2.x or later only) executables on an agent-specific basis. If you configure repository isolation for a particular Maven executable capability, each agent that uses this executable will have its own private Maven 2.x artifacts directory, thereby allowing you to avoid these jar and dependency file corruptions. Each isolated repository directory has the path:

```
$BAMBOO_HOME/.m2/AGENT-${bamboo.agentId}/repository
```

Related pages:

- [Defining a new executable capability](#)

You may want to configure repository isolation for Maven executables, if you run multiple Maven executables on one server machine which *run under the same user account on that server*, but belong to *different Bamboo agents*. In this case, the agents will use the same default Maven artifacts directory: `$HOME/.m2/repository` (or `%USERPROFILE%\.m2\repository` for Windows-based servers). This is the directory to which Maven dependency jars are downloaded and where project artifacts are installed during the "install" phase of a Maven build.

Hence, problems can arise if Bamboo uses these multiple Maven executables simultaneously. For example, if multiple agents on a single computer, each with a different Maven executable capability, start to run Maven builds simultaneously from the queue, the different Maven executables may attempt to download the same dependency to the same artifacts directory location, resulting in corruption of the downloaded jar and dependency files.

Before you begin:

- This feature is not available for Maven 1.x executables.
- When configuring any Maven executables in Bamboo in which you want to force local repository isolation, ensure that the executable label you use is one that identifies it as such — for example, 'Maven 2.x with local repository isolation'.

To configure a new local server Maven capability with repository isolation:

1. Click **Administration** in the top menu bar.
2. Click **Server Capabilities** in the left navigation panel.
3. In the 'Add Capability' section, choose your executable and enter its details as described:

Capability Type	Select Executable
Type	Select one of the Maven options (2.x or later)
Executable Label	Enter 'Maven with local repository isolation'  You can use any label you wish. However, it will help you and your Bamboo users if you enter an appropriate executable label that identifies this Maven 2.x executable as one that uses local repository isolation.
Path	Enter the path for your Maven executable

4. Click **Add**.
5. Click the label for the executable you have just added. The executable capability summary screen will be displayed (see '[Maven 2.x Executable](#)' [screenshot below](#)).
6. Click **Edit Capability Configuration**. The 'Configure Capability' screen will be displayed (see '[Maven 2.x Repository Isolation](#)' [screenshot below](#)).
7. Select the **Local repository isolation** check box.
8. Click **Save**.

Screenshot: Maven Executable

Maven 2 with local repository isolation

The screen shows the summary of a capability. You can see which jobs have a requirement on this capability and which agents have the capability.

Agents with capability

The following agents have this capability.

Agent	Path	Operations
All local agents	/Volumes/Pharlap/opt/devtools/maven	Edit Delete
bambooperf-sydney.atlassian.com	C:\Program Files\Java\apache-maven-2.1.0	Edit Delete

Elastic Image Configurations with capability

3 elastic image configurations have this capability.

Elastic Image Configuration	Path
KBTEST	/opt/maven-2.0
Default	/opt/maven-2.0
Maven 2.1 Image	/opt/maven-2.0

Jobs with requirement

28 jobs rely on this capability.

Plan	Path	Operations
Artifact Sharing Dogfooding › Artifact sharing › Consumer	exists	
Artifact Sharing Dogfooding › Artifact sharing › Final	exists	

Capability Configuration

Local repository isolation **Yes**

[Rename Capability](#) | [Edit Capability Configuration](#)

Screenshot: Maven Repository Isolation

Executable > Maven 2

Configure Capability

Update specific features of this capability.

☐ Local repository isolation
When enabled, executable will use per-agent private maven local repository.

[Save](#) [Cancel](#)

Defining a new JDK capability

A JDK must be installed, and defined in Bamboo as a capability, before Bamboo can make use of it when building jobs.

At least one JDK was automatically defined when you installed Bamboo. You can define additional JDK capabilities that are:

- for a [specific](#) local or remote agent
- shared by all [local](#) agents
- shared by all [remote](#) agents.

On this page:

- [Defining an agent-specific JDK capability](#)
- [Defining a local server JDK capability](#)
- [Defining a shared remote JDK capability](#)
- [Notes](#)

Once you have defined a new JDK capability in your Bamboo system, its label (e.g. '1.5') will appear in the **Build JDK** list when you configure a job's builder (see [Configuring tasks](#)). The JDK you select will be used for every one of that job's builds. That is, the job can only be built by agents which have a JDK capability whose label is specified in the job's **Build JDK** field.

i Note that if an agent has its own specific JDK capability, that value will override the value of a shared JDK capability of the same name (if one exists).

Defining an agent-specific JDK capability

To define a new agent-specific JDK capability:

1. Navigate to the desired agent, as described on [Viewing a Bamboo agent's details](#).
2. Click the **Capabilities** tab.
3. Click **Add Capability** (under 'Agent-Specific Capabilities').
4. Choose **Capability Type** > **JDK**.
5. In the **JDK Label** field, type a name/label for the JDK. Bamboo show this in the **Build JDK** list whenever a [job's builder is configured](#).
6. In the **Java Home** field, type the location of the JDK Home Directory.
7. Click **Add**.

Add Capability - Local Agent

You can add an agent-specific capability on this page. The value of this capability will override the value of a shared capability of the same name (if one exists).

Capability Type	JDK
JDK Label	JDK 1.6
Java Home	/System/Library/Frameworks/JavaVM.framework

Add **Cancel**

Defining a local server JDK capability

Local server capabilities are inherited by all local agents.

To define a new local server JDK capability:

1. Click **Administration** in the top navigation bar.
2. Click **Server Capabilities** in the left navigation panel.
3. Choose **Capability Type** > **JDK** in the 'Add Capability' section at the end of the page.
4. In the **JDK Label** field, type a name/label for the JDK. Bamboo displays this in the **Build JDK** list whenever a [job's builder is configured](#).
5. In the **Java Home** field, type the location of the JDK Home Directory.
6. Click **Add**.

Defining a shared remote JDK capability

Shared remote JDK capabilities are **not shared** with [elastic agents](#).

To define a new shared remote JDK capability:

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left navigation panel.
3. In the 'Remote Agents' section, click **Shared Remote Capabilities** at the right.
4. Choose **Capability Type** > **JDK** in the 'Add Capability' section at the end of the page (see [screenshot below](#)).

5. In the **JDK Label** field, type a name/label for the JDK. Bamboo displays this in the **Build JDK** list whenever a [job's builder is configured](#).
6. In the **Java Home** field, type the location of the JDK Home Directory.
7. Click **Add**.

Add Capability

Capability Type

JDK Label

Java Home

[Cancel](#)

Notes

- **Configuring generic JDK capabilities** — If you want to indicate that an agent is capable of running builds for a set of related JDKs (e.g. all point versions of JDK 1.5), you [set up generic JDK capabilities](#) to encompass these JDKs. For example, you can set up the following JDK capabilities for your Bamboo agent(s):
 - **JDK** (where 'JDK Label' = 'JDK' and 'Java Home' = '/usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is capable of running builds with **any** JDK requirement.
 - **JDK 1.5** (where 'JDK Label' = 'JDK 1.5' and 'Java Home' = '/usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is capable of running builds with a JDK 1.5 requirement or any point version of JDK 1.5, e.g. 1.5.0_07, 1.5.0_08, etc.
 - **JDK 1.5.0_07** (where 'JDK Label' = 'JDK 1.5.0_07' and 'Java Home' = '/usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is only capable of running builds with a JDK 1.5.0_07 requirement.
- If you wish to find redundant JDK capabilities, you can view the list of [JDK capabilities](#) set up in Bamboo and delete any unwanted JDK capabilities.
- **Automatically defined capabilities** – This depends on the system environment variables (e.g. 'JAVA_HOME=/opt/java/java_sdk1.5') that were present on the machine on which Bamboo was installed:
 - On the Bamboo server, environment variables that were present during installation were saved as shared local capabilities in Bamboo.
 - On remote agents, environment variables that were present during installation were saved as agent-specific capabilities in Bamboo.

Viewing your JDK capabilities

You can view all the JDK capabilities that have been defined in your Bamboo system on the **JDKs** page. These include local server capabilities, local agent-specific capabilities and remote agent-specific capabilities.

Note the following:

- **Bamboo's automatic detection of JDKs** — When you install either Bamboo or the Bamboo Remote Agent, it will automatically look for an existing JDK installed on the same machine (based on the machine's JAVA_HOME environment variable) and create a 'JDK capability' for that JDK installation, with its path being the value of JAVA_HOME.
- **Local agents and JDK capabilities** — Since Bamboo automatically looks for an existing JDK installed on the same machine and creates a 'JDK capability' for it, all existing and subsequent local agents that you create will possess this JDK capability. Hence, when you access the 'JDKs' page and view this JDK capability, all local agents will be grouped together in the 'All local agents' category and you will only be able to remove this JDK capability from *all* local agents, not from individual local agents.

Related pages:

- [Defining a new JDK capability](#)

To view and configure the JDK capabilities defined in Bamboo:

1. Click **Administration** in the top menu bar.
2. Click **JDKs** (under 'Build Resources') in the left navigation panel.
3. Click the tab for a specific JDK to see the agents and jobs related to this JDK capability.
 - View the capabilities and jobs associated with an agent with this JDK capability — click the linked name of the agent in the 'Agent' column. See [Viewing a capability's agents and jobs](#).
 - Edit JAVA_HOME for an agent — click **Edit** in the 'Operations' column for the agent you wish to configure. See [Defining a new JDK capability](#).

- Remove this JDK capability from an agent — click **Delete** in the 'Operations' column for the agent that currently possesses this JDK capability.
⚠ Be aware that you can only remove a JDK capability from *all* local agents, not from individual local agents. See [the note above](#) for more information.
- View details about (and configure) an elastic image with this JDK capability — click the name of the elastic image in the 'Elastic Image Configuration' column. See [Viewing an elastic image](#).
- [Configure a job that relies on this JDK capability](#) — click the name of the job in the 'Plan' column.
- To add a new JDK as a local server capability, click **add a JDK as a server capability** at the top of the page. This opens the 'Server Capabilities' page at the 'Add Capability' section, with the JDK selected as the **Capability Type**.

Screenshot: Viewing the JDKs in Bamboo

JDK 1.6

The screen shows the summary of a capability. You can see which jobs have a requirement on this capability and which agents have the capability.

Agents with capability

The following agents have this capability.

Agent	Java Home	Operations
All local agents	/System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home	Edit Delete
172.20.3.214	/System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/	Edit Delete
bambooperf-PC.sydney.atlassian.com	C:\Program Files\Java\jdk1.6.0_21	Edit Delete
bambooperf2.sydney.atlassian.com	C:\Program Files\Java\jdk1.6.0_22	Edit Delete
barnard.sydney.atlassian.com	/System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/	Edit Delete

Elastic Image Configurations with capability

14 elastic image configurations have this capability.

Elastic Image Configuration	Java Home
Clean instance	/opt/jdk-6
Default Image	/opt/jdk-6
Selenium Test	/opt/jdk-6
PostgreSQL 8.4	/opt/jdk-6
S3 Maven Repository	/opt/jdk-6
Default Image S3 i386	/opt/jdk-6
MySQL 5.1	/opt/jdk-6

Jobs with requirement

8 jobs rely on this capability.

Plan	Java Home	Operations
+core+ Bamboo > WebDriver Tests > Default Job	exists	
+core+ Bamboo Plugins > Bamboo Sandbox Plugin Dogfood Test > Dummy Private Deploy	exists	
+core+ Bamboo Plugins > Bamboo Sandbox Plugin Dogfood Test > Dummy Public Sandbox Deploy	exists	
Bamboo Plugins +non core+ > Support Tools Plugin > Integration Tests	exists	Edit Delete
Reporting > OpUp report > Default Job	exists	
Speakeasy > Trunk > Default Job	exists	
ZZ pstefaniak > clone of empty plan > Default Job	exists	
ZZ pstefaniak > empty plan for ssaasen > Default Job	exists	

[Rename Capability](#)

Defining a new version control capability

Version control capabilities let Bamboo know where the client application for a version control system is located, so that Bamboo can perform a checkout whilst building. Bamboo requires that a capability for at least one of the following version control repositories be set so that Bamboo can check out source code from that repository type:

- [Git](#) (If no capability is provided, Bamboo will use its built-in Git implementation. Note that the built-in Git implementation does not support symbolic links, submodules, automatic branch detection and automatic merging.)
- [Mercurial](#)
- [Perforce](#)

Note that there is no need to create a SVN capability as SVN support is built into every Bamboo agent.

Example version control executable paths

For the version control systems that require capabilities to be set on agents, the following table offers example paths for both Linux and Windows systems.

Note that these paths may differ on your actual system's configuration.

Capability type	Example paths
Git	• /usr/bin/git • C:\Program Files\Git\git.exe
Mercurial	• /usr/local/bin/hg • C:\Program Files\Mercurial\hg.exe
Perforce	• /usr/bin/p4 • c:\Program Files\Perforce Client\p4.exe

To define a new version control capability

1. Navigate to the desired agent, as described in [Viewing a Bamboo agent's details](#).
2. Select either a local or remote agent.
3. Choose the version control type you require from **Capability Type**.
4. Provide the full path to client executable on the agent machine.

i If you install a new agent on a machine that has Git already installed, the agent will find the Git client automatically.

Defining a new custom capability

Custom capabilities can be used to control which [jobs](#) will be built by a particular agent, since agent [capabilities](#) are required to match job [requirements](#). For example, if the builds for a particular job should only run in a Windows environment, you could create a custom capability 'operating.system=WindowsXP' for the appropriate agent(s), and specify it as a requirement for this job. (See [Configuring a job's requirements](#).)

You can define a custom capability that is:

- for a [specific local or remote agent](#)
- to be shared by all [local agents](#)
- to be shared by all [remote agents](#).

i Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

On this page:

- [Defining an agent-specific custom capability](#)
- [Defining a local server custom capability](#)
- [Defining a shared remote custom capability](#)

Defining an agent-specific custom capability

To define a new agent-specific custom capability:

1. Navigate to the desired agent, as described in [Viewing a Bamboo agent's details](#).
2. Click **Add Capability** in the top right of the 'Agent-Specific Capabilities' section.
3. Choose **Capability Type > Custom**.
4. Specify values for **Key** and **Value**.
5. Click **Add**.

Add Capability - Local Agent

You can add an agent-specific capability on this page. The value of this capability will override the value of a shared capability of the same name (if one exists).

Capability Type	Custom
Key	local
Value	true

[Cancel](#)

Defining a local server custom capability

Local server capabilities are inherited by all local agents.

To define a new local server custom capability:

1. Click **Administration** in the top navigation bar.
2. Click **Server Capabilities** in the left navigation panel (under 'Build Resources').
3. Locate the 'Add Capability' section at the bottom of the screen (see screenshot below).
4. Choose **Capability Type** > **Custom**.
5. Specify values for **Key** and **Value**.
6. Click **Add**.

Defining a shared remote custom capability

Shared remote custom capabilities are **not shared** with [elastic agents](#).

To define a new shared remote custom capability:

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left navigation panel (under 'Build Resources').
3. Locate the 'Remote Agents' section.
4. Click **Shared Remote Capabilities**.
5. Locate the 'Add Capability' section at the bottom of the screen (see screenshot below).
6. Choose **Capability Type** > **Custom**.
7. Specify values for **Key** and **Value**.
8. Click **Add**.

Add Capability

Capability Type	Custom
Key	local
Value	true

[Cancel](#)

Modifying and deleting capabilities

Depending on the capability type, you can edit parameters such as **Path**, **Java Home** and **Value** for the capability.

Note that:

- Because each agent can only run builds for jobs whose [requirements](#) are met by the agent's capabilities (see [Configuring a job's requirements](#)), modifying or deleting a capability may mean that some plans can no longer be built.
- Renaming a capability involves changing its key. See [Renaming a capability](#).

On this page:

- [Modifying an agent-specific capability](#)
- [Modifying a local server capability](#)
- [Modifying a shared remote capability](#)

Related pages:

- [About capabilities and requirements](#)
- [Configuring capabilities](#)
- [Renaming a capability](#)

Modifying an agent-specific capability**To delete an agent-specific capability:**

1. Navigate to the desired agent, as described in [Viewing a Bamboo agent's details](#).
2. Click either **Edit** or **Delete** for the capability you wish to modify.

Modifying a local server capability**To delete a local server capability:**

1. Click **Administration** in the top navigation bar.
2. Click **Server Capabilities** in the left navigation panel.
3. Click either **Edit** or **Delete** for the capability you wish to modify.

Modifying a shared remote capability**To delete a shared remote capability:**

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left navigation column.
3. Click **Shared Remote Capabilities** in the top right of the 'Remote Agents' section.
4. Click either **Edit** or **Delete** for the capability you wish to modify.

Renaming a capability

To rename a capability you have to change its key value.

Renaming an agent-specific capability**To rename a capability:**

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left panel (under 'Build Resources').
3. Click **View** for the agent that has the capability you wish to rename. A list of agent-specific capabilities and shared capabilities for that agent is displayed.
4. Click **View** for the capability you wish to rename.
5. Click **Rename Capability**. The 'Rename Capability' page will display.
6. Enter a value for **New key** and click **Rename Capability**.

On this page:

- [Renaming an agent-specific capability](#)
- [Renaming a local server capability](#)
- [Renaming a shared remote capability](#)

Related pages:

- [About capabilities and requirements](#)
- [Configuring capabilities](#)

Renaming a local server capability**To rename a local server capability:**

1. Click **Administration** in the top navigation bar.
2. Click **Server Capabilities** in the left panel (under 'Build Resources').
3. Click **View** for the capability you wish to rename.
4. Click **Rename Capability**. The 'Rename Capability' page will display.
5. Enter a value for **New key** and click **Rename Capability**.

Renaming a shared remote capability**To rename a shared remote capability:**

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left panel (under 'Build Resources').
3. Click **Shared Remote Capabilities** in the 'Remote Agents' section.
4. Click **View** for the capability you wish to rename.

5. Click **Rename Capability**. The 'Rename Capability' page will display.
6. Enter a value for **New key** and click **Rename Capability**.

Screenshot: Renaming a custom capability

Rename Capability

Update the label of this capability. Jobs with it as a requirement will be updated and all agents with this capability will be updated accordingly.

Capability Type **Custom**

Old Key **inelastic**

New key

Rename Capability [Cancel](#)

Remote agents

For information about installing and using remote agents, see the following pages:

- [Bamboo remote agent installation guide](#)
- [Configuring remote agent capabilities using bamboo-capabilities.properties](#)
- [Disabling and enabling remote agents support](#)

Disabling and enabling remote agents support

A Bamboo *agent* is a service that can run [job builds](#). There are two types of Bamboo agents:

- *local agents* run as part of the Bamboo server.
- *remote agents* run on computers, other than the Bamboo server, that run the [remote agent](#) tool. An [elastic agent](#) is a remote agent that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#).

i Local agents run in the Bamboo server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.

Each agent has a defined set of [capabilities](#) and can only run builds for jobs whose [requirements](#) match the agent's capabilities.

Related pages:

- [Configuring agents](#)
- [Agents and capabilities](#)
- [Configuring a job's requirements](#)

Remote agent support

Disabling remote agent support in Bamboo will disable all remote agents and prevent any users from creating new remote agents. This function will not delete any remote agents that you have already created. To delete a remote agent, see [Disabling or deleting an agent](#).

Note that remote agent support must be enabled to use [Elastic Bamboo](#). Disabling remote agent support will disable Elastic Bamboo.

To enable, or disable, remote agent support, go to the Admin area and click **Agents** (under 'Build Resources'). Click either **Enable Remote Agent Support** or **Disable Remote Agent Support**.

Screenshot: Disabling remote agent support

Remote Agents

[Shared Remote Capabilities](#)

Remote agents run on computers other than the Bamboo server.

☐ Online Remote Agents ☐ Offline Remote Agents

There are currently no online remote agents configured on this Bamboo instance.

[Install Remote Agent](#) | [Disable Remote Agent Support](#)

Agents

Add Local Agent

Enable Remote Agent Support

An agent is a service that executes Bamboo builds. You can use this page to view, add and delete agents. You can also use this [matrix](#) to determine which agents can execute which build plans.

⚠ Support for remote agents is disabled.
Before enabling the remote agent support, please carefully consider the [security implications](#).



Working with Elastic Bamboo

Elastic Bamboo allows you to use computing resources from the [Amazon Elastic Compute Cloud \(EC2\)](#) to run builds. Elastic Bamboo uses a [remote agent AMI \(Amazon Machine Image\)](#) to create [instances of remote agents](#) in the Amazon EC2.

The following pages and sub-pages describe how to work with Elastic Bamboo:

- [About Elastic Bamboo](#) — Elastic Bamboo concepts.
- [Getting started with Elastic Bamboo](#) — setting up Elastic Bamboo for the first time. It contains instructions on enabling Elastic Bamboo for your Bamboo installation and running your first build.
- [Configuring Elastic Bamboo](#) — changing settings for Elastic Bamboo. This includes instructions on how to use Amazon's Elastic Block Storage to persist build information for your builds on Elastic Bamboo.
- [Managing your elastic images](#)
- [Managing your elastic instances](#)
- [Managing your elastic agents](#)
- [Running job builds using Elastic Bamboo](#) — running your builds using Elastic Bamboo.
- [Elastic Bamboo Security](#) — setting up secure communication between Bamboo and the EC2.

About Elastic Bamboo

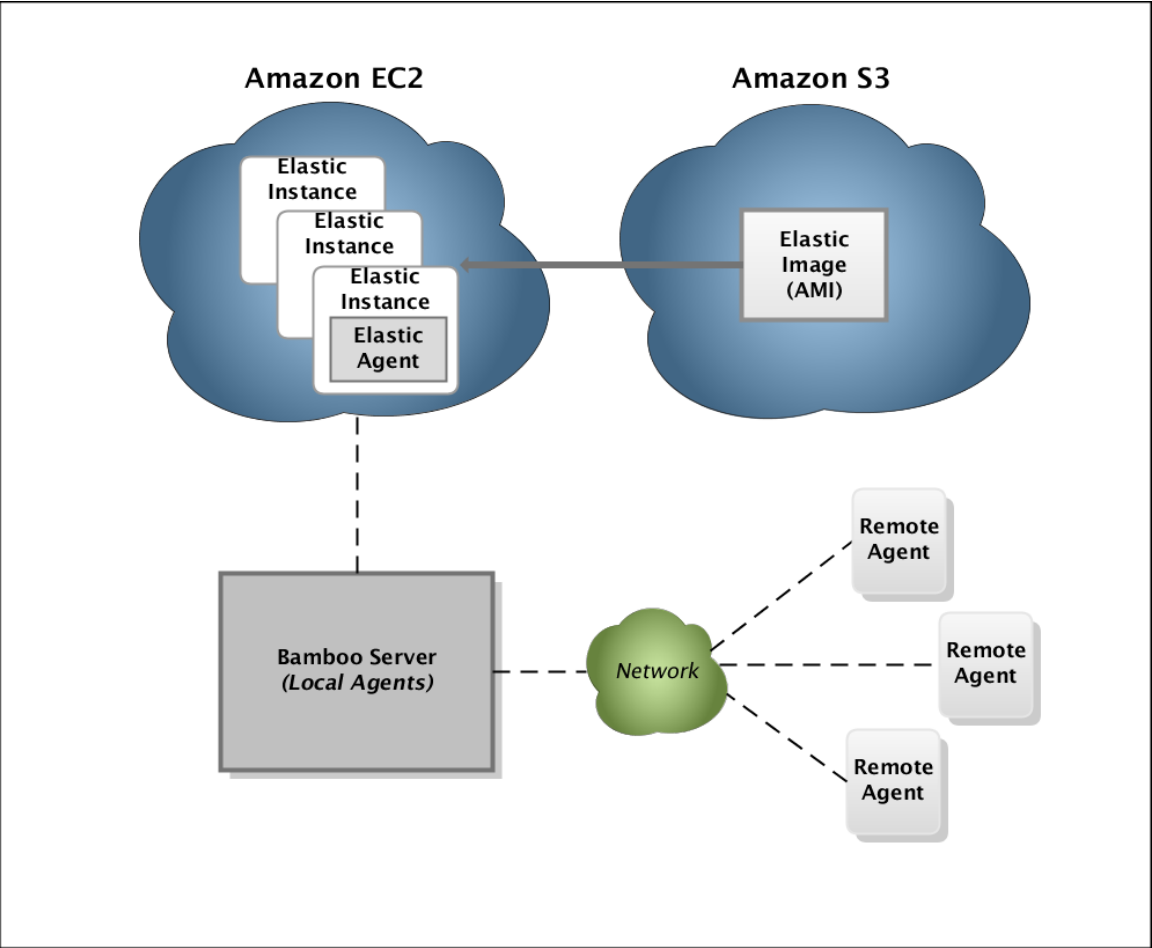
On this page:

- [Conceptual Overview](#)
- [Key Terms](#)
- [Setting Up Elastic Bamboo](#)

Conceptual Overview

Elastic Bamboo allows you to use computing resources from the [Amazon Elastic Compute Cloud \(EC2\)](#) to run builds. Elastic Bamboo uses a [remote agent AMI \(Amazon Machine Image\)](#) to create [instances of remote agents](#) in the Amazon EC2.

[Elastic Bamboo Conceptual Overview](#)



A Job's build can be run on an elastic agent, provided that the capabilities of the elastic agent meet the requirements of the Job. Bamboo will assign the relevant Job build to an available elastic agent from the build queue automatically, in the same way that Job builds are assigned to non-elastic agents. The elastic agent must already be running for a Job build to be assigned to it.

An elastic agent is started by creating a new instance of an elastic image. Creating this new elastic instance automatically runs an elastic agent process in the instance. The agent inherits the capabilities of the image it was created from. Only one agent process can be run in an instance, although multiple instances can be created from the same image.

Once a Job's build has completed running on an elastic agent, its results are made available (like those of any other Job build executed on a non-elastic agent). The elastic agent and instance will continue to run until they are shut down. Shutting down an elastic instance will terminate the agent, not take it offline. However, Bamboo will store historical information about the terminated elastic agent, such as the Job builds which it has run.

An [Amazon Web Services \(AWS\)](#) account is required to use Elastic Bamboo. [Elastic Bamboo Costs](#) are charged by Amazon, separate to Bamboo licence costs, as Elastic Bamboo is powered by Amazon resources.

🟢 Did you know you can configure Bamboo to start and shut down elastic instances **automatically**, based on build queue demands? Please refer to [Configuring Elastic Bamboo](#) for more information.

Key Terms

Elastic Image	An <i>elastic image</i> is an Amazon Machine Image (AMI) that is stored in one of Amazon data centres for use with the Elastic Bamboo feature. An elastic image is used to create elastic instances, which in turn create elastic agents. Conceptually, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system. Each elastic image registered with the Amazon Web Services (AWS) has its own unique identifier, known as an AMI ID. You can associate multiple elastic images with a Bamboo server. One default shared image is maintained by Atlassian in AWS, and is available to all Elastic Bamboo users. You can also create your own custom elastic images.

Elastic Instance	An <i>elastic instance</i> is a running instance of an elastic image. One elastic instance is created whenever an elastic image is started. Hence, starting one elastic image multiple times, results in the creation of multiple elastic instances. Each time an elastic instance is created, one elastic agent is created on that instance. Conceptually, an elastic instance can be thought of as a computer. The elastic agent's processes are run on this computer and the elastic image is the boot hard drive. Unlike computers, however, elastic instances are temporary and stateless. When an elastic instance is shut down: - Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) will not persist - Any customisations to the instance itself will also be lost.  The Amazon Elastic Block Store can provide persistent storage for your elastic instances.
Elastic Agent	An <i>elastic agent</i> is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2). An elastic agent process runs in an elastic instance of an elastic image. An elastic agent inherits its capabilities from the elastic image that it was created from.

Setting Up Elastic Bamboo

If you would like to set up Elastic Bamboo for your Bamboo installation, please read [Getting started with Elastic Bamboo](#). This document guides you through the initial configuration of Elastic Bamboo and running your first Job build.

Elastic Bamboo Costs

This page provides high level guidelines to Elastic Bamboo costs. As usage patterns vary from user to user, these guidelines are only intended to provide a picture of how Elastic Bamboo operates, not to make definitive pricing statements.

 The [Bamboo pricing page](#) on the Atlassian website details the costs for Elastic Bamboo. This page is intended to complement that information.

Amazon EC2 Pricing Information

You can use Elastic Bamboo to run [remote agents](#) on [elastic instances](#) in the [Amazon Elastic Compute Cloud \(EC2\)](#). If you choose to do this, you will be charged by Amazon for your EC2 compute usage. These charges will be billed to the AWS account that you provide.

 Please note, if you do not have an AWS account, you must register for one on the [AWS registration page](#) before you can enable Elastic Bamboo.

Full details on Amazon EC2 pricing is available on the [Amazon EC2 pricing page](#). Please also note the following important information, which is relevant to EC2 usage by Elastic Bamboo:

- You are responsible for all EC2 compute usage costs incurred on your AWS account.**
- Elastic Bamboo creates "High-CPU Medium" Instances by default, however you can configure the [EC2 instance type](#). Read [Managing your elastic image configurations](#) for instructions on how to change your default instance type. Please note the different costs for different instance types.
- You are responsible for creating and shutting down elastic instances to run agents in EC2.
- You can track your EC2 usage in near real-time on the [AWS Account page](#).
- Your Elastic Bamboo compute usage will not be distinguishable from your non-Bamboo EC2 compute usage in your AWS billing.

General Notes about EC2 Usage and Costs

The following information is based on our usage of Elastic Bamboo at Atlassian. These points are intended to be guidelines to EC2 usage and costs only.

- The bulk of EC2 costs from using Elastic Bamboo is for the uptime of EC2 instances. We strongly recommend that you shut down your instances when not in use.
- The costs for storing and moving data in and out of the EC2 will vary. However these costs are minimal (e.g. storing image) compared to instance uptime costs. Using the Amazon Elastic Block Store (EBS) with Elastic Bamboo can significantly reduce the data transfer (and associated costs) in and out of the EC2. Read more about [configuring elastic instances to use EBS](#).
- The costs for using the Amazon Elastic Block Store (EBS) is minimal, relative to instance uptime costs.

Getting started with Elastic Bamboo

Elastic Bamboo allows you to use computing resources from the [Amazon Elastic Compute Cloud \(EC2\)](#) to run builds. Elastic Bamboo uses a [remote agent AMI \(Amazon Machine Image\)](#) to create [instances of remote agents](#) in the Amazon EC2.

On this page:

1. Read important documents
 2. Enable and configure Elastic Bamboo
 3. Start an Elastic Instance
 4. Run a plan build
 5. Shut down your Elastic instance
- Further information

1. Read important documents

If you are using Elastic Bamboo for the first time, we highly recommend that you start by reading the following important documents:

- **About Elastic Bamboo** — This high-level overview explains the key concepts behind the Elastic Bamboo feature.
- **Elastic Bamboo Security** — We *strongly recommend* that you read this document to understand the security implications of enabling Elastic Bamboo. This includes important information on securing your version control system (VCS) for use with Elastic Bamboo.
- **Elastic Bamboo Costs** — Elastic Bamboo sources resources from the Amazon Elastic Compute Cloud (EC2) which are charged separately to your Bamboo license fee. We recommend that you read this document to understand how you will be charged for using Elastic Bamboo.

2. Enable and configure Elastic Bamboo

Once you have understood the concepts, security implications and costs of Elastic Bamboo, you can enable and configure Elastic Bamboo for your Bamboo installation. You will also need to make your version control system (VCS) available to Amazon for Elastic Bamboo to work correctly.

2.1. Enabling Elastic Bamboo

To enable Elastic Bamboo:

1. Enable remote agent support in Bamboo — if you have disabled remote agent support, you must enable it before you can enable Elastic Bamboo. The [Disabling and enabling remote agents support](#) documentation also contains instructions on how to enable remote agent support.
2. In Bamboo, click **Administration** in the top menu bar.
3. Click **Configuration** in the left navigation panel (under 'Elastic Bamboo'). The 'Elastic Bamboo Configuration' screen will display.
4. Click **Enable**.

2.2. Configuring Elastic Bamboo

Before you can use Elastic Bamboo, you must configure it as detailed in the 'Configuring Elastic Bamboo' document. This is a simple three-step process:

1. [Provide your Amazon Web Services account details](#)
 2. [Configure your Elastic Bamboo global settings](#)
 3. [Configure your elastic instance settings](#)
- Read the [Configuring Elastic Bamboo](#) document.

2.3. Providing access to your VCS

You need to make your version control system available to Amazon to run job builds using Elastic Bamboo. This has security implications, particularly if your VCS is behind a firewall.

- Read the [Elastic Bamboo Security](#) document for further instructions, if you have not read it already.

3. Start an Elastic Instance

Now that you have enabled and configured Elastic Bamboo for your Bamboo installation, you can try building a plan with Elastic Bamboo. You can manually start an elastic instance using the Bamboo administration console. Starting an elastic instance will automatically start an elastic agent process on it.

- Read about [starting an elastic instance](#)

4. Run a plan build

To run a plan build on your elastic agent, you must [set up a plan](#) with its Default Job (plus any other optional jobs) all of whose [requirements](#) can meet your elastic agent's [capabilities](#). Elastic agents inherit the capabilities of the image they are started from. We recommend that you use the Bamboo default image to start with.

- Read about the [capabilities of the default image](#).

For the purposes of this guide, you should set up your plan so that its jobs' requirements can *only* be met by the elastic agent's capabilities. This will ensure that the jobs' builds run on your elastic agent. If you cannot set up your jobs' requirements to meet your elastic agent's capabilities, you can customise your elastic agent's capabilities to add a unique custom capability, e.g. 'elastic=true').

- [Read about configuring the capabilities of elastic agents.](#)

Job builds on elastic agents are run just like job builds on any other agent. You will see the progress of your build on your [dashboard](#) and can [view the build result](#) when it has completed.

 **Tip:** You can significantly reduce the costs and time taken to run a job build by [configuring Elastic Bamboo to use Amazon's Elastic Block Store \(EBS\)](#).

5. Shut down your Elastic instance

When your job builds successfully, shut down your elastic instance. As described in [Elastic Bamboo Costs](#), the bulk of your Elastic Bamboo costs are from instance uptime. We strongly recommend that you shut down your elastic instances when not in use.

- [Read about shutting down an elastic instance.](#)

Please note, that when you shut down an elastic instance, the agent process it is running is terminated. This means that elastic agents are not present on the 'Agents' page in Bamboo unless they are online. If you wish to view information about a terminated elastic agent, you can find the agent in the elastic agent usage history.

- [Read about viewing your elastic agent usage history.](#)

Congratulations! You have successfully set up and run a job build with Elastic Bamboo.

Further information

You may be interested in reading the following related topics below to help you manage and improve Elastic Bamboo's handling of job builds:

- [Managing your elastic images, Managing your elastic instances, Managing your elastic agents](#) — information hubs for managing Elastic Bamboo images, instances and agents.
- [Running job builds using Elastic Bamboo](#) — general questions about running builds using Elastic Bamboo.
- [Configuring elastic instances to use the EBS](#) — information on configuring Elastic Bamboo to use the Amazon Elastic Block Store (EBS) to improve job build times.

Configuring Elastic Bamboo

Elastic Bamboo allows you to use computing resources from the [Amazon Elastic Compute Cloud \(EC2\)](#) to run builds. Elastic Bamboo uses a [remote agent AMI \(Amazon Machine Image\)](#) to create [instances of remote agents](#) in the Amazon EC2.

Builds run on these 'elastic agents' in a similar way to how they run on local and remote agents.

 If you have disabled remote agent support, you **must enable it** before you can enable Elastic Bamboo. Refer to [Disabling and enabling remote agents support](#) for instructions on how to enable remote agent support.

To configure your Amazon Web Services (AWS) account details or settings for Elastic Bamboo:

1. Click **Administration** in the top navigation bar.
2. Click **Configuration** in the left navigation panel (under 'Elastic Bamboo').
3. Click **Edit**.
4. Configure settings as described in the sections below.
5. Click **Save** when finished.

On this page:

- [AWS account settings](#)
- [Global settings](#)
- [EC2 spot instances](#)
- [AWS settings](#)
- [Automatic elastic instance management](#)

Related pages:

- [Generating your AWS Private Key File and Certificate File](#)
- [Configuring elastic instances to use the EBS](#)
- [Managing Elastic Bamboo](#)
- [Disabling Elastic Bamboo](#)

AWS account settings

Before you use Elastic Bamboo for the first time in your Bamboo instance, you must enter your [Amazon Web Services \(AWS\)](#) account details into the Bamboo application. If you do not have an AWS account, you must register for one on the [AWS registration page](#) before you can

enable Elastic Bamboo.

Before you begin:

- Please note, Elastic Bamboo dynamically creates and runs remote agents in the [Amazon Elastic Compute Cloud \(EC2\)](#). Hence, if you choose to use Elastic Bamboo, *you will be charged by Amazon for your EC2 compute usage* (separately to your Bamboo license fee). These charges will be billed to the AWS account that you provide. Please read [Elastic Bamboo Costs](#) for more details.
- Please note, if you change your AWS account details, Bamboo will *stop all elastic agents* that are currently running.

To set your AWS account details:

1. Enter or update your **AWS Access Key ID** (you can get the "AWS Access Key ID" and "AWS Secret Access Key" if you go to your account: "My Account/Console" > Security Credentials).
2. To enter or update your AWS Secret Access Key, select the **Change AWS Secret Access Key?** checkbox, and enter or update **AWS Secret Access Key**.
3. Click **Save**.

Note that your AWS Access Key ID and AWS Secret Access Key are used together to identify you when accessing Amazon EC2 services. If you are unsure what your AWS Account ID and AWS Secret Access Key are, please refer to the Amazon documentation on [AWS access identifiers](#).

Global settings

Elastic Bamboo provides you with a number of global configuration options to help you optimise EC2 usage for your Bamboo job builds. These settings control how the Bamboo server operates and how it manages its elastic instances and agents.

Maximum Number of Elastic Instances	The number of elastic instances that can be running at any one time. You may wish to decrease this value if you are concerned about EC2 compute costs, and you have a large number of concurrent job builds that cannot be supported by your non-elastic agents.
Automatically terminate elastic instance when elastic agent process ends	Controls whether your elastic instances will automatically shut down after the elastic agent processes running on them terminate. • Shutdown Delay —controls how long an elastic instance will wait before shutting down, after its elastic agent process terminates.

EC2 spot instances

Elastic Bamboo provides support for [Amazon EC2 Spot Instances](#). Amazon spot instances allow you to bid on unused EC2 capacity and use it, as long as your bid exceeds the current "Spot price". You can configure Elastic Bamboo to bid for a spot instance of a particular type, and fall back to a regular instance after a set amount of time if no instances are available.

Enable support for spot instances	Select this checkbox to enable support for spot instances.
Fallback to a regular instance after	The time (in minutes) after which Elastic Bamboo will fall back to using a regular instance, if a spot instance has not become available.
Your current bid levels (per hour)	Fill out this table with your bids. The bids are categorised by EC2 instance type and operating system.

AWS settings

These settings allow you to specify your AWS configuration settings in Bamboo so that Bamboo can operate elastic instances through your AWS account. This section includes settings that are used to configure elastic instances to work with the Amazon [Elastic Block Store \(EBS\)](#).

Using EBS with your elastic instances can significantly reduce the amount of data transfer required to run a job build, compared with starting a clean elastic instance. To find out more about this feature and how to set it up in Elastic Bamboo, read [Configuring elastic instances to use the EBS](#).

Upload AWS account identifiers to new elastic instances	Select to upload the AWS Account Private Key File and Account Certificate File to all new elastic instances started. This is mandatory if you wish to use EBS to store job build information in a snapshot. However, you can also check this option if you are not using EBS (e.g. if you wish upload the AWS account identifiers in order to use Amazon's AWS command line tools).
Key files location	Choose how private key and certificate will be provided.

Account Private Key File	You must specify the location of this file to use the Amazon EBS with Elastic Bamboo. This file is generated by Amazon.
Account Certificate File	You must specify the location of this file to use the Amazon EBS with Elastic Bamboo. This file is generated by Amazon.

i If you haven't downloaded an AWS private key file or certificate file to your Bamboo server yet, please see [Generating your AWS Private Key File and Certificate File](#) for instructions.

Automatic elastic instance management

The Automatic Elastic Instance Management feature allows Bamboo to start and shut down elastic instances automatically (based on build queue demands), so that you do not have to [perform these action manually](#). This feature reduces Bamboo administration overhead and can help minimise your overall elastic instance usage costs.

If a job's requirements cannot be met by any available online agents, this feature will start any elastic instance whose elastic agent has the capabilities to execute the job, so that the job's build can be generated. Regardless of how an elastic instance was started, all elastic instances will be shut down based on the settings specified below.

Elastic Instance Management	Choose from the following elastic instance management presets. Each of these presets define values for the five criteria described in the 'Custom' user-defined options (below). (Bear in mind that both the 'Aggressive' and 'Passive' presets have trade-offs.) • Default — Balances build queue clearance rates with elastic instance usage costs. • Aggressive — Favours higher build queue clearance rates but with higher elastic instance usage costs. • Passive — Favours lower instance usage costs but with lower build queue clearance rates. • Custom — Choose your own settings, as described below. • Disabled — Disables Bamboo's automatic elastic instance management feature.
Idle Agent Shutdown Delay	Specify the number of minutes that an elastic agent must be idle before Bamboo shuts down the elastic instance running that agent. i Elastic instances running in the Amazon EC2 compute cloud are charged in hourly blocks from the time they are started. To maximise usage of elastic instances in a cost-effective manner, Bamboo only performs these checks just prior to the expiry of each hourly block.
Allowed non-Bamboo instances	The maximum number of elastic instances allowed on your AWS account that are not controlled by this Bamboo instance.
Maximum Number of Instances to Start at Once	The maximum number of elastic instances that Bamboo can start in one go. Bamboo only starts this maximum number of elastic instances on a 'per minute' basis.
Number of Builds in Queue Threshold	The total number of builds in a queue. When this and all other thresholds have been reached, new elastic instances will be started.
Number of Elastic Builds in Queue Threshold	The number of builds in the queue that can be executed on elastic instances. When this and all other thresholds have been reached, new elastic instances will be started.
Average Queue Time Threshold	The average number of minutes that job builds have been waiting in a queue. When this and all other thresholds have been reached, new elastic instances will be started.

Generating your AWS Private Key File and Certificate File

There are several security mechanisms associated with [Amazon Web Services \(AWS\)](#) and EC2:

- Your password, along with your email address, which you use to access the AWS portal, where you can (amongst other things) generate and deactivate all the credentials mentioned below.
- The AWS Access Key ID and Secret Access Key that are used by the Bamboo server to authenticate with AWS.
- A login key pair that you can use to log in to EC2 instances that have been started by Bamboo. The key pair is automatically generated, either the first time you use Elastic Bamboo, or if you delete the key pair. The key pair is listed as 'elasticbamboo' in your AWS console. Bamboo does not use this key pair.
- The AWS private key file and certificate file that are generated by Amazon and used together to allow Elastic Bamboo to securely access some of the AWS services, such as [EBS for elastic instances](#) and the Amazon command line tools. These are described [below](#).

On this page:

- [AWS private key file and certificate file](#)
 - [Generating the files](#)
 - [Downloading the files](#)
 - [Notes](#)

Related pages:

- [Configuring Elastic Bamboo](#)

AWS private key file and certificate file

The [Amazon Web Services \(AWS\)](#) private key file and certificate file are generated by Amazon and work together to allow Elastic Bamboo to securely access your AWS account. These are required to enable certain features, such as [EBS for elastic instances](#) and the Amazon command line tools.

- The **certificate file** contains the public key associated with your AWS account. This file is kept by Amazon, (not on your Bamboo server).
- The **private key file** contains the private key that is used to authenticate requests to AWS. This file must be stored on your Bamboo server, if you are using [EBS for elastic instances](#) or the Amazon command line tools.
- The public key and private key from these files together form an **X.509 certificate**.

Generating the files

The certificate file will be kept by Amazon (to inject into your elastic instances) and the private key file will be downloaded to your Bamboo server in your Bamboo Home directory. If you are setting up Elastic Bamboo on multiple Bamboo servers using the same AWS account, you can simply copy the private key file across from the original Bamboo server. You should not need to regenerate the private key file and certificate file unless your private key file is lost or corrupted.

If you do need to regenerate the private key file and certificate file, please follow the instructions in the [Amazon X.509 Certificates documentation](#). The Amazon documentation also contains instructions on using your own certificate, if you wish.

Downloading the files

Once the files are generated, you will be able to download them (see screenshot below). We recommend that you store the files in the [Home directory](#) of your Bamboo server.

Screenshot: Downloading the generated AWS private key file and certificate file

Create Success

You have successfully created a new X.509 Certificate for your AWS account.

Please download your Private Key file now. You must download your Private Key file (pk-), by clicking the link below before you navigate away from this page. AWS does not store your private key information. You will not be able to download the Private Key file at any other time. If you do not download the Private Key file now, you will have to create a new certificate and private key.

Your Private Key file
pk-Z6KBX7B75UDHHFJOWVZPL2K3LA7U4SZM.pem
[Download Private Key File](#)

IMPORTANT: You should store your Private Key file in a secure location. If you lose your Private Key file you will need to create a new certificate to use with your account. AWS does not store Private Key Information.

Your Private Key is secret, and should be known only by you. You should never include your Private Key information in a requests to AWS, except encrypted as a signature. You should also never e-mail your Private Key file to anyone. It is important to keep your Private Key confidential to protect your account.

Please download your certificate file. You can download your certificate file now using the link below, or at your convenience from the Access identifiers page.

Your X.509 Certificate file
cert-Z6KBX7B75UDHHFJOWVZPL2K3LA7U4SZM.pem
[Download X.509 Certificate](#)

[Return to Access Identifier Page](#)

Notes

- If you wish to use this security mechanism with **multiple Bamboo installations using the same AWS account** (e.g. you have [configured your elastic instances on each installation to use EBS](#)), you will need to copy the AWS private key file and certificate file to each Bamboo server.
- You can only download the AWS private key file **at the time it is generated**. If the private key file has already been generated for your AWS account, you will not be able to download it from AWS again (for security purposes). You will have to copy it from wherever it was previously downloaded to. Otherwise you will have to generate a new private key file and certificate file to go with it.
 *If you regenerate a new private key file and certificate file, any Bamboo servers using the old private key file and certificate file will no longer be able to access the Amazon EC2, as only one X.509 certificate can be associated with your AWS account.*
- You can download the AWS certificate file as many times as you want. This file does not need to be regenerated.

Configuring elastic instances to use the EBS

The Amazon [Elastic Block Store \(EBS\)](#) provides 'EBS volumes' which can attach to EC2 instances. EBS volumes (and the 'EBS snapshots' created from these volumes) provide persistent storage for your [elastic instances](#).

If you have relatively static resources required for building your Bamboo jobs (such as, source code checkouts and Maven repository artifacts), you can add these to an EBS volume. From this volume, you can create an EBS snapshot, which effectively records the 'state' of an EBS volume at a given point in time.

After setting up an EBS snapshot, you can then associate it with an [elastic image configuration](#). When this [elastic image](#) is started:

- its [elastic instance](#) will be started, along with the EBS volume (derived from the EBS snapshot associated with the elastic image) and
- this EBS volume will be attached to this elastic instance
 any build resources (added to the EBS volume prior to creating its snapshot) will be available to this elastic instance.

Why should I use the EBS with Elastic Bamboo?

Because an elastic instance is stateless, so also is the elastic agent that runs on it. Hence, every time an elastic instance is restarted from the same image:

- Any resources that its elastic agent must retrieve externally (for example, Maven repository artifacts), must be downloaded in their entirety.
- Full checkouts must be performed by elastic agents when new Jobs are built.

Therefore, you can use the EBS to store these external resources in an EBS volume and snapshot so that they do not have to be downloaded or source code checked out each time you start up an elastic instance from an image. If your jobs rely heavily on downloading such resources and/or you are not performing clean builds each time, the EBS may significantly **improve your build times**.

Additionally, the EBS provides an easy mechanism for **customising elastic agents**, rather than you having to [create a custom elastic image](#) from scratch (with your own elastic agent capabilities). For example, you could upload files and scripts to your EBS volume that would install resources such as PostgreSQL databases for your elastic agents, which will be available when the agent's elastic instance is started.

On this page:

- [Creating your first EBS snapshot](#)
- [Configuring an elastic image to use an EBS snapshot](#)
- [Updating your EBS snapshot](#)
- [Important EBS directories and files](#)

Related page:

- [Configuring Elastic Bamboo](#)
- [Populating your EBS volume](#)

Creating your first EBS snapshot

To create your first EBS snapshot:

1. Download Amazon Web Services (AWS) account identifiers to your Bamboo server — You will need to store the AWS private key file and certificate file on your Bamboo server to use Elastic Bamboo with EBS. If you haven't downloaded an AWS private key file or certificate file to your Bamboo server yet, please see [Generating your AWS Private Key File and Certificate File](#) for instructions.
2. Update your Bamboo configuration settings with the location of the AWS account identifier files you have downloaded. This will ensure that these files are uploaded to any new elastic instances started. See the Elastic Instance Settings section on the [Configuring Elastic Bamboo](#) for instructions (you will need to update the **Upload AWS account identifiers to new elastic instances (mandatory if EBS Snapshot ID specified)** checkbox and **Account Private Key File** and **Account Certificate File** fields described on this page).
3. Start a single elastic instance via Bamboo. See [Starting an elastic instance](#) for instructions.
4. Access your elastic instance via SSH (see [Accessing an elastic instance](#) for instructions).
5. As **root** user, follow the steps below to create an EBS volume and attach it to the elastic instance (*steps a & b*), upload content to the EBS volume (*step c & d*), and generate the snapshot (*step e & f*):



All the scripts described below are bundled with Bamboo.

- a. Run `createInitialVolume.sh <volume size>` — This script creates a EBS volume (where <volume size> is the size of the volume), attaches the volume and mounts it on the elastic instance. For example, `createInitialVolume.sh 100` will create a 100GB EBS volume and attach and mount it on the elastic instance.
- b. Run `rewarmEbsSnapshot.sh` — This script sets up the standard structure for Elastic Bamboo on the EBS volume. The directories and files for this standard volume structure are detailed in the [Important EBS Directories and Files](#) section below.
- c. (*optional*) Populate your EBS volume — Your EBS volume can now be populated with any files and scripts that you wish to make available to the elastic instances that use the EBS volume. For example, you may want to upload maven repository

data, source code, scripts and files to install databases on your elastic agents, etc. You must upload your files to the `/mnt/bamboo-ebs` folder or its subfolders, if you want them to be included in the snapshot. We recommend that you read [Populating your EBS volume](#) for guidelines on how to populate your EBS volume effectively.

i The EBS volume is attached to the elastic instance, so accessing your elastic instance via SSH will give you full access to the EBS volume (see [Important EBS Directories and Files](#) below).

- d. Ensure all uploaded content has the owner **bamboo:bamboo** — You can set the owner of a file by executing the following command: `chown -R bamboo:bamboo <filename>`
- e. Execute the `killall java` command — This command kills all processes on the instance, such as agent processes, so that the volume can be unmounted to be snapshotted.
- f. Run `generateSnapshot.sh` — This script unmounts and detaches the volume, before creating a snapshot based on the volume. The time taken to create the snapshot will vary depending on the amount of content that you have uploaded to the EBS volume. The **Snapshot ID** for the snapshot will be available in the logs for the elastic instance. See [Accessing an Elastic Instance](#) for instructions on how to access the logs for your elastic instance.

i The device can not unmount if any terminals are currently in the mounted volume.

6. Shut down your elastic instance. See [Shutting down an elastic instance](#) for instructions.

Configuring an elastic image to use an EBS snapshot

Once you have set up an EBS snapshot, the final step is to add the snapshot details to an elastic image configuration, so that any instances started from that image will have EBS volumes attached to them. You can associate different snapshots with different elastic image configurations.

To configure Elastic Bamboo to use an EBS snapshot:

1. Determine the Snapshot ID of the EBS snapshot you have just created. The Snapshot ID should be recorded in the logs of the elastic instance you created it on. You can also view your EBS snapshots in the [AWS Console](#) by clicking the **Snapshots** menu item.

Home > Your Account > AWS Management Console **BETA** [Show Navigation](#)

Overview **Amazon EC2** Welcome, Atlassian Elastic Bamboo | [Sign Out](#)

Navigation

- > **EC2 Dashboard**
- IMAGES & INSTANCES
 - > **Instances**
 - > **AMIs**
 - > **Bundle Tasks**
- ELASTIC BLOCK STORE
 - > **Volumes**
 - > **Snapshots**
- CONFIGURATION
 - > **Elastic IPs**
 - > **Key Pairs**
 - > **Security Groups**

EBS Snapshots

Create Snapshot Delete Create Volume Show/Hide Refresh Help

Viewing: All Snapshots 1 to 19 of 19 EBS

	Snapshot ID	Volume	Status	Started
☐	snap-1508ea7c	vol-8ef115e7	completed	2008-12-30 18:46 UTC+

1 Elastic Block Store Volume Snapshot selected

Snapshot ID: snap-1508ea7c

Volume: vol-8ef115e7	Status: completed	Progress: 100%
Started:	2008-12-30 18:46 UTC+1100	

2. Click **Administration** in the top navigation bar.
3. Click **Image Configurations** in the left navigation column (under 'Elastic Bamboo').
4. Click **Edit** in the 'Operations' column for the elastic image configuration that you would like to add your EBS snapshot to. The 'Edit Elastic Image Configuration - <imagename>' screen will display (see screenshot below).

Edit Elastic Image Configuration - Default

Edit the details of elastic image configuration, then click Save.

Elastic Image Configuration Details

Name *	Default
Description	Default Elastic Image Configuration shipped with Bamboo
AMI ID	ami-0ab64563 As this is the default configuration you can not edit the AMI ID.
	☒ Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances.
EBS Snapshot ID	snap-68204c00 Specify the EBS Snapshot ID to mount volumes from, when starting new elastic instances.
Instance Type	High-CPU Medium EC2 Instance Type
Availability Zone	Default (chosen by EC2) EC2 Availability Zone
Save Cancel	

5. Check **Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances**.
6. Enter the Snapshot ID of your EBS snapshot in the **EBS Snapshot ID** field.
7. Click **Save**. A new EBS volume will be created from the specified snapshot and attached to any new elastic instances started from that image.

Updating your EBS snapshot

If you are currently using EBS with Elastic Bamboo and want to update your snapshot, follow the instructions below. These are similar to the instructions for creating a new EBS snapshot.

To update your EBS snapshot:

1. Start a single elastic instance via Bamboo. See [Starting an elastic instance](#) for instructions.
2. (optional) Run a build on the elastic agent of the instance to populate the attached EBS volume. We recommend that you read [Populating your EBS volume](#) for guidelines on how to populate your EBS volume effectively.
3. Access your elastic instance via SSH (see [Accessing an elastic instance](#) for instructions) and do the following:
 - i** All the scripts described below are bundled with Bamboo.
 - a. (optional) Upload any additional content to the attached EBS volume via Secure Copy (SCP). You must upload your files to the `/mnt/bamboo-ebs` folder or its subfolders, if you want them to be included in the snapshot.
 - b. Execute `killall java` — This command kills all agent processes, so that nothing is using the mounted volume.
 - c. Execute `jps -vl` — This command displays a list of all java processes running on your instance. There should be no java processes running.
 - d. Run `generateSnapshot.sh` — This script unmounts and detaches the volume, before creating a snapshot based on the volume.
 - i** The device can not unmount if any terminals are currently in the mounted volume.
 - e. Check the elastic instance logs for the Snapshot ID of the snapshot you just created. See [Accessing an Elastic Instance](#) for instructions on how to access the logs for your elastic instance.
 - f. Update the new Snapshot ID in your Elastic Bamboo configuration, as described in [Configuring an Elastic Image to use an EBS snapshot](#) above.

Important EBS directories and files

By convention, Bamboo will attach an EBS device at `/dev/sdh`. This will be mounted at `/mnt/bamboo-ebs`. The contents of the standard structure are:

- `bin/customiseInstance.sh` - This script is run on startup of an elastic instance. We recommend that you do not customise this script, as it is overwritten when `rewarmEbsSnapshot.sh` is run.
- `bin/customise-extras.sh` - This script is run on startup of an elastic instance as the root (as opposed to being run as the Bamboo user). This script is safe to customise, as it will never be overwritten. You can customise this script to automate processes such as setting up your database, move files to custom locations on the instance, etc.
- `profile-extras.sh` - This script gets appended to the profile that is run under the Bamboo user (as opposed to being run as the root). It is useful for setting up environment variables.
- `bamboo-agent/bamboo-agent.cfg.xml` - This configuration file modifies the build working directory to point to build working directory on the EBS volume.
- `bamboo-agent/build-dir` - This is the build working directory.
- `maven/build.properties` - This properties file is copied to `/home/bamboo` on startup of an elastic instance. It points the Maven 1 default repository to `/mnt/bamboo-ebs/maven/.maven`

- `maven/.m2/settings.xml` - This configuration file is copied to `/home/bamboo/.m2` on startup of an elastic instance. It points the Maven 2 default repository to `/mnt/bamboo-ebs/maven/.m2/repository`.
- `tmp-extras` - The contents of this directory is copied to `/tmp` on startup of an elastic instance.

Populating your EBS volume

This page is intended to complement the instructions for [Configuring elastic instances to use the EBS](#). It lists different methods of for populating your EBS volume, depending on the data you wish to have available in your snapshot.

On this page:

- [Uploading Maven 2 repository data](#)
- [Uploading Ant repository data](#)
- [Setting up PostgreSQL for elastic agents](#)
- [Setting up Selenium on elastic agents](#)

Related pages:

- [Configuring elastic instances to use the EBS](#)

Uploading Maven 2 repository data

You can upload **Maven 2 repository data** to your EBS volume, so that it does not have to be downloaded every time an elastic agent (running on an instance which uses the EBS volume) is started.

To populate your EBS snapshot with your Maven repository data, we recommend that you upload it via SCP (see **step 5c** of the 'Creating your first EBS snapshot' section in [Configuring elastic instances to use the EBS](#)). In most cases, you will have a modified `settings.xml` file if you are using Maven 2. This means that you will need to upload this file and Maven repository data to your EBS volume, rather than populating your volume by running a build.

Uploading Ant repository data

You can upload **Ant repository data** to your EBS volume, so that it does not have to be downloaded every time an elastic agent (running on an instance which uses the EBS volume) is started.

To populate your EBS snapshot with your Ant repository data, we recommend that you run a build on an elastic agent with a blank EBS volume attached to the elastic instance (see **step 2** of the 'Updating your EBS snapshot' section in [Configuring elastic instances to use the EBS](#)). This is a faster and more reliable method of populating your volume, if you are using Ant.

Setting up PostgreSQL for elastic agents

You can upload scripts to your EBS volume so that the elastic agent started on any elastic instances which use this EBS volume, will have **PostgreSQL** automatically installed.

 *These elastic instances must be started from an elastic image which is associated with an EBS snapshot derived from this EBS volume.*

To set up the automatic installation of PostgreSQL on your EBS volume for elastic agents, you will need to create the following script:

setupPostgreSQL.sh

```
#!/bin/sh
yum install -y postgresql-server
service postgresql initdb
cat > /var/lib/pgsql/data/pg_hba.conf << EOF
local all all trust
host all all 127.0.0.1/32 trust
EOF
/etc/init.d/postgresql start
```

This script uses the package management tools provided by Fedora to install and configure PostgreSQL on the agent when its started.

1. Uses **yum** to install the PostgreSQL server packages. Details on the yum tool can be found in the [Fedora Software Management Guide](#).
2. Initialises the PostgreSQL server environment by creating the database directories and default config files.
3. Creates a new `pg_hba.conf` file which trusts all local connections and all connections coming from localhost.
4. Starts PostgreSQL.

You then need to update the `customise-extras.sh` file on your EBS volume (see [Important EBS Directories and Files](#)) to invoke this script.

Finally, you need to add a custom capability (e.g. `postgres=true`) to the elastic agents with PostgreSQL installed. You can do this by updating the elastic image configuration that the agents inherit their capabilities from. Read [Configuring elastic agent capabilities](#) for detailed instructions.

Setting up Selenium on elastic agents

You can upload scripts to your EBS volume so that the elastic agent started on any elastic instances which use the EBS volume, will be able to run **Selenium** tests.

 These elastic instances must be started from an elastic image which is associated with an EBS snapshot derived from this EBS volume.

To set up elastic agents to support Selenium test, you will need to create the following script:

setupSelenium.sh

```
#!/bin/sh

centosMajorVersion=5
centosVersion=${centosMajorVersion}

cat >/etc/yum.repos.d/centos-${centosVersion}.repo <<EOF
[centos-base]
name=CentOS - Base
mirrorlist=http://mirrorlist.centos.org/?release=${centosVersion}&arch=\$basearch&r
epo=os
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-${centosMajorVersion}
enabled=0

[centos-update]
name=CentOS - Updates
mirrorlist=http://mirrorlist.centos.org/?release=${centosVersion}&arch=\$basearch&r
epo=updates
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-${centosMajorVersion}
enabled=0

EOF

yum -y --enablerepo=centos-base install firefox

yum -y install xorg-x11-server-Xvfb xterm xorg-x11-server-utils
xorg-x11-fonts-IS08859-1-75dpi xorg-x11-fonts-Type1

/usr/bin/killall Xvfb

#Start virtual screen
Xvfb :100 -ac -screen 0 1024x768x24 &
```

This script uses the package management tools provided by Fedora to install Mozilla's Firefox and enough of X to get a VNC (Virtual Network Computing) server running.

1. Uses **yum** to install the following packages. Details on the yum tool can be found in the [Fedora Software Management Guide](#).
 - vnc-server — the vnc server used by the selenium test server.
 - xorg-x11-server-Xvfb xterm xorg-x11-server-utils twm xorg-x11-fonts- — these packages cover the xorg dependencies to get Firefox to run.
2. The script then copies some prepared VNC authentication files into the bamboo home directory and sets their permissions accordingly. These files are:
 - vncpasswd — this is the password file used by the VNC server (copied to /home/bamboo/.vnc/passwd)
 - vncxstartup — this is the script executed by the VNC server when a connection is made (copied to /home/bamboo/.vnc/xstartup)
3. The last step of this script is to manually install Firefox into /opt/firefox (we manually install Firefox because the package that would be installed by the Fedora 8 package management appears to be outdated).
 - The tar is extracted to the appropriate directory
 - The .bashrc file is customised to include the Firefox directory when searching for libraries. This is so Firefox will be able to find its libraries.

You then need to update the customise-extras.sh file on your EBS volume (see [Important EBS Directories and Files](#)) to invoke this script.

Finally, you need to add a custom capability (e.g. selenium=true) to the elastic agents with PostgreSQL installed. You can do this by updating the elastic image configuration that the agents inherit their capabilities from. Read [Configuring elastic agent capabilities](#) for detailed instructions.

Elastic IPs and Bamboo



If you're currently allocating Elastic IPs to your VPC instances on the AWS account you use to run Bamboo, make sure you read the Elastic IP management section below. This applies also if you don't use VPCs in Bamboo.

Elastic IPs (EIPs) are public IP addresses you can reserve for use on your AWS account and later assign to your instances. For a complete description of Elastic IPs, consult [the official Amazon documentation](#). Bamboo supports Elastic IPs in two ways:

- basic Elastic IP support - if the user assigns a new elastic IP to the agent instance, Bamboo is able to recognise that and will connect to the agent under the newly assigned address,
- Elastic IP management - when VPC instances are used, Bamboo will automatically assign Elastic IPs to VPC instances. As of Bamboo 4.3, EIPs are needed to communicate with elastic Agents deployed in VPC. We may remove that limitation in future Bamboo versions.

Elastic IP management and VPCs

By default, elastic instances deployed into VPCs get only private addresses and cannot connect to the Internet. This prevents Bamboo elastic agents from starting, because they require access to the AWS S3 service. In order to make this access possible, Bamboo will assign an Elastic IP to each instance started in a VPC. If there are any addresses already available, Bamboo will use one of them. Otherwise, a new address will be allocated and assigned to the instance.

Periodically, Bamboo will deallocate all currently unassigned Elastic IP addresses. **If you have any Elastic IPs you'd like to permanently allocate to your account, make sure you add them to the list of ignored IPs in your Elastic Bamboo configuration:**

Virtual Private Cloud Configuration

Reserved Elastic IPs

A comma-separated list of Elastic IPs Bamboo should ignore

AWS will not charge your account for Elastic IPs associated with running instances, so the fact that your VPC instances require an Elastic IP has not influence on your AWS usage fee.

Elastic IP management limitations

By default, your Amazon account supports only 5 elastic IPs. This limitation is important for users of agents deployed in VPC networks - unless you increase your EIP limit, you won't be able to run more than 5 Elastic Bamboo Agents. You can increase that limit by submitting [this form](#). For your convenience, here's a use case description you could use when submitting your request:

I am using Atlassian Bamboo Continuous Integration Server together with the Amazon Virtual Private Cloud. My Atlassian Bamboo server will automatically manage VPC instances. I need each of my VPC instances to have an externally-reachable IP address. The addresses will be allocated upon instance startup and automatically released after the instance shuts down. This scenario does not differ from a regular, non-VPC deployment, where each instance has its own IP address while running.

Managing Elastic Bamboo

The following pages and the related sub-pages contain information on managing your elastic image, instances and agents.

- [Managing your elastic images](#) — please see this page and the related sub-pages for detailed information about Elastic Bamboo images in Bamboo. This includes instructions on how to view and customise the capabilities of your Elastic Bamboo images.
- [Managing your elastic instances](#) — please see this page and the related sub-pages for detailed information about Elastic Bamboo instances in Bamboo. This includes instructions on how to view, start, stop and access an elastic instance.
- [Managing your elastic agents](#) — please see this page and the related sub-pages for detailed information about Elastic Bamboo remote agent instances in Bamboo. This includes instructions on how to view and disable an elastic instance.

Managing your elastic images

An *elastic image* is an Amazon Machine Image (AMI) that is stored in one of Amazon data centres for use with the [Elastic Bamboo feature](#). An elastic image is used to create [elastic instances](#), which in turn create [elastic agents](#). Conceptually, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system.

Each elastic image registered with the Amazon Web Services (AWS) has its own unique identifier, known as an **AMI ID**.

You can [associate multiple elastic images](#) with a Bamboo server. One [default shared image](#) is maintained by Atlassian in AWS, and is available to all Elastic Bamboo users.

You can also [create your own custom elastic images](#).

- To view an elastic image, including the image properties, capabilities and the jobs that an image can build, see [Viewing an elastic image](#).
- To associate an elastic image with your Bamboo installation, see [Managing your elastic image configurations](#).
- To customise the capabilities of an elastic image, see [Configuring elastic agent capabilities](#).
- To create your own custom elastic image, see [Creating a custom elastic image](#).

Viewing an elastic image

An [elastic image](#) is similar to an agent, so the 'Image' page closely resembles the '[Agent](#)' page. A number of functions available for agents are also available for images.

- **Viewing an elastic image's capabilities** — your image has capabilities, similar to how agents have capabilities. Read more about [viewing an agent's capabilities](#).
- **Viewing the jobs that an image can build** — you can also view the jobs that an image is capable of building (using the elastic agent created from the image), in a similar way to how you view the jobs that an agent is capable of building. Read more about [viewing the jobs that an agent can build](#) and [determining which agents can build which jobs](#).

Related pages:

- [Managing your elastic images](#)

To view an image:

1. Click **Administration** in the top navigation bar.
2. Click **Image Configurations** in the left navigation panel (under 'Elastic Bamboo').
3. Click the name, or **View**, for the image that you want to view.

Name	The name of the image.
AMI ID	The Amazon Machine Image identifier that uniquely identifies the image.
EBS Snapshot ID	The ID of the EBS Snapshot that you have associated with this image. See Configuring elastic instances to use the EBS and Managing your elastic image configurations for more information on how to use EBS with Elastic Bamboo.
Instance Type	The instance type of new instances started from this image. Read more about Amazon instance types .
Availability Zone Preference	New instances started from this image will be run in the Amazon availability zone named here.
Active Instances	The number of currently active instances that were started from this image.

Screenshot: Elastic Bamboo image configuration

Manage Elastic Image Configurations › Maven 2.1 Image

Elastic Image Configuration

Name	Maven 2.1 Image Contains Maven 2.1 and the necessary bits for Selenium 2
AMI ID	ami-0ab64563
EBS Snapshot ID	snap-68204c00
Instance Type	High-CPU Medium
Availability Zone Preference	Default (chosen by EC2)
Active Instances	13

[Start Instances](#) | [Disable](#) | [Edit](#)

Elastic Image Capabilities Executable Jobs

Elastic Image Capabilities

[Add Capability](#) | [Revert to Default Capabilities](#)

A capability is a feature of an agent. There are 3 types of capabilities: builders, JDKs and custom. You can use this page to view, add and delete capabilities associated with this Elastic Image Configuration. Any existing elastic instances will need to be restarted to pick up these changes.

The following capabilities exist on Elastic Agents running on an instance of this Image:

Custom

'Custom' capabilities are key-value pairs that define particular characteristics of an agent (e.g. 'operating.system=WindowsXP', 'fast.builds=true'). For an agent to be able to build a job, both the 'Key' and 'Value' must match the jobs's requirements.

Key	Value	Operations
bamboo.functionalTest	true	View Edit Delete

Builder

'Builder' capabilities define the builders which are available to your build plans.

Builder Label	Path	Operations
Ant (Ant)	/opt/apache-ant-1.7.1	View Edit Delete
Maven 2 (Maven 2.x)	/opt/maven-2.0	View Edit Delete
Maven 2.1 (Maven 2.x)	/opt/maven-2.1	View Edit Delete
Maven 2.2 (Maven 2.x)	/opt/maven-2.2	View Edit Delete

JDK

'JDK' capabilities define the JDKs which are available to your build plans.

JDK Label	Java Home	Operations
JDK	/opt/jdk-5	View Edit Delete
JDK 1.5	/opt/jdk-5	View Edit Delete
JDK 1.6	/opt/jdk-6	View Edit Delete

Mercurial

The path to the Mercurial executable (e.g. 'C:\Program Files (x86)\Mercurial\hg.exe' or '/usr/local/bin/hg')

Executable	Path	Operations
Mercurial	/usr/bin/hg	View Edit Delete

Add Capability

Capability Type

Key

Value

Managing your elastic image configurations

An *elastic image* is an Amazon Machine Image (AMI) that is stored in one of Amazon data centres for use with the [Elastic Bamboo feature](#). An elastic image is used to create [elastic instances](#), which in turn create [elastic agents](#). Conceptually, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system.

Each elastic image registered with the Amazon Web Services (AWS) has its own unique identifier, known as an **AMI ID**.

You can [associate multiple elastic images](#) with a Bamboo server. One [default shared image](#) is maintained by Atlassian in AWS, and is available to all Elastic Bamboo users.

You can also [create your own custom elastic images](#).

On this page:

- [Associating custom elastic images with Bamboo](#)
- [Creating elastic images with custom agent capabilities](#)

Related pages:

- [Managing your elastic images](#)

Associating custom elastic images with Bamboo

Associating a [custom elastic image](#) with your Bamboo installation allows you to start elastic instances with capabilities that are different from those inherited from the [default image](#). For example, you may wish to associate a Ubuntu operating system-based elastic image with your Bamboo installation, so that you can run Ubuntu-related tests on the instances started from that image.

Once you have associated a custom elastic image with Bamboo, the settings for your elastic image are stored as an elastic image configuration.

To associate a custom image with Bamboo:

1. Click **Administration** in the top navigation bar.
2. Click **Image Configurations** in the left navigation panel (under 'Elastic Bamboo').
3. Enter the details of your custom elastic image in the panel under 'Create Elastic Image Configuration':

Name	The name of your custom elastic image. If you created your own custom image, you should have named it in step 6 of the Creating a custom elastic image instructions. You can also view the image name via the AWS console .
Description	A description for your image, which is used in Bamboo only.
AMI ID	The AMI ID obtained as an output from step 6 of the Creating a custom elastic image instructions. You can also view the AMI IDs of elastic images via the AWS console .
Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances	Select this option if you want the elastic instances started from this image to use the EBS. Read more about Configuring elastic instances to use the EBS . EBS Snapshot ID — Specify the EBS Snapshot ID of the EBS volume that you wish to attach to new instances.
Instance Type	The instance type for new instances started from this image. Amazon offers a number of instance types that provide different computing capacity. Read more about Amazon EC2 instance types .
Virtual Private Cloud Subnet	The Subnet of the Virtual Private Cloud where your Elastic Bamboo agent will start up. For more about VPC, see the Amazon VPC FAQ .

Availability Zone	The availability zone used to start your new instances from this image in (e.g. if you wish to use Elastic Bamboo with reserved instances). We recommend that you select "Default (chosen by EC2)" to allow Amazon to select the best zone for your instance. Read more about Amazon EC2 availability zones .
Product	The EC2 product name.

Screenshot: Manage your Elastic Image Configurations

Elastic Image Configuration Details

Name*

Description

AMI ID*

☐ Automatically attach an [Amazon Elastic Block Store \(EBS\) volume](#) to new elastic instances.

Instance Type EC2 Instance Type

Virtual Private Cloud Subnet

Availability Zone EC2 Availability Zone

Product EC2 Product Name

[Cancel](#)

Creating elastic images with custom agent capabilities

You can customise the agent capabilities of an elastic image that is already associated with Bamboo. The initial process is similar to that of associating a custom elastic image with Bamboo ([above](#)). Here, however, you use the AMI ID of an image already associated with Bamboo — most commonly the [default image](#).

To create an elastic image with customised agent capabilities:

1. Click **Administration** in the top navigation bar.
2. Click **Image Configurations** in the left navigation panel (under 'Elastic Bamboo').
3. Click the name, or **View**, for the image that you want to view.
4. Enter the details of your custom elastic image in the panel (under 'Create Elastic Image Configuration'). See the section above for details.
5. You now have a new elastic image configuration based on an existing elastic image. Follow the procedure on [Configuring elastic agent capabilities](#) to customise this elastic image's agent capabilities.

Creating a custom elastic image

An *elastic image* is an Amazon Machine Image (AMI) that is stored in one of Amazon data centres for use with the [Elastic Bamboo feature](#). An elastic image is used to create [elastic instances](#), which in turn create [elastic agents](#). Conceptually, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system.

Each elastic image registered with the Amazon Web Services (AWS) has its own unique identifier, known as an **AMI ID**.

You can [associate multiple elastic images](#) with a Bamboo server. One [default shared image](#) is maintained by Atlassian in AWS, and is available to all Elastic Bamboo users.

At a high level, the process for creating a custom elastic image consists of taking one of the existing Amazon Machine Images (AMIs) available on Amazon EC2, starting an instance of the AMI, customising the instance and then creating an image from the customised instance. This image can then be used as an elastic image in your Bamboo installation.

Before you begin:

- This is *not a trivial procedure*. If you're using Linux/UNIX, instead of creating a custom elastic image, you may want to consider customising an existing Bamboo image by using Amazon's Elastic Block Store (EBS), as described in [Configuring elastic instances to use the EBS](#). This is a much simpler option. If you are having problems, please don't hesitate to [contact us](#) for further help.
- Please note, Atlassian *does not support custom elastic images*. Consider customising the elastic agents started from your existing image instead.
- A number of the EC2 commands in the steps below can be completed using the [AWS console](#) rather than command line tools (e.g. registering an image). You should use the method you're feel most comfortable with.

On this page:

1. Requirements
2. Selecting an existing AMI
3. Starting an instance
4. Accessing your instance
5. Customising your instance
6. Creating an image of your customised instance
7. Next steps
8. Need more help?

1. Requirements

First ensure that you have set up the following:

- **Amazon Web Services (AWS) account with EC2** — if you are already using Elastic Bamboo, you should already have an AWS account with EC2 set up. If not, please read [Getting started with Elastic Bamboo](#).
- **Amazon EC2 API Tools** — you must install the [EC2 API tools](#) on your local machine, otherwise you will not be able to start and access your AMI instance. Note: you need Java Runtime Environment to run these tools. You can install the EC2 API tools by executing the following commands:

```
wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip
unzip ec2-api-tools.zip
```

- **Environment Variables**— you must set up the following environment variables on your local machine before creating a custom elastic image:
 - EC2_HOME — set this to the path to the installed EC2 API Tools
 - EC2_CERT — set this to the path to the certificate assigned to EC2 account
 - EC2_PRIVATE_KEY — set this to the path to the private key assigned to your AWS account
- **Registered Key Pair**— you need a registered EC2 key pair, which consists of a private key file and certificate file, to use the EC2 API tools with your AMI instance. If you have previously generated and registered an EC2 key pair (e.g. to use the EC2 API tools), you can re-use it. If you need to generate a new key pair, you can use the following command to do so:

```
ec2-add-keypair <key_pair_name>
```

The content of the private key will be displayed in the command-line output on your console. Save this content in a file , starting with the line:

```
--BEGIN RSA PRIVATE KEY--
```

and ending with the line:

```
--END RSA PRIVATE KEY--
```

This private key file will be used to access your AMI instance. Set up the appropriate permissions on the private key file by executing the following command:

```
chmod 600 <private_key_file>
```

2. Selecting an existing AMI

We strongly recommend that you select an existing Linux/UNIX AMI to customise, rather than starting with a blank AMI. When choosing an AMI, decide whether you want to launch **32-bit** or **64-bit** instances from your custom elastic image and select an existing AMI matching your choice.

We recommend the following existing Linux/UNIX AMIs for customisation (in order of preference):

Source	Description	AMI list
Atlassian	One of the stock images provided by Atlassian. It is an Amazon image, for either Linux or Windows, updated and prepared for Bamboo, i.e. you will not have to install any Bamboo prerequisites.	Available on your Bamboo instance under Administration/Image Configurations

Amazon	CentOS-based image provided by Amazon. It does not have any Bamboo prerequisites installed. Typically, you will be better off using the Atlassian AMI.	Amazon's site
Canonical(Ubuntu)	An official Ubuntu image provided by Canonical (the company behind the Ubuntu Linux project). It does not have any Bamboo prerequisites installed.	Canonical's site

Atlassian's AMIs (and hence, their IDs) may change with each release of Bamboo, including both major and minor releases. To quickly access Atlassian's AMI IDs for your a currently-running version of Bamboo, open that Bamboo site in a web browser and access its 'Image Configurations' page (see [Managing your Elastic Image Configurations](#) for more information). The AMI IDs of Atlassian's AMIs are labelled with "(stock image)".

▼ If you want to find out the AMI IDs for a version of Bamboo you don't have running or you're starting an image from scratch and you need the image baseline, click here...

1. Open the following URL: <https://maven.atlassian.com/content/repositories/atlassian-public/com/atlassian/bamboo/atlassian-bamboo/> in a web browser.
2. On the resulting directory page, click the link that represents the version of Bamboo you are currently running. For example, if you are running Bamboo 3.4.4, click on the **3.4.4** link. Another directory page opens, listing a .pom and some additional checksum files.
- ⚠ Do not click on a version number link that contains 'mX', 'rcX' or 'betaX' (where 'X' is a number), since these relate to publicly available developmental releases of Bamboo.
3. Open the `atlassian-bamboo-x.x.x.pom` file (where `x.x.x` is your version of Bamboo). The image version/baseline is stored in `elastic-image.version` tag. For example, for version 3.4.4, the baseline was 1.7 .
4. Open the following URL: <https://maven.atlassian.com/content/repositories/atlassian-public/com/atlassian/bamboo/atlassian-bamboo-elastic-image/> in a web browser.
5. Click on the image baseline version you found in the `elastic-image.version` tag.
6. On the resulting directory page, the file with `ami` extension contains all stock image AMI ids.

3. Starting an instance

After you have selected an existing AMI to customise, the next step is to start an instance of the AMI.

3.1 Starting an instance of Atlassian's default AMI

If you chose to customise Atlassian's default AMI, you will have to start the instance from the admin section of Bamboo. See [Starting an elastic instance](#).

Note that Atlassian's default AMI cannot be started using the command line `ec2` tools. This is because, on start up, the Bamboo agent on Atlassian's AMI checks to see if it was started from a Bamboo server, and immediately shuts itself down if it was not.

Once started, see [Accessing an elastic instance](#) for details on how to access the running instance.

3.2 Starting an instance from the command line

Use the `ec2-run-instances` command to start your instance, as follows:

```
ec2-run-instances <image_name> -k <key_pair_name>
```

where `<image_name>` is the name of the AMI selected in the previous step and `<key_pair_name>` is the name of the registered key pair generated in '[1. Requirements](#)'. The public certificate of this key will be injected into your instance.

For example, if you wanted to start an instance of image `ami-e55bbd8c` using key pair `my-keypair`, you would run the following command:

```
ec2-run-instances ami-e55bbd8c -k my-keypair
```

This command would produce the following command-line output:

```
INSTANCE          i-25b86743    ami-e55bbd8c    running    my-keypair
```

`i-25b86743` is the name of the new instance in the above example. You should note down the name of your new instance, as you will need that to [access your instance](#) in the next step.



Don't forget to shut down unused instances

Please note that once you start an instance, you will be billed by Amazon for instance uptime. If you decide to abandon the setup of a custom elastic image after this step, please ensure that you [shut down your instance via the AWS console](#).

3.3 Starting an instance from Bamboo

You can also start a fresh, uncustomised image from Bamboo and begin customisation. The drawback of this approach is that you have only 40 minutes before Bamboo shuts down your instance. The advantage is that you can customise the agent in a single step (as opposed to having to customise/create image/start from Bamboo/save image again).

4. Accessing your instance



If you started the instance from Bamboo, see [Accessing an elastic instance](#) for details on how to access the running instance.

Once your instance is running, you will need to obtain the address of the instance so you can access it. To do this, use the following command:

```
ec2-describe-instances <instance_name>
```

For example, if you wanted to find the address of instance `i-25b86743`, you would enter:

```
ec2-describe-instances i-25b86743
```

This command would produce the following command-line output similar to this:

```
RESERVATION r-790f7210 121852097033 default
INSTANCE i-25b86743 ami-e55bbd8c
ec2-174-129-94-241.compute-1.amazonaws.com
domU-12-31-39-04-38-87.compute-1.internal running elasticbamboo 0
m1.small
2009-06-24T12:36:20+0000 us-east-1c aki-a71cf9ce ari-a51cf9cc
monitoring-disabled
```

The address of the instance in the above example is `ec2-174-129-94-241.compute-1.amazonaws.com`

You can then use this address to access the instance via SSH. See [Accessing an elastic instance](#) for instructions. If you are using the example command text from that document, you will need to adjust it as follows:

- replace `/opt/bamboo/home/xml-data/configuration/elasticbamboo.pk` in the example command text with the private key file you generated in '[1. Requirements](#)'.
- replace `ec2-68-111-185-197.compute-1.amazonaws.com` in the example command text with the address of your instance.

5. Customising your instance

Now that you have a running instance, customisation steps heavily depend on the operating system you're using. We've prepared separate pages with [Linux-specific instructions](#) and [Windows-specific instructions](#).

6. Creating an image of your customised instance

The process of creating a new image varies depending whether you based your image on an instance-store or EBS-root image. You can check your image type via AWS console or using `ec2-describe-images`.

Creating an image from EBS-root instances

See here for instructions: [Amazon Tutorial](#)

Creating an image from instance-store (S3) instances

The final step is to create an image from your customised instance. To do this, you will require the following information:

- Amazon Account Number
- Access Key ID
- Secret Access Key
- Amazon S3 bucket name that will be used to store image (if you don't have access to Amazon S3, you can sign up on [this page](#).)

1. Transfer Amazon private key file and certificate to your instance

Transfer the key files to your instance by running these commands on your **local machine**:

```
scp -i <private_key_file> $EC2_PRIVATE_KEY  
root@<instance_address>:/mnt  
scp -i <private_key_file> $EC2_CERT root@<instance_address>:/mnt
```

where <private_key_file> is the private key file from your local machine created in step 'Registered Key Pair' of [1. Requirements](#) and the <instance_address> is the address of your instance from '[4. Accessing your Instance](#)'.

2. Set up EC2_HOME and JAVA_HOME environment variables

Set up these environment variables by running the following commands on your instance:

```
export EC2_HOME=<location of your EC2 tools installation>  
export EC2_PRIVATE_KEY=/mnt/<ec2_private_key_file>  
export EC2_CERT=/mnt/<ec2_certificate_file>  
export JAVA_HOME=<path to JRE used to start the agent>
```

3. You can create an image of your customised instance by using the ec2-bundle-vol command, as follows:

```
ec2-bundle-vol -c $EC2_CERT -k $EC2_PRIVATE_KEY -u  
<amazon_account_number> -p <elastic_image_name> --batch --debug
```

where <elastic_image_name> is the name that you want to assign to your custom image (e.g. 'CustomImage1').

4. Once the image is created, you need to upload it to Amazon S3 by running the command below:

```
ec2-upload-bundle -b <s3_bucket_name> -m  
/tmp/<elastic_image_name>.manifest.xml -a <access_key_id> -s  
<secret_access_key> --retry --debug
```

where <s3_bucket_name>, <access_key_id> and <secret_access_key> are the Amazon S3 bucket name, Access Key ID and Secret Access Key described previously, and <elastic_image_name> is the name that you want to assign to your custom image (e.g. 'CustomImage1').

You will then need to register your image with Amazon EC2 by using the ec2-register command:

```
ec2-register <s3_bucket_name>/<elastic_image_name>.manifest.xml
```

where <s3_bucket_name> is the Amazon S3 bucket name described previously and <elastic_image_name> is the name that you want to assign to your custom image (e.g. 'CustomImage1'). The output of this command will show the AMI ID of your custom image.

7. Next steps

Now that you have created a custom elastic image, there are **two more steps** that you will need to complete before you can use it.

First, you will need to **associate your custom elastic image with your Bamboo installation** by creating an Elastic Image Configuration. Please note the AMI ID of your new custom image and read [Managing your Elastic Image Configurations](#) for further instructions.

Secondly, you will need to **configure the capabilities of the elastic agents** that will run on instances started from your image. This is done by adding the appropriate builder, JDK, Perforce and custom capabilities to your elastic image configuration, so that it reflects what your custom elastic image actually can do. For example, if you have created a custom elastic image with JDK 1.6 and Maven 2 installed, you will need to add capabilities for JDK 1.6 and Maven 2 to the elastic image configuration. Read [Configuring Elastic Agent Capabilities](#) for further instructions.

8. Need more help?

If you need **more help**, there are a number of resources that you can take advantage of:

- [AWS Support Center](#) — if you are having problems with any of your Amazon services, not specifically related to Bamboo, you can obtain basic support from the AWS Support Center. Note, you will need to sign up for Premium Support to get access to web/phone support.
- [AWS Resource Center](#) — the AWS Resource Center has links to online documentation, code samples and tools for AWS services.
- [Bamboo Developer Forums](#) — please feel free to discuss any useful tips or issues regarding this process in the Bamboo Developer Forums.

Creating a custom elastic image - Linux

5. Customising your instance

Customising your instance is the most complicated part of creating a custom elastic image. You need to [install the packages](#) that are prerequisites for Bamboo onto your instance (if you didn't choose the Elastic Bamboo [stock images](#) as your base AMI), [add your customisations](#), [deploy Bamboo](#) onto your instance and [set up an EC2 environment](#) on your instance.

5.1 Installing Bamboo prerequisite packages

If you selected Atlassian's AMI as your base AMI in '[2. Selecting an Existing AMI](#)', you can skip this step and go to '[5.2 Adding Customisations](#)' as this image has been pre-configured for Bamboo. If you have selected a different AMI, you will need to install the following packages onto your instance using the commands shown below:

Amazon EC2 API tools

```
wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip
unzip ec2-api-tools.zip
mv ec2-api-tools-* /opt/ec2-api-tools
```

Note: if your distribution already contains ec2-api-tools package, you can install it instead.

Java JRE

You need to install JRE (or JDK) on your instance to be able to run the agent. The preferred way of doing this is to install a package that came with your distribution. For a list of supported JREs, see [supported platforms](#).

5.2 Adding user customisations to your instance

Adding your own customisations is quite a simple process, once you have made it this far.

To add user customisations to your instance:

1. Log into your elastic instance (as previously described in '[4. Accessing your Instance](#)').
2. Once you have logged into your elastic instance, you can treat it as a standalone machine and install anything you want. For example, if you want to install Tomcat on an Ubuntu instance you would run 'sudo apt-get install tomcat6', configure it, ensure that your startup scripts are in place, etc, just as you would when installing Tomcat on a standalone machine.
 Please note however, you cannot customise the operating system of a running instance. If you want to create an instance with a customised operating system (e.g. Ubuntu), you will need to select an AMI with that operating system installed (as previously described in '[2. Selecting an Existing AMI](#)').
3. Everything that you install will be saved in snapshot image created at the end of these instructions (see '[6. Creating an Image of your Customised Instance](#)'). Any instances started from this image will have all of your user customisations automatically installed.

5.3 Deploying Bamboo onto your instance

Once you have installed the Bamboo pre-requisites on you instance and added your customisations, you can deploy Bamboo elastic bootstrap files onto your instance.

5.3.1 Creating Bamboo user

First, you need to create a 'bamboo' user on your instance by running the following command:

```
useradd -m bamboo
```

5.3.2 Downloading agent installer to the instance

Then, install Bamboo Agent binaries as described below. In this case we're using image version 2.2, you should use the latest version available at <https://maven.atlassian.com/content/repositories/atlassian-public/com/atlassian/bamboo/atlassian-bamboo-elastic-image/>.

```
imageVer=2.2
wget
https://maven.atlassian.com/content/repositories/atlassian-public/com/at
lassian/bamboo/atlassian-bamboo-elastic-image/${imageVer}/atlassian-bamb
oo-elastic-image-${imageVer}.zip
sudo mkdir -p /opt/bamboo-elastic-agent
sudo unzip -o atlassian-bamboo-elastic-image-${imageVer}.zip -d
/opt/bamboo-elastic-agent
sudo chown -R bamboo /opt/bamboo-elastic-agent
sudo chmod -R u+r+w /opt/bamboo-elastic-agent
```

5.4 Instance configuration

At this stage, you should have a customised instance with Bamboo deployed onto it. The last step in creating a customised instance is to set up an EC2 environment on your instance. Carry out the following steps to set this up:

1. Run the following command on your instance to set permissions on the bamboo user directory:

```
chown -R bamboo:bamboo /home/bamboo/
```

2. Configure path variables

Create a file **profile.sh** in your instance's `/mnt` directory. This file contains the default Elastic Bamboo path configuration settings, as seen below:

```
export JAVA_HOME=<path to JRE used to start the agent>
export EC2_HOME=<location of your EC2 tools installation>
export EC2_PRIVATE_KEY=/root/pk.pem
export EC2_CERT=/root/cert.pem
export
PATH=/opt/bamboo-elastic-agent/bin:$EC2_HOME/bin:$JAVA_HOME/bin:$M2
_HOME/bin:$MAVEN_HOME/bin:$ANT_HOME/bin:$PATH
```

If all of the tools on this page were installed in recommended locations, no changes are required. Otherwise, you can update the file as required.

Once **profile.sh** is correctly customised for your instance, you need to copy it to the `/etc/profile.d` directory by running the following command on your instance in the `/mnt` directory:

```
mv profile.sh /etc/profile.d/bamboo.sh
```

3. Configure automatic startup of the Bamboo agent

You will need to configure your instance to start the Bamboo agent automatically when the instance is started. You can do this by sourcing Bamboo `rc.local` file in `/etc/rc.local` file:

```
. /opt/bamboo-elastic-agent/etc/rc.local
```

Make sure you add this line before the "exit 0" line in your `/etc/rc.local`.

4. Final settings and cleanup

Finally, create a Bamboo welcome screen and clean up keys on your instance by running the following command:

```
cp /opt/bamboo-elastic-agent/etc/motd /etc/motd
echo bamboo-<x.x.x> >> /etc/motd
rm -f /root/firstlogin /etc/ssh/ssh_host_da_key
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/ssh_host_key /etc/ssh/ssh_host_key.pub
/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key.pub /root/.ssh/authorized_keys
touch /root/firstrun
```

where <x.x.x> is the Bamboo version you are running (e.g. 4.1.2).

5. **Now, follow the instructions from section "Creating an image of your Customised Instance" to create an AMI.**
6. **Start the image from Bamboo. The agent should come up and download all necessary data to the EC2 instance.**
7. Run `/opt/bamboo-elastic-agent/bin/prepareInstanceForSaving.sh`.
8. **Now, follow the instructions from section "Creating an image of your Customised Instance" to create an AMI.** That's it, the newly created AMI contains everything you need to start Bamboo Agents.

Note: if you started your instance from Bamboo right at the start, instead of steps 5 & 6, you can just run:

```
su -c /opt/bamboo-elastic-agent/bin/bamboo-elastic-agent - bamboo
```

Creating a custom elastic image - Windows

To perform the tasks listed below, log in to your instance with an Administrator account using Remote Desktop Client.

Setting up the user account

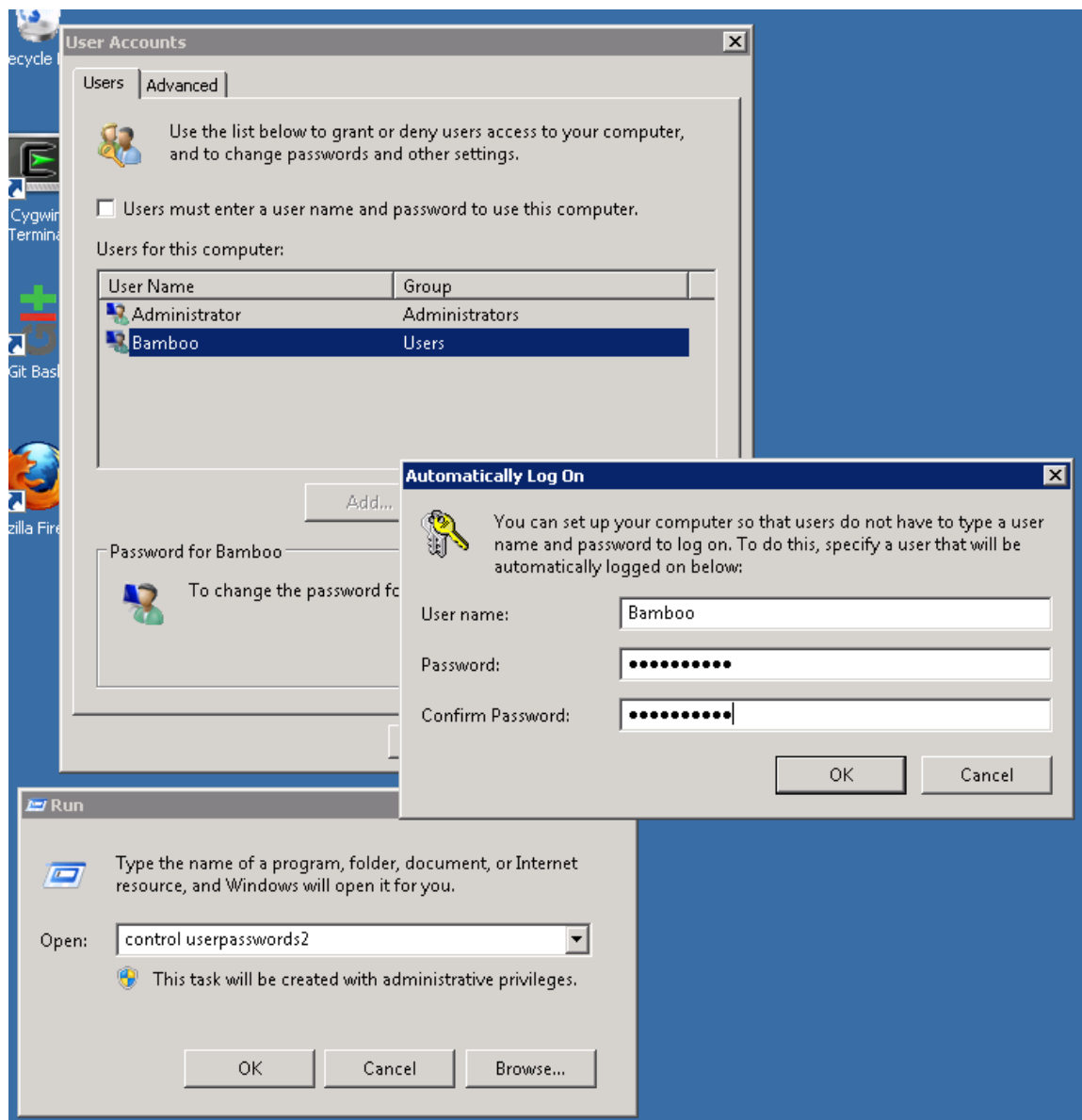
Create the user account that will be used by the Bamboo agent. The account name is up to you, I will use Bamboo in the examples below. All builds running on your machine will use this account. It can be a regular user (i.e. it does not need to be a Power User or Administrator, unless your builds require it). Set up a password for that user. The default user on a Windows image has a user name of Bamboo with a password of Atlassian1.



Important: by default, a newly created user should be denied remote login rights (which is as we want it to be). To be on the safe side, please make sure that you indeed **cannot** log in using that user's credentials (unless you change the credentials to non-default ones).

✓ If your builds are not headless (i.e. they show/manipulate windows, like Selenium does), [click here for additional instructions](#)

You'll need to set up autologin for your Bamboo account (don't worry, this will not let remote users in). To do this, run `control userpasswords2` and uncheck **User must enter a user name and password to enter this computer**:

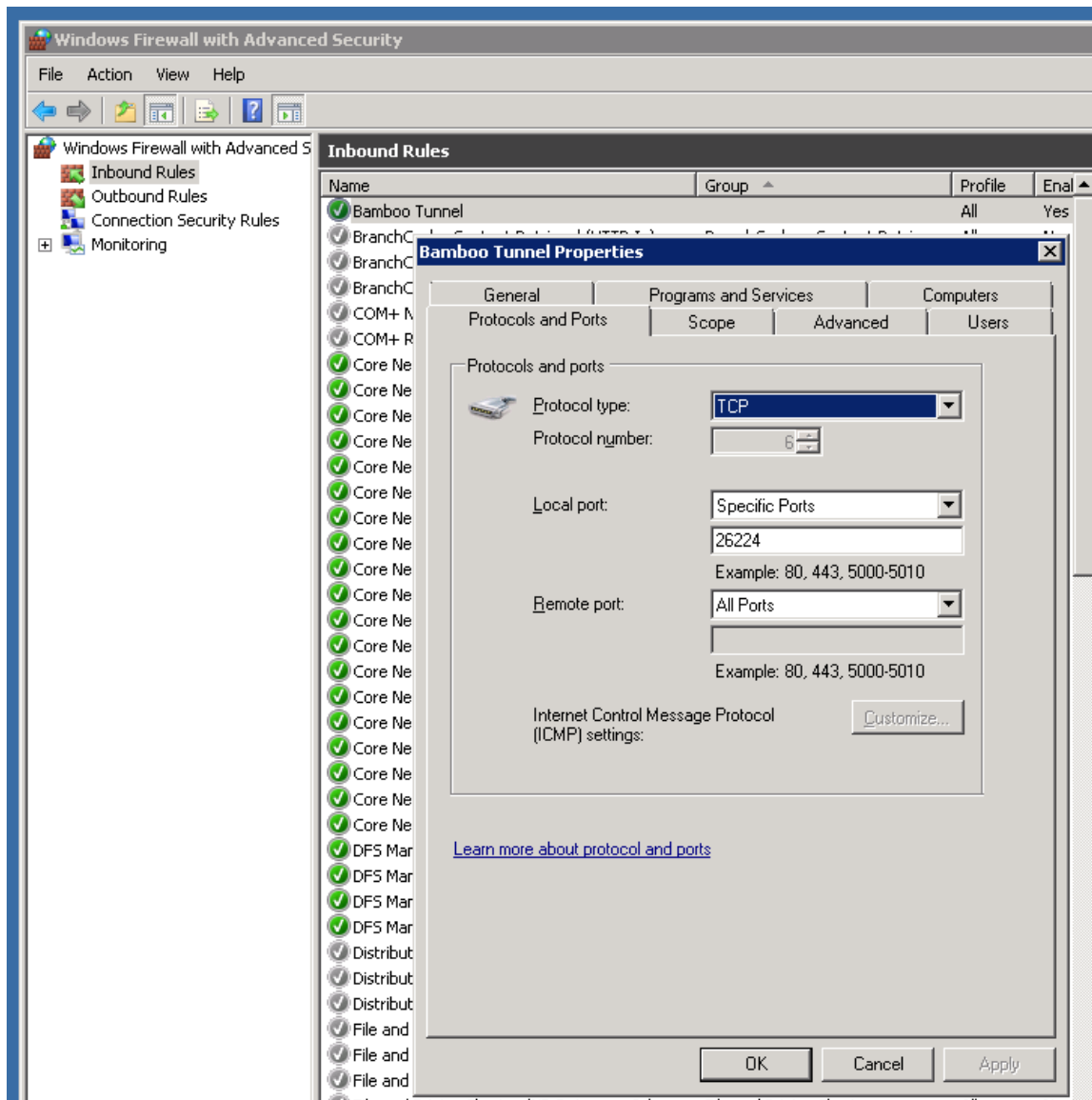


Setting up the firewall

Reconfigure the Windows firewall to accept TCP connections on port 26224. No other inbound connections are necessary for Bamboo.

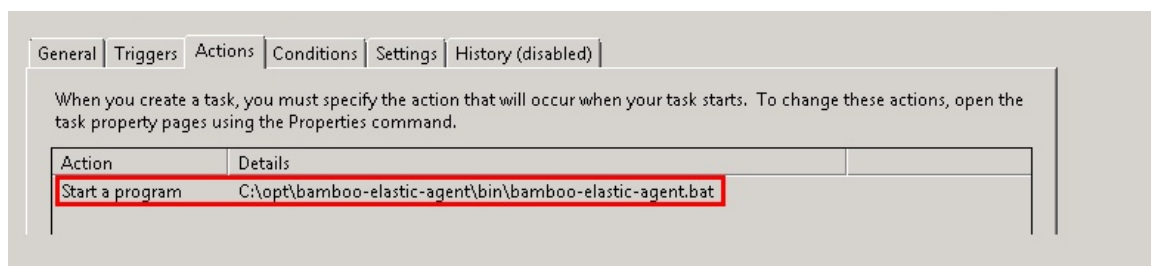


You don't need to worry about changing the EC2 security group setting for this port. Bamboo will set it up automatically:



Installing the required software

1. Install a [supported](#) Oracle Java version.
2. Download the latest version of agent installer zip from this [this location](#) (at the time this guide was written, the latest version was [this](#)). Unpack it to a desired location, we suggest using C:\opt\bamboo-elastic-agent to match stock images distributed with Bamboo.
3. A batch file should launch with your Windows instance startup. In order to do this, use the Windows Task Scheduler (**Start > Administrative Tools > Task Scheduler**), and set up a new Action task of "Start a program" with the <PATH TO YOUR BATCH FILE> as the Details:



Remember to select **Run whether user is logged on or not** in the "General" tab:

General | Triggers | Actions | Conditions | Settings | History (disabled)

Name: Start Bamboo Agent

Location: \

Author: IP-0AC24A2F\Administrator

Description:

Security options

When running the task, use the following user account:

☐ Run only when user is logged on

☒ Run whether user is logged on or not

☐ Do not store password. The task will only have access to local resources

☐ Run with highest privileges

☐ Hidden

Configure for: Windows Vista™, Windows Server™ 2008

And appropriately define the Trigger task so that the agent starts up only after the network connection is up and running:

General | Triggers | Actions | Conditions | Settings | History (disabled)

When you create a task, you can specify the conditions that will trigger the task.

Trigger	Details	Status
On an event	On event - Log: Microsoft-Windows-NetworkProfile/Operational	Enabled

Edit Trigger

Begin the task: On an event

Settings

☒ Basic

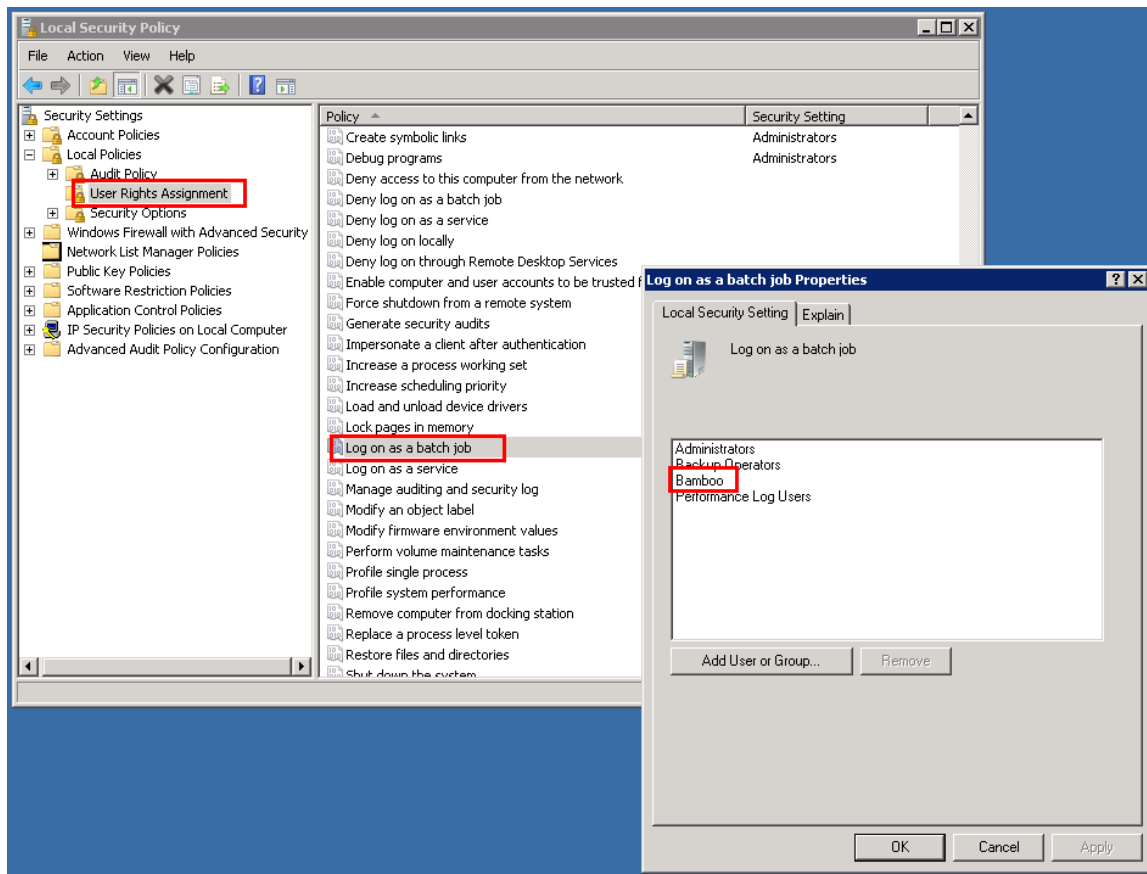
☐ Custom

Log: Microsoft-Windows-NetworkProfile/Operational

Source: NetworkProfile

Event ID: 10000

The task manager will warn you that the account needs to be able to log in as a batch job. Make sure the “Log on as batch job” policy is set for the user. This policy is accessible by opening the **Control Panel > Administrative Tools > Local Security Policy**. In the Local Security Policy window, click **Local Policies > User Rights Assignment > Log on as a batch job**:



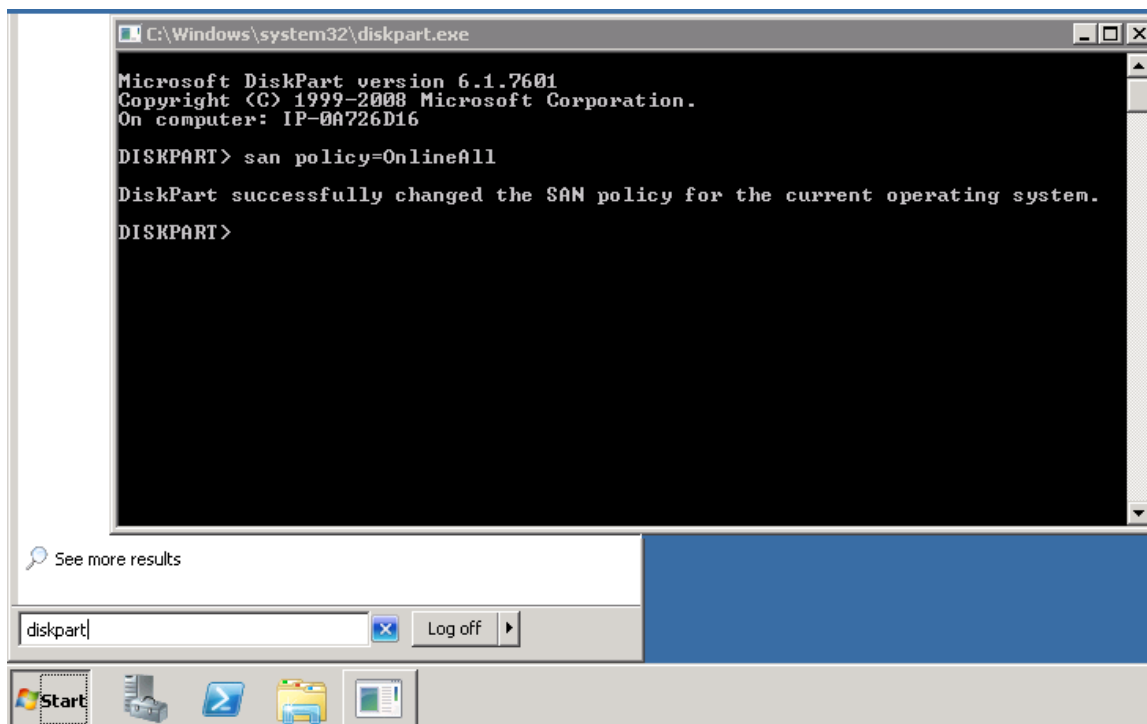
Enabling EBS usage on the instance

Starting with Bamboo 5, you'll be able to use custom Elastic Block Storage with your Windows instances. To do that, you need to change the SAN policy on your instance, otherwise the disks will be attached in 'Offline' state with status text "the disk is offline because of policy set by an administrator".

You can change the SAN policy using the Diskpart utility. Run Diskpart, type

```
san policy=OnlineAll
```

and press Enter. You can then quit Diskpart; the new policy will now be active.



Testing

The easiest way to check if everything is set up correctly is to run the task you've defined using Windows Task Scheduler (**Start > Administrative Tools > Task Scheduler**). Right-click on the task and select **Run**. Always test the script using the Task Scheduler; if you run the script manually, you'll use Administrator account, which is not what we want.

Look for the %USERPROFILE%/bamboo-elastic-agent.out file. If it exists and contains an error message stating that agent was **not** run within an EC2 instance started by Bamboo Server, you've successfully completed the customisation.

Run `c:\opt\bamboo-elastic-agent\bin\prepareInstanceForSaving.bat`

Bundle your instance. Make a note of the AMI id of the new image.

Start your image from Bamboo



If you fail to complete the following steps within ~40 minutes, Bamboo will shut down your instance, so remember to save your work if you're running out of time (i.e. create an interim image).

In Bamboo, define an image configuration for the image you've just created, and start it from Bamboo. If everything went well, the agent will start together with the instance. It will perform the following steps:

- Update/create `/opt/bamboo-elastic-agent` directory structure by creating additional directories. If they appeared, Java is working correctly on that machine and the connection to S3 is working.
- Start the agent, which will create the Bamboo Agent Home directory and populate it with data pulled from the Bamboo server.

If everything went well, you should see the agent appear in the Bamboo instance list. Congratulations, you have a working Bamboo agent.



Because the agent has just synchronised itself with the Bamboo server (because it has downloaded all the jars exactly matching what you have on your server), as an extra step, you may want to save that state to speed up future instance startup and reduce bandwidth usage.

To do that, run

`c:\opt\bamboo-elastic-agent\bin\prepareInstanceForSaving.bat`

save the image, define a new image configuration, kill the instance, and try running it from Bamboo.

Upgrading the agent for your custom elastic image



The instructions below are valid if you were using Bamboo 3.4 or newer. If you're upgrading from an earlier version, you should first reinstall the agent installer (see [Creating a custom elastic image](#)).

If you customised your instance according to [Creating a custom elastic image](#), your instance will keep itself updated across Bamboo. The synchronisation process takes a while and the time required increases as your image gets older. If you notice slow startup, you may want to refresh your image. You can either customise the instance from scratch, as when you created your customised image, or update just the agent data, which is much faster.

Related pages:

- [Managing your elastic image configurations](#)
- [Creating a custom elastic image](#)

To refresh your agent data:

1. Start your instance from Bamboo.
2. Log into your instance.
3. Run `/opt/bamboo-elastic-agent/bin/prepareInstanceForSaving.sh`.
4. Create an Image of your Customised Instance.

The final step is to create an image from your customised instance. To do this, you will require the following information:

- Amazon Account Number
- [Access Key ID](#)
- [Secret Access Key](#)
- [Amazon S3 bucket name](#) that will be used to store image (if you don't have access to Amazon S3, you can sign up on [this page](#).)

You can create an image of your customised instance by using the `ec2-bundle-vol` command, as follows:

```
/usr/local/bin/ec2-bundle-vol -c $EC2_CERT -k $EC2_PRIVATE_KEY -u  
<amazon_account_number> -p <elastic_image_name> --batch --debug
```

where `<elastic_image_name>` is the name that you want to assign to your custom image (e.g. 'CustomImage1')

Once the image is created, you need to upload it to Amazon S3 by running the command below:

```
/usr/local/bin/ec2-upload-bundle -b <s3_bucket_name> -m  
/tmp/<elastic_image_name>.manifest.xml -a <access_key_id> -s  
<secret_access_key> --retry --debug
```

where <s3_bucket_name>, <access_key_id> and <secret_access_key> are the Amazon S3 bucket name, Access Key ID and Secret Access Key described previously, and <elastic_image_name> is the name that you want to assign to your custom image (e.g. 'CustomImage1')

You will then need to register your image with Amazon EC2 by using the `ec2-register` command:

```
$EC2_HOME/bin/ec2-register  
<s3_bucket_name>/<elastic_image_name>.manifest.xml
```

where <s3_bucket_name> is the Amazon S3 bucket name described previously and <elastic_image_name> is the name that you want to assign to your custom image (e.g. 'CustomImage1')

The output of this command will show the AMI ID of your custom image.

5. Associate the new Custom Image with Bamboo.

Finally, you will need to associate your custom elastic image with your Bamboo installation by creating an Elastic Image Configuration. Please note the AMI ID of your new custom image and read [Managing your elastic image configurations](#) for further instructions.

Updating elastic images for Bamboo upgrades

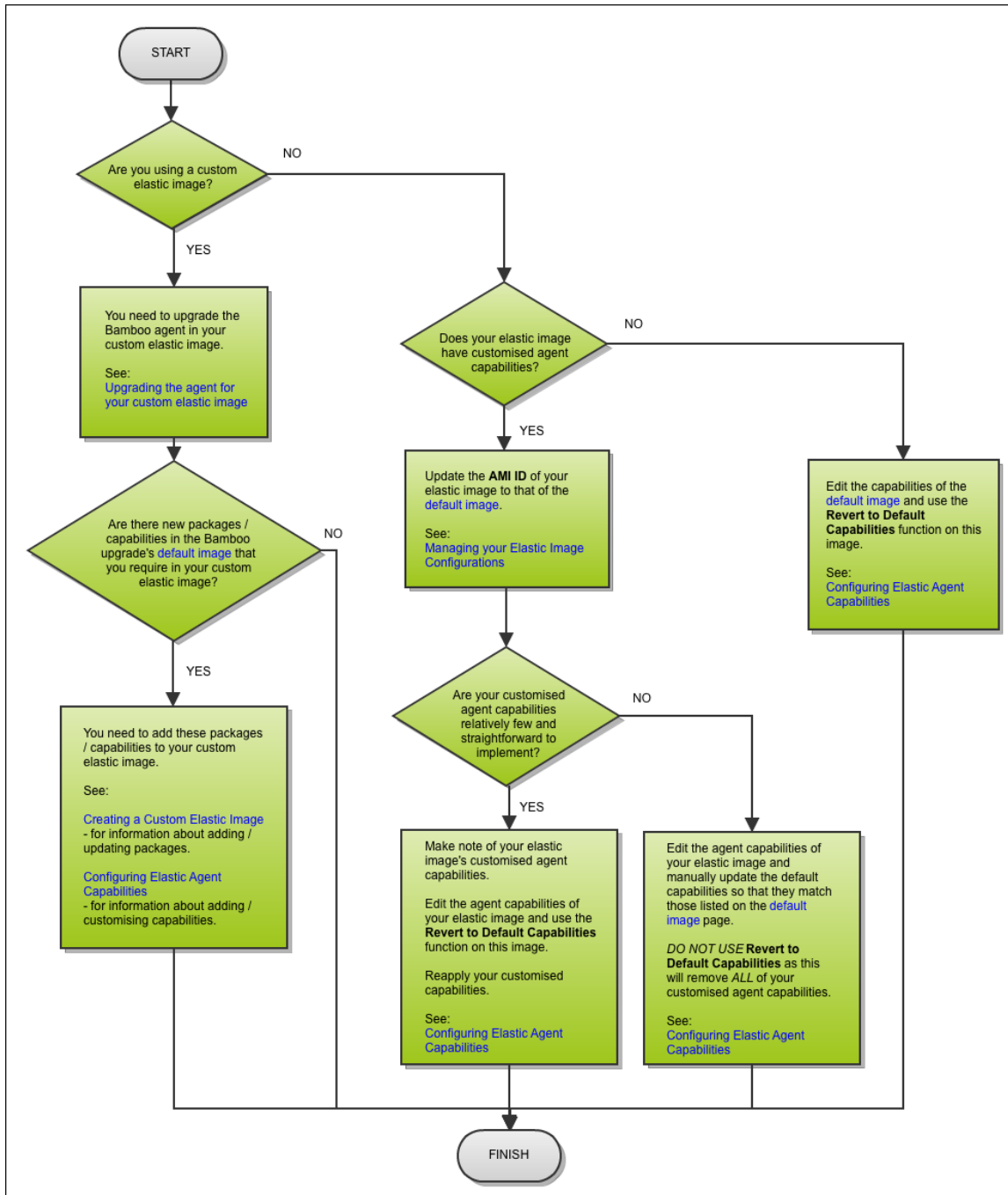
Various updates to default packages and capabilities are made to the [default image](#) with each major release of Bamboo.

Therefore, if you are using either a:

- [custom elastic image](#), or
- [an elastic image with customised agent capabilities](#)

then to ensure this elastic image acquires these package/capability updates, use the flow chart below to update your elastic image.

 Use this flowchart **only after Bamboo has been upgraded**. For each elastic image you wish to update, follow this flow chart from the start.



Elastic Images with Customised Capabilities:

This flow chart assumes that all elastic images with customised agent capabilities are based off the [default image](#). Please check the [default image](#) page to identify the packages and related capabilities available in the default image for **Bamboo 5.0.x**

Bamboo 4.4.x

Managing your elastic instances

An *elastic instance* is a running instance of an [elastic image](#). One elastic instance is created whenever an elastic image is started. Hence, starting one elastic image multiple times, results in the creation of multiple elastic instances. Each time an elastic instance is created, one *elastic agent* is created on that instance.

The following list directs you to details on managing elastic instances manually in Bamboo. However, you can configure Bamboo to automatically manage your elastic instances. Please refer to [Automatic Elastic Instance Management](#) for more information.

- To view a running elastic instance, see [Viewing an elastic instance](#).
- To access your elastic instance via a client, see [Accessing an elastic instance](#).
- To start one or more elastic instances, see [Starting an elastic instance](#).
- To shut down one or more elastic instances, see [Shutting down an elastic instance](#).
- To configure your Elastic Bamboo settings for elastic instances, see the [Elastic Instance Settings](#) section in the [Configuring](#)

Viewing an elastic instance

An *elastic instance* is a running instance of an [elastic image](#). One elastic instance is created whenever an elastic image is started. Hence, starting one elastic image multiple times, results in the creation of multiple elastic instances. Each time an elastic instance is created, one [elastic agent](#) is created on that instance.

Conceptually, an elastic instance can be thought of as a computer. The elastic agent's processes are run on this computer and the elastic image is the boot hard drive. Unlike computers, however, elastic instances are temporary and stateless. When an elastic instance is shut down:

- Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) will not persist
- Any customisations to the instance itself will also be lost.



The Amazon [Elastic Block Store](#) can provide persistent storage for your elastic instances.

You can also view information about your elastic instances on the [AWS Management Console](#). Please note, we strongly recommend that you use the console for *viewing* instance information only. You may experience errors if you attempt to manage your instances outside of Bamboo.

Related pages:

- [Managing your elastic instances](#)

To view an elastic instance:

1. Click **Administration** in the top navigation bar.
2. Click **Instances** in the left navigation panel.
3. Click the name of the instance that you want to view, e.g. 'i-05ff716c'.

Current status	The status of the elastic instance. Values include, 'Pending' (instance starting up), 'Running' and 'Shutting down'.
Public DNS	The public DNS address of the elastic instance. The IP address of the elastic instance is displayed here.
Start Time	The start time of the instance, based on the Amazon EC2 timezone (US Eastern Time for Elastic Bamboo). Start time is the time when you sent the request to start an instance, not the time when the instance progresses to 'Running' status. Up time of the instance (including the time taken for the instance to start up) is shown in brackets after the start time.
Elastic Agent	The elastic agent process currently running on your elastic instance. Currently, Elastic Bamboo only supports one elastic agent per elastic image. Click the link to view the elastic agent. If the agent is running a job, the job's key will be shown in brackets after the elastic agent name.
Current Availability Zone	The availability zone that your elastic instance is running in. Read more about Amazon EC2 availability zones . Your availability zone preference is shown in brackets after the current availability zone. For instructions on how to set the availability zone for your instances, please see Managing your elastic image configurations .
Attached Volumes	The IDs of the attached EBS volumes, if you have configured your elastic instances to use EBS .
Configuration	The name of the elastic image configuration that was used to create this elastic instance. Click the name to configure the elastic image .
AMI ID	The ID of the elastic image (i.e. Amazon Machine Image) that the elastic instance was created from (as part of the elastic image configuration).
EBS Snapshot ID	The ID of the EBS snapshot that was used to create the EBS volumes attached to your instance, if you have configured your elastic instances to use EBS.  <i>Bamboo polls the EBS volumes for an elastic instance every 60 seconds by default. If you want to change this interval, you need to modify the following system property: <code>bamboo.agent.elastic.ebsVolumeSupervisionIntervalInSeconds</code></i>

Instance Type	The instance type of your instance.
SSH Access	Please see Accessing an elastic instance for information on using this function.
Accessing Logs	Please see Accessing an elastic instance for information on using this function.

Screenshot: Viewing an elastic instance

Elastic Bamboo › Instances › i-07ec936b

Information

This is an instance running on the Amazon EC2 compute cloud. You can get more extensive information about this instance from the [Amazon AWS Web Console](#)

Current status  Running

Public DNS ec2-184-72-81-22.compute-1.amazonaws.com
IP: 184.72.81.22

Start Time 14/02/11 10:57 AM (41 minutes ago)

Elastic Agent [Elastic Agent on i-07ec936b \(Idle\)](#)
This is the Bamboo agent that is running in this instance in EC2.

Current Availability Zone us-east-1c (chosen by EC2)

Attached Volumes

vol-fe3c5396

Configuration

Configuration [Maven 2.1 Image](#)
Contains Maven 2.1 and the necessary bits for Selenium 2

AMI ID ami-0ab64563

EBS Snapshot ID snap-68204c00

Instance Type High-CPU Medium

[Shut Down](#)

SSH Access

You can SSH into this instance in the EC2. To do this, simply execute the following command from the bamboo server home directory. Bamboo can not find the elasticbamboo.pk file on the server. For more information on where to find your elasticbamboo.pk file see our [online documentation](#)

```
ssh -i elasticbamboo.pk root@ec2-184-72-81-22.compute-1.amazonaws.com
```

Accessing Logs

You can use SCP to download the logs from this EC2 instance. To do this, simply execute the following command.

```
scp -i elasticbamboo.pk root@ec2-184-72-81-22.compute-1.amazonaws.com:/home/bamboo/bamboo-elastic-agent.out ./
```

Amazon EC2 Console

You can manage your EC2 instances using the [Amazon EC2 Console](#). Once youve logged in through the console, you can [access logs for this instance directly](#). Logs are reproduced below in the iframe below. (note that the logs are usually a little delayed).



Accessing an elastic instance

It is possible to connect directly to a running [elastic instance](#) to access logs or upload files. Access is available via SSH (secure shell) and file transfer is enabled via SCP (secure copy).

Please note, you can only access elastic instances that are running. You may need to configure the automatic termination of elastic instances.

On this page:

- [Using SSH](#)
- [Using SCP](#)
- [Notes](#)

Related pages:

- [Managing your elastic instances](#)

Using SSH

To access your elastic instance using SSH:

1. Navigate to the desired elastic instance, as described on [Viewing an elastic instance](#).
2. Copy the command text listed under the 'SSH Access' section. It will be similar to the following example command text:

```
ssh -i /opt/bamboo/home/xml-data/configuration/elasticbamboo.pk  
root@ec2-68-111-185-197.compute-1.amazonaws.com
```
3. Execute the text in your terminal and you will have full SSH access to the Elastic Instance.

 You can also download the private key via the link in the 'SSH Access' section to access your elastic instance via SSH. Click 'here' in the following text on screen to download the key: 'You can also download the SSH private key file from here and use the private key to access the EC2 instance.'

Using SCP

Note, you can also use SCP to upload files to your elastic instance.

To access your elastic instance using SCP:

1. Navigate to the desired elastic instance, as described on [Viewing an elastic instance](#).
2. Copy the command text listed under the 'Accessing Logs' section. It will be similar to the following example command text:

```
scp -i /opt/bamboo/home/xml-data/configuration/elasticbamboo.pk  
root@ec2-68-111-185-197.compute-1.amazonaws.com:/home/bamboo/bamboo-elastic-agent.out ./
```
3. Execute the text in your terminal to download the logs from your elastic instance.

Notes

- **Permission issues for SSH access** — If you are experiencing permission issues when attempting to access your elastic instance via SSH, you may need to modify permissions on your Elastic Bamboo private key file. See [this FAQ](#) for further details.

Starting an elastic instance

An [elastic agent](#) process runs in an [elastic instance](#) and will automatically start when an instance is started. If you want to [run a Job build on an elastic agent](#), you can start an elastic instance for the agent to run in. The elastic agent will inherit the capabilities of the [image](#) that the instance is started from.

Limitations on the number of elastic instances — An elastic agent is counted as a remote agent for licensing purposes. Hence, if starting an elastic instance (and hence an elastic agent) causes you to exceed the total number of remote agents allowed under your license, you will not be able to start the instance.

Related pages:

- [Managing your elastic instances](#)

To start an elastic instance:

1. Click **Administration** in the top navigation bar.
2. Click **Instances** in the left navigation panel.
3. Click **Start New Elastic Instances**.
 - Use **Number of instances** to specify the number of new instances you would like to start.
 - Use **Elastic Image Configuration Name** to select the [elastic image configuration](#) that you would like your instances to use.
4. Click **Submit**. The 'Manage Elastic Instances' page will be displayed, showing your new instances starting:
 - a. A note will display stating that the elastic instances (and corresponding agents) are starting.

 1 new elastic instance is pending. New instances and corresponding elastic agents may take a few minutes to start up.

- b. Your elastic instances will then display with a status of 'Pending' while they start up. This generally takes a few minutes.

Instance → Agent	Status	Up Time	Operations
Instance i-2204914b	 Pending	8 seconds	View Shut Down

- c. Once your elastic instances have started up, they will progress to 'Running' status. An elastic agent process will then start up for each instance. They will display a status of 'Pending' while they start.

Instance → Agent	Status	Up Time	Operations
Instance i-2204914b	 Running	1 minute	View Shut Down
 Elastic Agent on i-2204914b	 Pending		

- d. Once the elastic agents have started, they will display a status of 'Online'.

Instance → Agent	Status	Up Time	Operations
Instance i-2204914b	 Running	10 minutes	View Shut Down
 Elastic Agent on i-2204914b	 Online		Disable

Notes

What if my elastic agent doesn't start? Bamboo has a set period of time that it waits for the agent to start on an elastic instance. If no response is received by the end of this time period, Bamboo will shut down the elastic instance.

You can configure this time period by modifying the following system property (default is 600):

`bamboo.agent.elastic.startupTimeoutSeconds`

Read [Configuring system properties](#) for instructions on how to set a system property.

Scheduling your elastic instances

You can schedule the startup and shutdown of elastic instances in Bamboo. For example, you may wish to shut down all elastic instances on weekends or start up additional instances to help cope with job builds during regular busy periods.

On this page:

- [Managing your elastic instance schedules](#)
- [Adding a new elastic instance schedule](#)

Related pages:

- [Managing your elastic instances](#)

Managing your elastic instance schedules

To manage your elastic instance schedules:

1. Click **Administration** in the top navigation bar.
2. Click **Instance Schedule** in the left navigation panel (under 'Elastic Bamboo').
3. Do any of the following:

Task	Action
Add a new schedule	Click Add Elastic Instance Schedule to create a schedule from new. Click Copy to use an existing schedule as a template. See the Adding a New Elastic Instance Schedule section below for further instructions.
Edit an existing schedule	Click Edit for an existing schedule. You can also Delete existing schedules.
Enable existing schedules	Click Enable for a particular schedule, or click Enable All .
Disable existing schedules	Click Disable for a particular schedule, or click Disable All .

You can also view the configuration for the elastic image that the instances will be created from, by clicking the image configuration name (e.g. 'Default') in the table of schedules.

Screenshot: Viewing elastic instance schedules

View Elastic Instance Schedules

Add Elastic Instance ScheduleEnable AllDisable All

Configure when to start up or shut down elastic instances of a particular elastic image.

Next Scheduled Run	Cron Expression	Image Config	Active Instances	Operations
Trigger disabled	Each Saturday at 4:30 am	Stop all elastic instances		Edit Enable Copy Delete
Trigger disabled	Each Monday at 9:00 am	Stop all elastic instances		Edit Enable Copy Delete

Adding a new elastic instance schedule

1. Click **Administration** in the top navigation bar.
2. Click **Instance Schedules** in the left navigation panel (under 'Elastic Bamboo').
3. Click either **Add Elastic Instance Schedule** to create a schedule from new, or **Copy** for an existing schedule to use it as a template.

Enabled	Clear if you do not want this schedule to be enabled when you create it.
Trigger On	Choose when this schedule should start: • Next Bamboo startup • A cron schedule — edit Schedule as required. For information on constructing cron expressions, see this FAQ.
On Trigger Bamboo Should	Choose the action Bamboo should perform: • Stop all elastic instances • Adjust number of active instances
Image Config	Choose which image the elastic instances should be started from. The elastic agents running on the instances will inherit the capabilities from the image.
Active Instances	Choose the logical operator and specify a value for the number of active instances.

Screenshot: Adding an elastic instance schedule

Add Elastic Instance Schedule

Choose when your schedule will run and how many elastic instances of a particular configuration you want active at any point in time. When bringing the number of instances down, idle instances will be shut down preferentially. Instances with busy agents will be brought down after the builds they are running have completed.

Schedule Details

☒ Enabled

Trigger On ☐ Next Bamboo startup
☒ A cron schedule

Schedule* **Daily at 11:50 pm** 

On Trigger Bamboo Should ☐ Stop all elastic instances
☒ Adjust number of active instances

Image Config

Active Instances
Number of instances that Bamboo will attempt to adjust to.

Shutting down an elastic instance

We recommend that you shut down any elastic instances that are not being used. Amazon EC2 charge for the period of time that you have an instance running, so you can minimise your costs simply by shutting down instances with inactive agents. You should also shut down your elastic instances if you are going to restart your Bamboo server, otherwise you will orphan them from your Bamboo server.

If you have set up automated procedures via the [Bamboo Remote API - Deprecated](#) to terminate agents (e.g. cron jobs), you can also configure Elastic Bamboo to automatically shut down instances after the agent processes terminate.

On this page:

- [Shutting down an elastic instance](#)
- [Shutting down all elastic instances](#)
- [Configuring automatic shutdown of instances after agent termination](#)
- [Shutting down elastic instances using the AWS Console](#)

Related pages:

- [Managing your elastic instances](#)

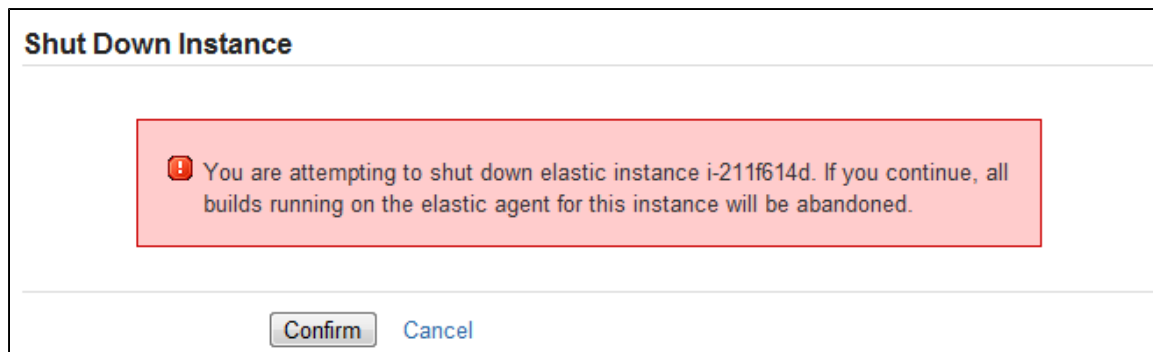
Shutting down an elastic instance

Before you begin:

- Please ensure that the agent on an elastic instance is not running a job build, before shutting down the instance. Any job builds running on the agent **will be abandoned** when you shut down the elastic instance.

To shut down an elastic instance:

1. Click **Administration** in the top navigation bar.
2. Click **Instances** in the left navigation panel. The 'Manage Elastic Instances' screen will display.
3. Click **Shut Down** for the instance that you wish to shut down (in the 'Operations' column).
4. Click **Confirm**. In the 'Manage Elastic Instances' screen, the elastic instance that you have shut down will show a 'Shutting down' status for a few minutes, before it shuts down and disappears from this screen.



Shutting down all elastic instances

Before you begin:

- Please ensure that the agent on an elastic instance is not running a Job build, before shutting down the instance. Any Job builds running on the agent **will be abandoned** when you shut down the elastic instance.

To shut down all elastic instances:

1. Click **Administration** in the top navigation bar.
2. Click **Instances** in the left navigation panel. The 'Manage Elastic Instances' screen will display.
3. Click **Shut Down All Instances**. The 'Shut Down All Instances' screen will display.
4. Click **Confirm**. The 'Manage Elastic Instances' screen will display again. The elastic instances will display 'Shutting down' status' for a few minutes, before they shut down and disappear from this screen.

Configuring automatic shutdown of instances after agent termination

To configure Elastic Bamboo to automatically shut down instances when agents are terminated:

Please refer to [Configuring Elastic Bamboo](#) and follow the instructions for setting the **Automatically shut down elastic instance when elastic agent process ends** option in the 'Elastic Bamboo Global Settings' section.

Shutting down elastic instances using the AWS Console

We **strongly recommend** that you manage your instances using the Elastic Bamboo user interface. If your elastic instances become orphaned from your Bamboo server, you may need to shut your elastic instances down directly in the Amazon Web Services (AWS) console.

Your elastic instances can become orphaned from your Bamboo server, for example if you restart your Bamboo server without shutting down your elastic instances first.

Please refer to [How do I shut down my elastic instances if I have restarted my Bamboo server](#) for further details.

Managing your elastic agents

An *elastic agent* is a [remote agent](#) that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#). An elastic agent process runs in an [elastic instance](#) of an [elastic image](#). An elastic agent inherits its capabilities from the [elastic image](#) that it was created from.

- To view your elastic agents, see [Viewing your elastic agents](#).
- To view elastic agents that have terminated, see [Viewing your elastic agent usage history](#).
- To configure your elastic agent's capabilities, see [Configuring elastic agent capabilities](#).
- To disable an elastic agent, see [Disabling an Elastic Agent](#).

Viewing your elastic agents

An *elastic agent* is a [remote agent](#) that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#). An elastic agent process runs in an [elastic instance](#) of an [elastic image](#). An elastic agent inherits its capabilities from the [elastic image](#) that it was created from.

An elastic agent will always have an 'Online' status, (i.e. 'Online' or 'Online (Disabled)'). If you [disable an elastic agent](#), the elastic instance will remain online. However, if you shut down the elastic instance, then the elastic agents process is killed and will not appear in the remote agents list. Hence, an elastic agent will never have an 'Offline' status.

Related pages:

- [Managing your elastic agents](#)

To view your elastic agents:

1. Click **Administration** in the top navigation bar.
2. Click **Agents** in the left navigation panel.

The agents for your Bamboo instance will be displayed (see screenshot below). Any elastic agents that are running will be listed in the 'Remote Agents' section. The elastic agent name will be prefixed with 'Elastic Agent', e.g. 'Elastic Agent on i-2204914b'

Screenshot: Elastic agents

Agents

An agent is a service that executes Bamboo builds. You can use this page to view, add and delete agents. You can also use this [matrix](#) to determine which agents can execute which build plans.

Local Agents

[Server Capabilities](#)

Local agents run on the Bamboo server.

Select: [All](#), [None](#), [Idle](#), [Disabled](#) Action: [Delete](#) [Disable](#) [Enable](#)

	Agent	Status	Operations
☐	Local Agent	 Building - SANDBOX-RFIN-JOB1-22	View Edit
☐	Reporting agent	 Idle (Disabled)	View Edit

[Add Local Agent](#)

Remote Agents

[Shared Remote Capabilities](#)

Remote agents run on computers other than the Bamboo server.

[Online Remote Agents](#) [Offline Remote Agents](#)

There are currently 3 remote agents online (3 elastic). Start [elastic agents here](#). A maximum of 25 agents are [supported](#) by your license.

Select: [All](#), [None](#), [Idle](#), [Disabled](#) Action: [Delete](#) [Disable](#) [Enable](#)

	Agent	Status	Operations
☐	 Elastic Agent on i-01bcc86d	 Idle	View Edit
☐	 Elastic Agent on i-03bcc86f	 Idle	View Edit
☐	 Elastic Agent on i-1fbcc873	 Building - BTS-FUNC-JOB1-7816	View Edit

```
Feb 11, 2011 10:13:43 AM A remote agent is loading on ip-10-204-186-247.ec2.internal (172.20.6.108).
Feb 11, 2011 10:13:44 AM A remote agent is loading on ip-10-102-61-209.ec2.internal (172.20.6.108).
Feb 11, 2011 10:13:45 AM A remote agent is loading on ip-10-102-61-80.ec2.internal (172.20.6.108).
Feb 11, 2011 10:13:49 AM A remote agent is loading on ip-10-102-61-238.ec2.internal (172.20.6.108).
Feb 11, 2011 10:13:50 AM A remote agent is loading on ip-10-204-185-205.ec2.internal (172.20.6.108).
Feb 11, 2011 10:14:00 AM A remote agent is loading on ip-10-204-185-229.ec2.internal (172.20.6.108).
Feb 11, 2011 10:15:40 AM Remote agent "Elastic Agent on i-01bcc86d" has registered.
Feb 11, 2011 10:15:43 AM Remote agent "Elastic Agent on i-03bcc86f" has registered.
Feb 11, 2011 10:15:50 AM Remote agent "Elastic Agent on i-1fbcc873" has registered.
```

These are live logs that are refreshed every 10 seconds, you will need to refresh the page to see any other updates.

[Install Remote Agent | Disable Remote Agent Support](#)

Viewing your elastic agent usage history

When you shut down an elastic instance, the agent process for that instance is killed. Consequently, the elastic agent will not display an offline status, but will be removed altogether from your [available elastic agents](#) in Bamboo.

However, information about these elastic agents is recorded in Bamboo and can be viewed on the 'Elastic Agent History' page.

Related pages:

- [Managing your elastic agents](#)

To view the history of an elastic instance that has been shut down:

1. Click **Administration** in the top navigation bar.
2. Click **Agent History** in the left navigation panel.
3. To view the usage history of the elastic agent, click the agent name, or **View** next to the agent. The 'Elastic Agent History' page (see screenshot) will show the following information:
 - Elastic instance — the elastic instance that the elastic agent ran in.
 - Last startup time — the last time that the elastic agent was started. This is based on the Bamboo server time.
 - Last shutdown time — the last time that the elastic instance was stopped. This is based on the Bamboo server time.
 - Up time — the total time that the elastic agent was online.
 - Build History — this table lists the job builds run by the elastic agent and information about the job build, such as the status, duration, test results, etc. You can access the full results by clicking the build number.

Screenshot: Elastic agent history

Elastic Agent History › Elastic Agent on i-e9ec9385 (elastic)

Elastic agent on instance i-e9ec9385, configuration: Maven 2.1 Image

You can view historical information for this elastic agent in this page. This agent is offline and will never come back alive.

Agent Details

50% 1 / 2 Successful	Elastic instance Instance i-e9ec9385 (Maven 2.1 Image image configuration) The elastic instance this agent is/was running on.
Last startup time	14/02/2011 11:05:39 AM
Last shutdown time	14/02/2011 12:56:22 PM
Up time	1 hour, 50 minutes

Build History

The agent was building for **28.15%** of the up time.

Status	Reason	Completed	Duration	Test Results
✓ Bamboo Testing › Remote Agent Functional Tests on JDK 1.6 › #1088 › Default Job	Scheduled build	Mon, 14 Feb, 11:52 AM	21 minutes	13 passed
! Bamboo Main › Stable CI Tests › #248 › Func Test Repository	Manual build by James Dumay	Mon, 14 Feb, 11:14 AM	6 minutes	No tests found

Showing 1-2 of 2

Configuring elastic agent capabilities

An *elastic agent* is a [remote agent](#) that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#). An elastic agent process runs in an [elastic instance](#) of an [elastic image](#). An elastic agent inherits its capabilities from the [elastic image](#) that it was created from.

You can customise the capabilities of your elastic agents by configuring the capabilities on the relevant elastic image.

You may want to configure the capabilities on your elastic image to force your job builds to run on particular elastic agents (e.g. running slow acceptance tests on your most powerful elastic agents). You may also need to configure the capabilities on any [custom elastic images](#) that you have created and/or associated with your Bamboo installation.

Please note that adding a builder, JDK or Perforce capability to the image does not install the actual builders, JDKs or Perforce modules on the image. Please take particular note of this, if you are adding capabilities to a custom image.

Related pages:

- [Managing your elastic agents](#)

To configure the capabilities on an elastic image:

1. Click **Administration** in the top navigation bar.
2. Click **Configuration** in the left navigation panel (under 'Elastic Bamboo').

3. Click the name, or **View**, for the elastic image whose capabilities you want to configure. The configuration screen will be displayed, showing the capabilities of the image.
4. You can add new capabilities to the image using the 'Add Capability' panel at the bottom of the screen. Please see the following pages for further information:
 - [Defining a new executable capability](#)
 - [Defining a new JDK capability](#)
 - [Defining a new version control capability](#)
 - [Defining a new custom capability](#)
5. You can also edit, rename or delete a capability from an elastic image. Please see the following pages for further information:
 - [Configuring capabilities](#)
 - [Renaming a capability](#)
6. You can also view the agents and elastic image configurations with a particular capability and the jobs with the related requirement by clicking **View** for the capability.
7. Any changes that you have made to elastic agent capabilities will only be reflected in new agents started after the changes were made. You will need to restart any existing agents, if you want them to pick up your changes.

Screenshot: Configuring elastic agent capabilities

Elastic Image Capabilities

[Add Capability](#) | [Revert to Default Capabilities](#)

A capability is a feature of an agent. There are 3 types of capabilities: builders, JDKs and custom. You can use this page to view, add and delete capabilities associated with this Elastic Image Configuration. Any existing elastic instances will need to be restarted to pick up these changes.

The following capabilities exist on Elastic Agents running on an instance of this Image:

Custom

'Custom' capabilities are key-value pairs that define particular characteristics of an agent (e.g. 'operating.system=WindowsXP', 'fast.builds=true'). For an agent to be able to build a job, both the 'Key' and 'Value' must match the jobs's requirements.

Key	Value	Operations
bamboo.functionalTest	true	View Edit Delete

Builder

'Builder' capabilities define the builders which are available to your build plans.

Builder Label	Path	Operations
Ant (Ant)	/opt/apache-ant-1.7.1	View Edit Delete
Maven 2 (Maven 2.x)	/opt/maven-2.0	View Edit Delete
Maven 2.1 (Maven 2.x)	/opt/maven-2.1	View Edit Delete
Maven 2.2 (Maven 2.x)	/opt/maven-2.2	View Edit Delete

JDK

'JDK' capabilities define the JDKs which are available to your build plans.

JDK Label	Java Home	Operations
JDK	/opt/jdk-5	View Edit Delete
JDK 1.5	/opt/jdk-5	View Edit Delete
JDK 1.6	/opt/jdk-6	View Edit Delete

Mercurial

The path to the Mercurial executable (e.g. 'C:\Program Files (x86)\Mercurial\hg.exe' or '/usr/local/bin/hg')

Executable	Path	Operations
Mercurial	/usr/bin/hg	View Edit Delete

Add Capability

Capability Type

Key

Value

[Add](#) [Cancel](#)

Disabling an Elastic Agent

An *elastic agent* is a [remote agent](#) that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#). An elastic agent process runs in an [elastic instance](#) of an [elastic image](#). An elastic agent inherits its capabilities from the [elastic image](#) that it was created from.

If you would like to stop an elastic agent, you can disable it in Bamboo. This will abandon any job build it is running and prevent it from running any further job builds.

Please note, disabling an elastic agent will not shut down the elastic instance it is running on (i.e. you will still be charged for the instance uptime). You can permanently stop an elastic agent and instance by [shutting down the elastic instance](#).

The Bamboo server also "supervises" your elastic agents. If the Bamboo server detects that an elastic agent is offline, it will automatically terminate the elastic instance.

Related pages:

- [Managing your elastic agents](#)

To disable an elastic agent:

1. Navigate to the desired elastic agent, as described in [Viewing your elastic agents](#).

Instance → Agent	Status	Up Time	Operations
Instance i-2776e24e	⚡ Running	5 minutes	View Shut Down
↪ Elastic Agent on i-2776e24e	🟢 Online		Disable

2. Click **Disable** in the 'Operations' column for the elastic agent. The elastic agent will display with a status of 'Online (Disabled)'.
 Re-enable the elastic agent by clicking **Enable**.

Instance → Agent	Status	Up Time	Operations
Instance i-2776e24e	⚡ Running	6 minutes	View Shut Down
↪ Elastic Agent on i-2776e24e	🟢 Online (Disabled)		Enable

Running job builds using Elastic Bamboo

This page provides answers to common questions about running builds using Elastic Bamboo. If you are using Elastic Bamboo for the first time, we highly recommend that you read [Getting started with Elastic Bamboo](#) for instructions on setting up Elastic Bamboo and running your first build.

What job builds can I run on Elastic Bamboo?

You can run any of your job builds on any [elastic agent](#) (which in turn runs on an [elastic instance](#)), provided that the elastic agent's capabilities meet the job's requirements. An elastic agent inherits the capabilities of the [elastic image](#) it was created from. Hence, you can see which of your jobs can run on elastic agents by checking that your job's requirements match your [elastic image's capabilities](#).

✓ You can view your elastic image and the job builds that meet its requirements on the [Agents and Plans matrix](#).

On this page:

- [What job builds can I run on Elastic Bamboo?](#)
- [How do I run a plan build and its jobs on an elastic agent?](#)
- [How do I automatically start or shut down elastic instances for job builds?](#)
- [How do I know whether my job build was run on an elastic agent?](#)
- [How do I customise the capabilities of my elastic agents?](#)
- [How much does it cost to run a build?](#)
- [What is EBS and how does it affect my job builds?](#)

How do I run a plan build and its jobs on an elastic agent?

An [elastic agent](#) operates in a similar way to a non-elastic agent. The Bamboo server will determine if any job builds in the queue can be built on any of the available agents (including elastic agents), based on whether or not the capabilities of these agents meet the requirements of these jobs.

If an available elastic agent (like any other available agent) has capabilities which meet the requirements of a build in the build queue, the Bamboo server will assign the job build to that elastic agent.

If you do not have any free elastic agents running, you can [configure Bamboo](#) to automatically start up elastic instances whose elastic agents are capable of running job builds in the queue, or you can start up an appropriate elastic instance manually. (When an [elastic instance](#) is started, its elastic agent is also started, automatically.) For more information about starting elastic instances manually, refer to [Starting an elastic instance](#).

If you do not use Bamboo's **Automatic Elastic Instance Management** feature and prefer to manage your elastic instances manually, then we strongly recommend that you **shut down** any elastic instances (running your elastic agents), when they are not in use. Minimising unutilised elastic instance uptime will help reduce costs. Read [Shutting down an elastic instance](#) for instructions on how to shut down an elastic instance.

How do I automatically start or shut down elastic instances for job builds?

Bamboo can automatically start elastic instances based on demand from the build queue and shut them down once the elastic agents running on them have been idle for a specified period of time. For more information, please refer to the [Automatic Elastic Instance Management](#) section of the [Configuring Elastic Bamboo](#) topic.

While Bamboo's **Automatic Elastic Instance Management** feature is the easiest and most effective method of managing elastic instances in Bamboo, you can also manage elastic instances via the [Bamboo Remote API - Deprecated](#) . For example, you could implement cron jobs to intelligently start and stop elastic instances, so that elastic agents are available at key times for your job builds.

How do I know whether my job build was run on an elastic agent?

The name of the image and elastic agent that ran a job build can be viewed as part of the build result. Please see the [Viewing a build result](#) page for more information.

How do I customise the capabilities of my elastic agents?

You may want to customise the capabilities of your [elastic agents](#) to suit certain jobs in your plans. For example, if you want to force certain job builds to only run on elastic agents, you can add a custom capability of `elastic=true` to your elastic agents and add the same requirement to these jobs.

To customise the capabilities for your elastic agents, you need to customise the capabilities of the [image](#) that they are created from. Read [Configuring elastic agent capabilities](#) for instructions.

How much does it cost to run a build?

As Elastic Bamboo usage varies from customer to customer, we cannot provide a definitive cost estimate for running a job build using Elastic Bamboo. We do provide high level guidelines for Elastic Bamboo costs, based on our own experience of using Elastic Bamboo at Atlassian, on the [Elastic Bamboo Costs](#) page.

You can significantly reduce the costs and time taken to run a job build by configuring Elastic Bamboo to use [Automatic Elastic Instance Management](#) and [Amazon's Elastic Block Store \(EBS\)](#).

What is EBS and how does it affect my job builds?

The [Amazon Elastic Block Store \(EBS\)](#) provides persistent storage volumes that can be attached to EC2 instances. Elastic Bamboo can use the EBS to store snapshots of relatively static build information, such as checkouts of source code and Maven repository data. You can choose a snapshot to create EBS volumes from. These volumes can then be attached to your elastic instances when they start up.

Disabling Elastic Bamboo

If you do not want to execute Plan builds and their Jobs in the Amazon EC2 anymore, you can disable Elastic Bamboo for your Bamboo installation. Your AWS account details will be preserved when you disable Elastic Bamboo, so you can just enable it if you want to start using it again.

Related pages:

- [Configuring Elastic Bamboo](#)

Before you begin:

- Please ensure that you do not require your elastic agents before disabling Elastic Bamboo, as they will be **stopped immediately**.

To disable Elastic Bamboo:

1. Click **Administration** in the top navigation bar.
2. Click **Configuration** in the left navigation panel (under 'Elastic Bamboo').
3. Click **Disable**. Elastic Bamboo will be disabled and a confirmation message will be displayed.

Users and permissions

Bamboo provides several options for user management:

- Manage users and groups in Bamboo.
- Manage users and groups in [Atlassian's JIRA](#)
- Connect to an external user directory, such as [Atlassian's Crowd](#) or an LDAP server.

Note that this information *does not* relate to application-level security for Bamboo. If you are looking for information on security of the Bamboo application, please refer to the [Security](#) page.

On this page:

- [About users and authors](#)
- [About groups](#)
- [About permissions](#)

Related pages:

- [Configuring a plan's permissions](#)
- [Managing users](#)
- [Managing groups](#)
- [Connecting to external user directories](#)
- [Managing permissions](#)

About users and authors

An *author* is any person who checks in code to a repository that is associated with a Bamboo [plan](#). An author need not be a Bamboo user. Depending on your organisation's needs, you can configure Bamboo to [grant access to non-users](#). However, only Bamboo users can:

- view the **My Bamboo** tab on the [Dashboard](#).
- belong to a [group](#).

About groups

Bamboo *groups* are used to specify which [users](#) will have [global permissions](#) and [plan permissions](#). They can also be used to specify which users will receive [notifications about a plan's build results](#). You can create and delete as many groups as you need. You will typically create at least one group per [project](#).

A special group called **bamboo-admin** is automatically created when you install Bamboo. Members of this group have [Bamboo administration rights](#).

About permissions

A *plan permission* is the ability to perform a particular operation on a [plan](#) and its [jobs](#). For each plan, different permissions can be granted to particular groups and/or users. A *global permission* is the ability to perform a particular operation in relation to Bamboo as a whole.

Managing users

Bamboo provides several options for user management:

- Manage users and groups in Bamboo — see
- Manage users and groups in [Atlassian's JIRA](#) or [Atlassian's Crowd](#) — see [Integrating Bamboo with Crowd](#).
- Connecting to an external user directory, such as an LDAP server — see [Integrating Bamboo with LDAP](#).

To choose how users are managed in Bamboo:

1. Navigate to **Administration > User Repositories** (under 'Security').
2. Choose one of the user management options:
 - **Local users and groups** — manage users and groups in Bamboo.
 - **Users and groups from JIRA or Crowd** — manage users and groups using Atlassian's Crowd. For instructions on how to connect Bamboo to Crowd, read [Integrating Bamboo with Crowd](#).
 - **Custom user repository** — Choose this option to manage your users and groups via an LDAP server or a custom repository. For instructions on how to connect Bamboo to an LDAP server, read [Integrating Bamboo with LDAP](#).
3. Click **Save**.

For information about tasks for managing users, see the following topics:

- [Creating a user](#)
- [Changing a user's password or details](#)
- [Deleting or deactivating a user](#)
- [Granting administration rights to a user](#)
- [Changing usernames](#)

Notes

An *author* is any person who checks in code to a repository that is associated with a Bamboo [plan](#). An author need not be a Bamboo user. Depending on your organisation's needs, you can configure Bamboo to [grant access to non-users](#). However, only Bamboo users can:

- view the **My Bamboo** tab on the [Dashboard](#).
- belong to a [group](#).

Creating a user

A *user* is someone who can log in to Bamboo.

Depending on your organisation's needs, you can configure Bamboo to [grant access to non-users](#). However, only Bamboo users can:

- view the **My Bamboo** tab on the [Dashboard](#).
- belong to a [group](#).

Related pages:

- [Managing users](#)

To create a Bamboo user:

1. Choose **Administration**, and then **Users** in the left navigation panel.
2. Complete the 'Add User' form.

Field	Notes
Username	Username cannot be changed after the user is created.
Password	The user can easily change their password later.
Email	The address to which notifications are sent.
Instant Messaging Address	If no IM address is specified, Bamboo will not be able to recognise the user's <i>context</i> when interacting using IM.
Source Repository Aliases	If the user is a Bamboo author, click Add Alias to enter the user's login name for their source-code repository. If you don't know their login name, they can specify it themselves later.

Changing a user's password or details

To change a user's password or details:

1. Click **Administration** in the menu bar.
2. Click **Users** in the left panel (under 'Security').
3. Locate the user by typing part of their username, full name or email, and clicking **Search**. This will display a list of matching users.
4. Click **Edit** for the required user.
5. Edit the user's details or password as necessary. See [Creating a user](#).
 - If you have configured [SMTP email](#) on your Bamboo server, the user will automatically receive an email containing their new password.
 - The user can easily [change their password](#) later.
6. Click **Save**.

Related pages:

- [Managing users](#)

Notes

- Users who have forgotten their passwords can click the **Forgotten your password?** link on the Bamboo login screen. This will automatically generate a new password and email it to the user (provided the Bamboo server has been [configured to send SMTP email](#)).
- Logged-in users can also change their own password and details, as described in [Managing your user profile](#).
- See [Associating your author name with your user profile](#) for information about **Source Repository Aliases**.

Deleting or deactivating a user

Deleting a user removes their Bamboo user account. Deactivating a user prevents them from logging in to Bamboo.

Deleting a Bamboo user

Before you begin:

- Deleting a Bamboo user will not delete their *author* data — that is, their [author statistics](#) and [code check-in comments](#) will still exist in Bamboo.
- You cannot delete a user who has created [labels](#) or [comments about build results](#). You may want to deactivate them instead.
- You cannot delete the user account with which you are currently logged in to Bamboo.

On this page:

- [Deleting a Bamboo user](#)
- [Deactivating a Bamboo user](#)

Related pages:

- [Managing users](#)

To delete a Bamboo user:

1. Choose **Administration**, and then **Users** in the left navigation panel.
2. Use the **Delete** link in the 'Operations' column.

Manage Users

You can use this page to view, update and delete users.

Username

Full Name

Email

Search

Enter your search criteria in one or more of the fields above. Wildcards (*) are permitted at the beginning and end of your search terms.

Username	Email	Full Name	Groups	Source Repository Alias	Operations
aahmed	aahmed@atlassian.com	Adam Ahmed	atlassian-dev atlassian-dev-tools atlassian-staff jira-users	aahmed	Edit Delete
aasonitis	aasonitis@atlassian.com	Amelia Asonitis	atlassian-finance atlassian-staff jira-users		Edit

Deactivating a Bamboo user

Deactivate a Bamboo user account (rather than deleting it), requires you to change the password so that the user cannot login.

To deactivate a Bamboo user:

1. Choose **Administration**, and then **Users** in the left navigation panel.
2. Click **Edit** for the user to be deactivated.
3. Enter a new password for the user.
 If you have configured SMTP email on your Bamboo server, the user will automatically receive an email containing their new password.
4. To get around the email problem, enter an invalid email address in the **Email** field, for example `foobar@foobaremailaddress.foobar`.
5. Delete the user's **Instant Messaging Address** so that he or she does not receive notifications on build events.
6. Click **Save**.

Granting administration rights to a user

In Bamboo, there are two types of administrators:

- *Global administrators* — that is, people with the 'Admin' [global permission](#). These people can access the Bamboo **Administration** menu. They can also administer every plan.
- *Plan administrators* — that is, people with the 'Admin' and 'Edit' [plan permissions](#). These people can administer a particular plan.

Granting global administration rights to a user

To grant global administration rights to a user:

- Either grant the 'Admin' global permission to the user explicitly (as described in [Granting global permissions to users or groups](#)); OR:
- Add the user to a *group* which has the 'Admin' global permission (as described in [Changing members of groups](#)).

On this page:

- [Granting global administration rights to a user](#)
- [Granting plan administration rights to a user](#)

Related pages:

- [Managing users](#)
- [Granting global permissions to users or groups](#)
- [Adding users to and removing them from groups](#)
- [Granting plan permissions in bulk](#)

Granting plan administration rights to a user

- Either grant the 'Admin' and 'Edit' plan permissions to the user explicitly (as described in [Granting plan permissions in bulk](#)); OR:
- Add the user to a *group* which has the 'Admin' and 'Edit' plan permissions (as described in [Changing members of groups](#)).

Changing usernames

The script below has been used by an Atlassian customer to change a username in Bamboo 3.1. No guarantees are implied by its presence here.

```
update ACL_ENTRY set SID='newusername' where SID='oldusername';
update ACL_OBJECT_IDENTITY set OWNER_SID='newusername' where
OWNER_SID='oldusername';
update AUDIT_LOG set USER_NAME='newusername' where
USER_NAME='oldusername';
update AUTHOR set LINKED_USER_NAME='newusername' where
LINKED_USER_NAME='oldusername';
update AUTHOR set AUTHOR_NAME='newusername' where
AUTHOR_NAME='oldusername';
update AUTH_ATTEMPT_INFO set USER_NAME='newusername' where
USER_NAME='oldusername';
update BUILDRESULTSUMMARY_CUSTOMDATA set CUSTOM_INFO_VALUE='newusername'
where CUSTOM_INFO_VALUE='oldusername' &&
CUSTOM_INFO_KEY='ManualBuildTriggerReason.userName';
update BUILDRESULTSUMMARY_LABEL set USER_NAME='newusername' where
USER_NAME='oldusername';
update LABEL set NAMESPACE='newusername' where NAMESPACE='oldusername';
update NOTIFICATIONS set RECIPIENT='newusername' where
RECIPIENT='oldusername';
update REMEMBERME_TOKEN set USERNAME='newusername' where
USERNAME='oldusername';
update USER_COMMENT set USER_NAME='newusername' where
USER_NAME='oldusername';
update external_entities set name='newusername' where
name='oldusername';
update users set name='newusername' where name='oldusername';
```

Managing groups

Bamboo *groups* are used to specify which [users](#) will have [global permissions](#) and [plan permissions](#). They can also be used to specify which users will receive [notifications about a plan's build results](#). You can create and delete as many groups as you need. You will typically create at least one group per [project](#).

A special group called **bamboo-admin** is automatically created when you install Bamboo. Members of this group have [Bamboo administration rights](#).

Read more about managing groups for your [users](#):

- [Creating a group](#)
- [Deleting a group](#)
- [Changing members of groups](#)

Creating a group

Bamboo *groups* are used to specify which [users](#) will have [global permissions](#) and [plan permissions](#). They can also be used to specify which users will receive [notifications about a plan's build results](#). You can create and delete as many groups as you need. You will typically create at least one group per [project](#).

A special group called **bamboo-admin** is automatically created when you install Bamboo. Members of this group have [Bamboo administration rights](#).

Related pages:

- [Managing groups](#)

To create a group:

1. Click **Administration** in the menu bar.
2. Click **Groups** (under 'Security') in the left navigation panel.
3. Type a name for your new group into **Group Name** (in the 'Create Group' section). Note that the group name cannot be changed after the group is created.

4. Select relevant users from the **Users to add** list. Hold <Ctrl> to select multiple users. You can also [add or remove users from the group](#) later if required.
5. Click **Save**.

Screenshot: Creating a Bamboo group

Create Group

Group Details

Enter the details of the group to add in Bamboo, then click Save.

Group Name: *

Users in Group:

admin

andreas

aro

ben.kuo

bmcocoy

booyah

christopher

cmiller

dave

don.willis

dushan

edwin

ernest

jason

jed

jens

jstepka

justin

mark

matt

Save

Deleting a group

Note that the *bamboo-admin* group cannot be deleted.

To delete a group:

1. Click **Administration** in the top menu bar.
2. Click **Groups** in the left navigation panel. The 'Manage Groups' screen will be displayed.
3. Click **Delete** for the relevant group, in the 'Operations' column.

Related pages:

- [Managing groups](#)

Screenshot: Deleting a group

Manage Group

You can use this page to view and delete groups.

Group	Number of Users	Operations
accounts-payable	3	Edit Delete
ams-sales	3	Edit
ams-staff	11	Edit

Changing members of groups

Bamboo *groups* are used to specify which [users](#) will have [global permissions](#) and [plan permissions](#). They can also be used to specify which users will receive [notifications about a plan's build results](#). You can create and delete as many groups as you need. You will typically create at least one group per [project](#).

A special group called **bamboo-admin** is automatically created when you install Bamboo. Members of this group have [Bamboo](#)

[administration rights](#).

To change the members of a group:

1. Click **Administration** in the top menu bar.
2. Click **Groups** in the left navigation panel. The 'Manage Groups' screen will be displayed.
3. Click **Edit** for the relevant group, in the 'Operations' column. The 'Edit Group Details' screen will be displayed. Users who already belong to the group are shown in blue; users who do not currently belong to the group are shown in white.
4. Press the <Ctrl> key and hold it while you select (or deselect) the users whom you want to add to (or remove from) the group.
5. Click **Save**.

Related pages:

- [Managing groups](#)

Screenshot: Adding users to a group

Group Details

Update the details of the group in Bamboo, then click Save.

Group Name: bamboo-admin

Users in Group:

admin

andreas

aro

ben

bmcocoy

booyah

christopher

cmiller

dave

don

dushan

edwin

ernest

jason

jed

jens

jstepka

justin

mark

matt

Save

Cancel

Connecting to external user directories

Bamboo provides a number of options for connecting to external user directories for user management:

- Manage users and groups in [Atlassian's JIRA](#) or [Atlassian's Crowd](#) — see [Integrating Bamboo with Crowd](#).
- Connect to a custom external user directory, such as an LDAP server — see [Integrating Bamboo with LDAP](#).

You can also manage [users](#) and [groups](#) within the Bamboo server itself.

Integrating Bamboo with Crowd

Atlassian's [Crowd identity management system](#) can be integrated with Bamboo. This allows you to use Crowd as a user directory manager for Bamboo.

The integration process requires you to configure Crowd to talk to Bamboo, then configure Bamboo to talk to Crowd. Hence, the instructions below reference the [Crowd documentation](#). Ensure that you are referring to the correct version of the Crowd documentation.

If you have **JIRA 4.3 or later**, you can also manage your users via JIRA. The process for connecting Bamboo to JIRA for user management is the same as the process for connecting Bamboo to Crowd for user management (described below).



Bamboo 3.2 should work with versions of Crowd from 2.1 onwards. We recommend **Crowd 2.3 or later** for performance reasons. Versions earlier than 2.1 are not supported.

On this page:

- [Step 1. Configuring Crowd to Talk to Bamboo](#)
- [Step 2. Configuring Bamboo to Talk to Crowd](#)
- [Notes](#)

Related pages:

- [Connecting to external user directories](#)
- [Integrating Crowd with Atlassian Bamboo \(Crowd documentation\)](#)

Step 1. Configuring Crowd to Talk to Bamboo

For instructions on how to configure Crowd to talk to Bamboo, please refer to the [Integrating Crowd with Atlassian Bamboo](#) for the latest version of Crowd, which can be found in the *Crowd Administrator's Guide*. If you are using an older version of Crowd, find the documentation from the [Crowd documentation homepage](#).

Step 2. Configuring Bamboo to Talk to Crowd

1. Navigate to **Administration > User Repositories** (under 'Security').
2. Choose **Users and groups from JIRA or Crowd** and configure the connection settings, as follows:

Server URL	Enter the URL of your Crowd server: If using Crowd 2.1 or older versions: http://localhost:8095/crowd/services/ If using Crowd 2.2 or newer versions: http://localhost:8095/crowd/ If your Crowd server's port is configured differently from the default (8095), set it accordingly.
Application Name	Enter the application name that you specified when configuring Crowd in Step 1 above.
Application Password	Select Change password and set the password that you specified when configuring Crowd in Step 1 above.
Cache Refresh Interval	Set to 0, if you want authentication checks to occur on each request. Otherwise set to the number of minutes between requests to validate if the user is logged in or out of the Crowd SSO server. Setting this value to 1 or higher will increase the performance of Crowd's integration.
Synchronise now	Click this link to synchronise users and groups from Crowd to Bamboo. Note, this operation may take a long time depending on the number of users that need to be synchronised.

3. Click **Save**

2.1 Configure External User Management in Bamboo

If you are connecting Bamboo to an external user management system and do not have rights to update user attributes there, you will need to prevent users from being updated in Bamboo. In this case, you should ensure that the **Read-only External User Management?** check-box is checked. For example, if Crowd directory permissions don't allow any remote changes, then Bamboo will give an error message if an attempt is made to change user account settings. We are tracking this bug:



BAM-12002 - Bamboo should allow changing the IM address of a user from Crowd ([➔](#) **Open**) .

To configure the external user management option in Bamboo:

1. Navigate to **Administration > Security Settings**.
2. Click **Edit**.
3. Select the **Read-only External User Management?** checkbox. The table below outlines the correct configuration for Bamboo, depending on your external user management setup:

External User Management Setup	Read-only External User Management? check-box
Bamboo integrated with — Crowd using the Crowd database (i.e. Internal Directories)	Unchecked
Bamboo integrated with — Crowd connected to a read-only LDAP	Checked
Bamboo integrated with — Crowd connected to a read-write LDAP	Unchecked
Bamboo integrated with — Crowd with authentication-only delegated to LDAP.	Unchecked

4. Click **Save**.

2.2 (Optional) Enable Single Sign-On

Single sign-on (SSO) is optional when integrating Bamboo and other Atlassian products with Crowd. To use centralised authentication *without* SSO, skip the steps below.

To enable single sign-on (SSO), you will configure Bamboo's authentication and access request calls to use Seraph. To configure Seraph-based authentication:

1. Shut down Bamboo.
2. Edit the `\BAMBOO\webapp\WEB-INF\classes\seraph-config.xml`
3. Comment out the authenticator node:

```
<!--<authenticator  
class="com.atlassian.bamboo.user.authentication.BambooAuthenticator"/>-->
```

4. Add a new authenticator, by adding the following tag:

```
<authenticator  
class="com.atlassian.crowd.integration.seraph.v25.BambooAuthenticator"/>
```

5. Start Bamboo. Bamboo's authentication and access request calls will now be performed using Seraph.

Notes

- Test times for synchronising Bamboo-Crowd — As a guideline, we were able to synchronise 5000 users in six seconds in our internal tests using Crowd 2.3.1. Older versions of Crowd took three minutes to complete the same task.
- If you want to configure the Bamboo-Crowd connection settings manually (e.g. to change the proxy settings), you can find the `crowd.properties` and `atlassian-user.xml` files in the `$BAMBOO_HOME/xml-data/configuration/` directory.

Integrating Bamboo with LDAP

Bamboo can be integrated with LDAP for the authentication and authorisation of LDAP users. The [Integrating Bamboo with LDAP](#) instructions below describe how you can set this up. Please note that Bamboo does not currently support multiple LDAP servers. If you need to connect to multiple LDAP servers, please consider using Atlassian's [Crowd](#).

If you choose to integrate Bamboo with LDAP, please note that you cannot manage LDAP accounts or user groups from Bamboo. Please refer to the [Integrating Bamboo with LDAP](#) instructions on this page for details on how to manage your users and groups.

 In Bamboo version 1.2.2 and later, XML backups of your Bamboo instance do not include LDAP user data.

 Please note that once LDAP has been enabled, reverting back to local user management is not possible.

On this page:

- [Integrating Bamboo with LDAP](#)
- [External User Management](#)
- [Notes](#)

Related pages:

- [Connecting to external user directories](#)

Integrating Bamboo with LDAP

Step 1 — Prerequisites

You will need to confirm that your LDAP server is compatible and set up correctly before integrating it with Bamboo. Please check your LDAP server against the requirements below:

- **Check your LDAP server version** — Supported versions are v2 and v3. Supported LDAP servers include OpenLDAP, Microsoft Active Directory, Novell eDirectory, and any server that uses Java JNDI-LDAP mapping.
- **Check whether your LDAP or Active Directory server supports static groups** — Your LDAP or Active Directory server must support static groups. This means that the *user DNs* must be stored against a membership attribute inside an LDAP group. An example of a static group is shown below:

```
Dn: CN=Sales and Marketing,CN=Users,DC=ad,DC=atlassian,DC=com
objectClass: top; group;
cn: Sales and Marketing;
distinguishedName: CN=Sales and
Marketing,CN=Users,DC=ad,DC=atlassian,DC=com;
name: Sales and Marketing;
...
member: CN=John Smith,CN=Users,DC=ad,DC=atlassian,DC=com
member: CN=Sally Smith,CN=Users,DC=ad,DC=atlassian,DC=com
...
```

The membership attribute in this case is **member**, but this is not required. Note that the full DNs of *John Smith* and *Sally Smith* are listed. If the values against **member** are not full DNs, but are just usernames, then you need to add the flag `<useUnqualifiedUsernameForMembershipComparison>true</useUnqualifiedUsernameForMembershipComparison>` to your LDAP configuration. Open Directory on OS X uses this configuration.

- **Ensure that you do not have an LDAP group called 'bamboo-admin'.** — The bamboo-admin group is reserved by Bamboo.
- **Ensure that you do not have duplicate users in your LDAP directory** — If you have users in your LDAP directory that are also on Bamboo, the first repository definition in your `atlassian-user.xml` file will take precedence.
- **Ensure that you do not have duplicate groups in your LDAP directory** — If you have groups in your LDAP directory that are also on Bamboo, this may cause unpredictable behaviour when you attempt to integrate your LDAP server with Bamboo.

Step 2 — Backup your data

We strongly recommend that you [backup your data](#) before attempting LDAP integration.

Step 3 — Configure Connection Details

The LDAP server connection is specified by manually editing the file `atlassian-user.xml`.

To configure your connection details:*

1. Edit the file `.../{BAMBOO-HOME}/xml-data/configuration/atlassian-user.xml` and configure the connection AD or LDAP.
2. Check your configuration against the example connection details shown below.

```
<ldap key="ldapRepository" name="LDAP Repository@hecate.atlassian.com"
cache="true">
  <host>hecate.atlassian.com</host>
  <port>389</port>
  <securityPrincipal>cn=admin,dc=atlassian,dc=private</securityPrincipal>
  <securityCredential>secret</securityCredential>
  <securityProtocol>plain</securityProtocol>
  <securityAuthentication>simple</securityAuthentication>
  <baseContext>dc=atlassian,dc=private</baseContext>
...

```

3. Please ensure that the following line is also active in your `atlassian-user.xml` (it should be there by default):

```
<hibernate name="Hibernate Repository" key="hibernateRepository"
description="Hibernate Repository" />
```

Step 4 — Map LDAP Data Tree

To map the LDAP Data Tree:

1. To configure the mappings in `atlassian-user.xml` for either AD or LDAP, please see:
 - [Mapping Active Directory](#)
 - [Mapping other LDAP servers](#)
2. Check your configuration against the example connection details shown below.

...

```
<baseUserNamespace>dc=staff,dc=perftest,dc=atlassian,dc=private</baseUserNamespace>

<baseGroupNamespace>dc=groups,dc=perftest,dc=atlassian,dc=private</baseGroupNamespace>
  <usernameAttribute>cn</usernameAttribute>
  <userSearchFilter>(objectClass=inetorgperson)</userSearchFilter>
  <firstnameAttribute>givenname</firstnameAttribute>
  <surnameAttribute>sn</surnameAttribute>
  <emailAttribute>mail</emailAttribute>
  <groupnameAttribute>cn</groupnameAttribute>
  <groupSearchFilter>(objectClass=groupOfNames)</groupSearchFilter>
  <membershipAttribute>member</membershipAttribute>
</ldap>
```

Filters: <userSearchFilter> and <groupSearchFilter> may use the AD specific filter syntax. Operators such as & need to be escaped.

```
<userSearchFilter>(&(objectClass=inetorgperson)(memberOf=cn=bamboo-usr,ou=groups,DC=atlassian,dc=com))</userSearchFilter>
```



Making Sure that the LDAP Filters are Precise

Depending on the LDAP server being used, different object types may have common objectClass values. In this case, please customize the User and Group filters so that Bamboo can fetch only the objects that are really useful for the application. If your directory server does not display the literal object LDIF details, you may want to use an LDAP browser tool like [Apache Directory Studio](#) to check if the filters are restricting the objects correctly based on one or more object attributes.

Step 5 — Optional LDAP Settings

The following settings do not appear in the default atlassian-user.xml file. Their default values are as follows:

```
<poolingOn>true</poolingOn>
<maxSize>0</maxSize>
<initSize>10</initSize>
<prefSize>10</prefSize>
<debugLevel>none</debugLevel>
<securityProtocol>plain ssl</securityProtocol>
<authentication>simple</authentication>
<timeout>0</timeout>
<initialContextFactory>com.sun.jndi.ldap.LdapCtxFactory</initialContextFactory>
<batchSize>100</batchSize>
<timeToLive>0</timeToLive>
<userSearchAllDepths>true</userSearchAllDepths>
<groupSearchAllDepths>true</groupSearchAllDepths>
```

If you want to override these default values, you can specify any or all of them by adding them to the end of the atlassian-user.xml file. For example, to add your own value for the <initSize> setting, you would add an extra line before the </ldap> line shown in 'Stage 3' above:

```
...
<groupnameAttribute>cn</groupnameAttribute>
<groupSearchFilter>(objectClass=groupOfNames)</groupSearchFilter>
<membershipAttribute>member</membershipAttribute>
<initSize>20</initSize>
</ldap>
```

It is important that the connection pool timeout value be set to 0, as this will force Atlassian User (via the JNDI layer) to clean up lingering connections that have lived past one request. For more information about LDAP pools please see <http://java.sun.com/products/jndi/tutorial/ldap/connect/config.html>.

External User Management

You cannot manage LDAP accounts or user groups from Bamboo. Bamboo will continue to use local users and groups, even when LDAP is enabled. If you need to assign LDAP users to particular groups referenced by Bamboo (e.g. for permissions or notifications), the workaround is to assign your LDAP users to local Bamboo groups, and reference these groups rather than LDAP groups.

 Please note that once LDAP has been enabled, *reverting back to local user management is not possible*.

1. Navigate to **Administration > User Repositories** (under 'Security').
2. Choose **Custom user repository**.
3. Click **Save**.

Step 1 — Configuring Bamboo for External User Management

Please ensure that **Read-only External User Management** is turned **OFF** in Bamboo before assigning LDAP users to Bamboo groups. The **Read-only External User Management** check-box in Bamboo controls whether users and groups in Bamboo are editable. Setting this option to **ON** will make users and groups in Bamboo read-only (i.e. the implication being that you will be managing your users and groups externally).

 If you are using Crowd together with an LDAP, please read the documentation on [Integrating Bamboo with Crowd](#) to see our recommended settings for the **Read-only External User Management** check-box.

To disable Read-only External User Management:

1. Click **Administration** in the top menu bar.
2. Click **Security Settings** (under 'Security') in the left navigation panel.
3. Clear the **Read-only External User Management** check-box.
4. Click **Save**.

Step 2 — Assigning LDAP Users to Bamboo Groups

Once Bamboo is started with '**Read-Only External User Management**' disabled, you can assign LDAP users to Bamboo groups. Please see [Changing members of groups](#).

 Please note, the 'View Users' and 'View Groups' screens in Bamboo currently will not list all of your LDAP users/groups (please see [BAM-1963](#) for details).

Notes

- To check whether the `atlassian-user.xml` file is correctly configured, please run the Paddle tool to debug the LDAP configuration in your `atlassian-user.xml` file. For further reference, please visit the [Paddle usage](#) page.

Configuring the caching of your LDAP repository

The instructions on this page describe how to configure the caching of your LDAP repository.

Controlling the caching of users

By default, caching is activated for your LDAP users. We recommend that you do not disable caching of your LDAP users, as your LDAP repository may be overloaded by the high volume of requests by Bamboo.

To control the caching of users:

1. Click **Administration** in the top navigation bar.
2. Edit the file `.../webapp/WEB-INF/classes/atlassian-user.xml`
3. To disable caching, set the property `cache="false"` on your LDAP repository, as shown in the example below:


```
<ldap key="myLdapRepository" name="LDAP Repository@hecate.atlassian.com"
cache="false">
    <host>hecate.atlassian.com</host>
    <port>389</port>
```

4. To enable caching, set the property `cache="true"` on your LDAP repository as shown in the example below:

```
<ldap key="myLdapRepository" name="LDAP Repository@hecate.atlassian.com"
cache="true">
    <host>hecate.atlassian.com</host>
    <port>389</port>
```

On this page:

- [Controlling the caching of users](#)
- [Configuring the LDAP caches](#)

Related pages:

- [Integrating Bamboo with LDAP](#)
- [Testing LDAP or Active Directory connectivity with Paddle](#)

Configuring the LDAP caches

Bamboo uses a number of caches for managing an LDAP repository, each of which can be configured differently. You must enable caching, as described above, before configuring the caches. The caches used by Bamboo are:

- [Configuring caches for users](#)
- [Configuring caches for groups](#)

Each cache can be configured by following the instructions below:

To configure a cache:

1. Edit the file `.../webapp/WEB-INF/classes/ehcache.xml`.
2. Find the cache that you wish to edit. Examples of each of the caches are described in the [Configuring Caches for Users](#) and [Configuring Caches for User Groups](#) sections below.
3. Modify the cache, as desired. The following properties can be configured for each cache:
 - **maxElementsInMemory** (mandatory) - Sets the maximum number of objects that will be created in memory
 - **eternal** (mandatory) - Sets whether elements are eternal. If eternal, timeouts are ignored and the element is never expired.
 - **timeToIdleSeconds** (optional) - Sets the time to idle for an element before it expires. i.e. The maximum amount of time between accesses before an element expires. This is only used if the element is not eternal. A value of 0 means that an Element can idle for infinity. The default value is 0.
 - **timeToLiveSeconds** (optional) - Sets the time to live for an element before it expires i.e. The maximum time between creation time and when an element expires. This is only used if the element is not eternal. A value of 0 means that an Element can live for infinity. The default value is 0.



If you have caching turned on Bamboo will, by default, set the cache to eternal (elements will never expire), and set the maximum number of elements stored to 500. These can be configured to speed up user retrieval, reduce memory usage or reduce the load on the LDAP repository.

Configuring caches for users



In each of the examples below, replace `myLdapRepository` with the key of the repository specified in `atlassian-user.xml`

- `LDAPUserManagerReadOnly.*.users`

LDAPUserManagerReadOnly.*.users stores the individual users, if you have difficulties with Bamboo picking up new user additions in the LDAP repository you will need to alter the configuration of this cache. In the example below, the users will expire after 5 minutes.

```
<cache
name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.us
ers"
    maxElementsInMemory="500"
    eternal="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
/>
```

- LDAPUserManagerReadOnly.*.users_ro

LDAPUserManagerReadOnly.*.users_ro stores whether or not the users are read only. This will have no effect on the functionality of Bamboo, but you may wish to modify this cache for performance and memory tuning purposes.

```
<cache
name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.us
ers_ro"
    maxElementsInMemory="500"
    eternal="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
/>
```

- LDAPUserManagerReadOnly.*.repository

LDAPUserManagerReadOnly.*.repository stores which repository the user belongs to. Bamboo does not yet support multiple repositories, so modifying this cache will have no effect on functionality. However, you may wish to modify this cache for performance and memory tuning purposes.

```
<cache
name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.re
pository"
    maxElementsInMemory="500"
    eternal="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
/>
```

Configuring caches for groups

 In each of the examples below, replace myLdapRepository with the key of the repository specified in atlassian-user.xml

- LDAPGroupManagerReadOnly.*.groups

LDAPGroupManagerReadOnly.*.groups stores the available groups in LDAP. If you wish Bamboo to pick up changes made to groups, then you will need to configure this cache appropriately.

```
<cache
name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.g
roups"
    maxElementsInMemory="500"
    eternal="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
/>
```

- LDAPGroupManagerReadOnly.*.groups_hasMembership* and *LDAPGroupManagerReadOnly.*.groups_getGroupsForUser

The **LDAPGroupManagerReadOnly.*.groups_hasMembership** and **LDAPGroupManagerReadOnly.*.groups_getGroupsForUser**

er caches store the associations between users and groups. If you wish Bamboo to pick up changes made to group memberships then you will need to configure these caches appropriately.

```
<cache
name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.g
roups_hasMembership"
    maxElementsInMemory="500"
    eternal="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
/>
<cache
name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.g
roups_getGroupsForUser"
    maxElementsInMemory="500"
    eternal="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
/>
```

- `LDAPGroupManagerReadOnly.*.repositories`

LDAPGroupManagerReadOnly.*.repositories stores which repository the group belongs to. Bamboo does not yet support multiple repositories, so modifying this cache will have no effect on functionality. However, you may wish to modify this cache for performance and memory tuning purposes.

```
<cache
name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.r
epositories"
    maxElementsInMemory="500"
    eternal="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
/>
```

Testing LDAP or Active Directory connectivity with Paddle

Paddle is a tool that will test the LDAP or Active Directory settings in your `atlassian-user.xml`.

Using Paddle

You do not need to have Bamboo running to run this tool. The steps are:

1. [Download](#) into a directory where you have permissions to create files.
2. Copy your `atlassian-user.xml` into that directory - this is found in your `.../{BAMBOO-HOME}/xml-data/configuration/` directory.
3. Run `java -jar paddle-x.x.jar` (where `x.x` is the version of Paddle you downloaded).

On this page:

- [Using Paddle](#)
- [Parameters](#)
- [Sample output](#)
- [Notes](#)

Parameters

Paddle currently supports the following parameters:

Name	Example	Purpose
debug	<code>java -jar paddle-x.x.jar debug</code>	Prints DEBUG messages to the console as well as <code>paddle.log</code> .
limit	<code>java -jar paddle-x.x.jar limit=100</code>	Sets the limit on the number of results returned by user and group queries. Defaults to 10.

Sample output

This is an example of a successful run:

```
#####
#####

LDAP Support Tool version 1.1

#####
#####

Connection to LDAP/Active Directory Server at ldap://192.168.0.86:389 SUCCESSFUL.

-----
TEST 1: Search and list 10 users
-----

User: CN=Administrator
Member of:
(1) CN=Schema Admins
(2) CN=Enterprise Admins
(3) CN=Domain Admins
(4) CN=Group Policy Creator Owners

User: CN=Guest
Does not belong to any LDAP groups.

User: CN=SUPPORT_388945a0
Member of:
(1) CN=HelpServicesGroup

User: CN=IUSR_MALTSHOVEL
Does not belong to any LDAP groups.

User: CN=IWAM_MALTSHOVEL
Member of:
(1) CN=IIS_WPG

User: CN=ASPNET
Does not belong to any LDAP groups.

User: CN=krbtgt
Does not belong to any LDAP groups.

User: CN=John\, Smith
Member of:
(1) CN=Domain Users
(2) CN=Sales and Marketing

User: CN=Matt Ryall
Member of:
(1) CN=Enterprise Admins
(2) CN=Domain Admins

User: CN=Justin Koke
Member of:
(1) CN=Domain Controllers
(2) CN=Enterprise Admins

Found more than 10 results.

-----
TEST 2: Search and list 10 groups
-----
```

Group: CN=HelpServicesGroup

Members:

- (1) CN=SUPPORT_388945a0,CN=Users,DC=ad,DC=atlassian,DC=com

Group: CN=TelnetClients

No members in this group.

Group: CN=IIS_WPG

Members:

- (1) CN=S-1-5-20,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com
- (2) CN=S-1-5-6,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com
- (3) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com
- (4) CN=IWAM_MALTSHOVEL,CN=Users,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005SQLBrowserUser\$MALTSHOVEL

Members:

- (1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005MSSQLServerADHelperUser\$MALTSHOVEL

Members:

- (1) CN=S-1-5-20,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005SQLAgentUser\$MALTSHOVEL\$MSSQLSERVER

Members:

- (1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005MSSQLUser\$MALTSHOVEL\$MSSQLSERVER

Members:

- (1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005MSFTEUser\$MALTSHOVEL\$MSSQLSERVER

Members:

- (1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005MSOLAPUser\$MALTSHOVEL\$MSSQLSERVER

Members:

- (1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005NotificationServicesUser\$MALTSHOVEL

No members in this group.

Found more than 10 results.

Notes

Related Topics

[Integrating Bamboo with LDAP](#)

Managing permissions

You can grant global permissions so as to control which users and groups have access to build plans and the Bamboo server, and the actions they can perform.

Common global permissions tasks are:

- [Granting plan permissions in bulk](#) — control the users and groups that can perform actions on plans (e.g. edit, build, clone).
- [Granting global permissions to users or groups](#) — control the users and groups that can create plans, delete plans, and administer Bamboo.
- [Allowing anonymous access to Bamboo](#) — allow people not logged in to Bamboo to generate reports, and view plans and build results.

Read about managing permissions for [users](#) and [groups](#)

You can also change the permissions for an individual plan: see [Configuring a plan's permissions](#).

Granting plan permissions in bulk

A [plan permission](#) is the ability to perform a particular operation on a [plan](#) and its [jobs](#). For each plan, different permissions can be granted to particular groups and/or users.

- People who have the 'Admin' global permission can 'bulk edit' permissions for multiple plans at the same time, as described below. Note that this will overwrite any pre-existing plan permissions.
- People who have the 'Admin' plan permission for one or more plans, but do not have the 'Admin' global permission, can only edit one plan at a time, as described in [Configuring a plan's permissions](#).

Note that it is recommended that you grant permissions to groups rather than to individual users.

To grant bulk plan permissions to a user or group:

1. Click **Administration** in the top navigation bar.
2. In the **Plans** section of left navigation panel, click **Bulk Edit Plan Permissions**.
3. Select the plans whose permissions you wish to edit, then click **Next** (at the bottom of the screen).
4. You can set plan permissions for the categories of users in the table below.
5. Select the check box for each permission that you wish to grant to the user or group.
6. Click **Save**.

User category	Description
Logged in Users	Users who are logged in to Bamboo.
Anonymous Users	Users who are not logged in to Bamboo.
User	A user already created in the Bamboo system. To edit plan permissions for an existing user: 1. In the Grant permission to list, select User. 2. Type the username into the box, or click the icon to select from a list. 3. Click Add. The user will be added to the list on the screen, and you can then select permissions for them.
Group	A group already created in the Bamboo system. To edit plan permissions for an existing group: 1. In the Grant permission to list, select Group. 2. Type the group name into the box, or click the icon to select from a list. 3. Click Add. The group will be added to the list on the screen, and you can then select permissions for the group.

Screenshot: *Bulk Edit Plan Permissions Wizard*

Bulk Edit Plan Permissions Wizard

Users	View	Edit	Build	Clone	Admin
Paul Watson	☒	☒	☒	☒	☒

Groups	View	Edit	Build	Clone	Admin
atlassian-docs	☒	☐	☐	☐	☐

Other	View	Edit	Build	Clone	Admin
Logged in Users	☒	☐	☐	☐	☐
Anonymous Users	☒				

Grant permission to

User



Add

Save Cancel

Granting global permissions to users or groups

Global permissions control which users and groups have access to build plans and the Bamboo server, and what actions they can perform.

Note that if you remove *all* permissions for a user or group, that user or group will disappear from the **Permissions** tab for all plans.

Related pages:

- [Managing permissions](#)
- [Configuring a plan's permissions](#)
- [Creating a user](#)
- [Creating a group](#)

To change global permissions:

1. Click **Administration** in the top navigation bar.
2. Click **Global Permissions** in the left navigation panel, and then **Edit Global Permissions**.
3. You can set plan permissions for the categories of users in the table below.
4. Select (or clear) the check box for each permission that you wish to change for a user or group.
5. Click **Save**.

User category	Description
Logged in Users	Users who are logged in to Bamboo.

Anonymous Users	Users who are not logged in to Bamboo.
User	A user already created in the Bamboo system. To edit plan permissions for an existing user: 1. In the Grant permission to list, select User. 2. Type the username into the box, or click the icon to select from a list. 3. Click Add. The user will be added to the list on the screen, and you can then select permissions for them.
Group	A group already created in the Bamboo system. To edit plan permissions for an existing group: 1. In the Grant permission to list, select Group. 2. Type the group name into the box, or click the icon to select from a list. 3. Click Add. The group will be added to the list on the screen, and you can then select permissions for the group.

You can grant the following global permissions:

Global permission	Description	Can be granted to
Access	Permission to view the Bamboo system. ⚠ The ability to view build plans and build results is subject to individual plan permissions.	• a particular user • a particular group • all logged-in users • anonymous users
Create Plan	Permission to create new build plans.	• a particular user • a particular group • all logged-in users
Admin	Permission to: • access the Bamboo Administration menu • create plans • delete plans The 'Admin' global permission also includes all plan permissions, for every plan.	• a particular user • a particular group

Screenshot: *Global Permissions*

Global Permissions

You can edit your global application level permissions here. Permissions can be granted to specific users or groups. Please note these are global application permissions. For Plan level permissions, please go to the Plan configuration page.

Users	Access	Create Plan	Admin
luke	✓	✓	✗

Groups	Access	Create Plan	Admin
bamboo-admin	✓	✓	✓

Other	Access	Create Plan
All logged in users	✓	✗
Anonymous users	✓	

[Edit Global Permissions](#)

Allowing anonymous access to Bamboo

Allowing *anonymous users* to access your Bamboo system means that people who aren't logged in to Bamboo will be able to perform functions such as generating reports, and viewing plans and build results — subject to individual [plan permissions](#).

Note that people who aren't logged in to Bamboo do not have a 'My Bamboo' tab on their [Dashboard](#).

See [Managing anonymous access](#).

To allow anonymous users to access Bamboo:

1. Click **Administration** in the menu bar.
2. Click **Global Permissions** (under 'Security'), and then **Edit Global Permissions**.
3. Select the **Access** check box for 'Anonymous users'.
4. Click **Save**.

i Anonymous users will now be able to access your Bamboo system. However, they will only be able to view plans and build results for plans where the 'Access' [plan permission](#) has been granted to 'Anonymous users'.

Screenshot: *Global Permissions*

Groups	Access	Create Plan	Admin
bamboo-admin	☒	☒	☒

Other	Access	Create Plan	Admin
All logged in users	☐	☐	
Anonymous users	☐		

Grant permission to: User Add

Save Cancel

Global security and permission properties

Global security and permission properties allow a Bamboo system administrator to configure security- and permission-related properties that apply to Bamboo at a site-wide level.

Read more about configuring Bamboo's global security and permission properties:

- [Allowing public signup](#)
- [Displaying full details about users](#)
- [Using Captcha for failed logins](#)

Note that the *Restricted Administrator Role* global permission is used in [Atlassian OnDemand](#) and grants access to 'Builds' administration: permission to delete plans and access to plan permissions for every plan - but not to administer Bamboo.

The *Restricted Administrator Role* global permission is equivalent to the [Admin global permission](#) but has the following restrictions:

- Local agents cannot be created because Atlassian OnDemand only uses [elastic agents](#).
- Users and groups are managed by Atlassian OnDemand.

Allowing public signup

If you enable *signup* for your Bamboo system, visitors can create their own Bamboo user accounts. Public signup is enabled on your Bamboo site if you see the 'Signup' link at the top-right of the Bamboo user interface.

To enable (or disable) signup:

1. Click **Administration** in the top menu bar.
2. Click **Security Settings** (under 'Security') in the left navigation panel to open the 'Global Security and Permission Properties' page.
3. Click **Edit** on this page.
4. Select, (or clear) the **Enable Signup?** check box.
5. Select **Enable Captcha On Signup** if you require an additional security measure to prevent brute force attacks.
6. Click **Save**.
7. Log out of Bamboo and verify that the top menu bar now contains (or does not contain) a **Signup** link.

Related pages:

- [Global security and permission properties](#)
- [Using Captcha for failed logins](#)

Screenshot: Security settings for Bamboo, including signup

Security and Permission

You can change the following security and permission related settings for Bamboo.

Change Global Security and Permission Properties

☒ Read-only External User Management?

Enable this option if you are connecting Bamboo to an external user management system and do not have update rights there.

☐ Enable Signup?

This will allow users to sign up for an account to Bamboo.

☐ Enable contact details to be displayed?

This will allow Bamboo user's contact details to be visible. Disabling this option will hide the email address, IM address, and the group the user is in.

☐ Enable Restricted Administrator Role

This will enable the Restricted Administrator Role

☒ Enable Brute Force Protection

Forces the user to enter a captcha code if they meet the maximum amount of failed login attempts

Login Attempts *

Number of login attempts before Captcha is shown

Save

Cancel

Displaying full details about users

If you enable the display of contact details on your Bamboo system, the full contact details for all users, including email address, IM address, and group membership, will be visible to any visitors to Bamboo. The email addresses of administrators on the 'Contact Administrators' page will also be visible.

To enable (or disable) the display of contact details:

1. Click **Administration** in the top menu bar.
2. Click **Security Settings** (under 'Security') in the left navigation panel to open the 'Global Security and Permission Properties' page.
3. Click **Edit** on this page.
4. Select (or clear) the **Enable contact details to be displayed?** check box.
5. Click **Save**.

Related pages:

- [Global security and permission properties](#)

Using Captcha for failed logins

Captcha is a tool that prevents brute force attacks on the Bamboo login screen. A brute force attack occurs when an attacker uses malicious code to make automated, repeated login attempts on a Bamboo site with the aim of gaining access to that Bamboo site.

A Bamboo system administrator can configure Bamboo to block automated login attempts. Once a certain number of failed login attempts has been reached (the default is three) Bamboo's Captcha feature will be activated. When Captcha is activated, users will need to recognise a distorted picture of a word and must type the word into a text field. This is easy for humans to do, but very difficult for computers.

Related pages:

- [Global security and permission properties](#)

To enable (or disable) Captcha for Bamboo:

1. Click **Administration** in the top menu bar.
2. Click **Security Settings** (under 'Security') in the left navigation panel to open the 'Global Security and Permission Properties' page.

3. Click **Edit** on this page.
4. Select (or clear) the **Enable Captcha** check box.
5. If required, specify the number of failed login attempts permitted by Bamboo before Captcha is activated. (This field is mandatory and requires a value of 1 or more.)
6. Click **Save**.

Screenshot: The Bamboo login screen with Captcha activated

Connecting Bamboo to an external database



Before you begin

Please note: if you are already using Bamboo with the embedded HSQL database (or any other database), and you want to keep your data, please see [Moving your Bamboo data to a different database](#).

Bamboo can be connected to an external database. For details and instructions please see:

- [PostgreSQL 8.2](#)
- [MySQL 5.1](#)
 - [Tomcat and External MySQL Datasource Example](#)
- [Oracle 11g](#)
- [Microsoft SQL Server 2005 and 2008](#)
- [How do I connect Bamboo to an unsupported database](#)
- [Troubleshooting Databases](#)

PostgreSQL 8.2

These instructions will help you connect Bamboo to a [PostgreSQL 8.2+](#) database. (PostgreSQL 8.0 and PostgreSQL 8.1 are not supported for use with Bamboo.)



Please note, the JDBC driver for PostgreSQL 8.2 (PostgreSQL Driver 8.4.x) is bundled with Bamboo. You do not have to download and install the driver.

On this page:

1. [Configuring PostgreSQL](#)
2. [Connecting Bamboo to PostgreSQL](#)

Related pages:

[Troubleshooting Databases](#)

1. Configuring PostgreSQL

Accept remote TCP connections (remote PostgreSQL server only)

If you are connecting Bamboo to a remote PostgreSQL server (i.e. if your PostgreSQL server is not installed locally on your Bamboo server

host system), you will need to configure your `data/postgresql.conf` and `data/pg_hba.conf` files to accept remote TCP connections from your Bamboo server's IP address.

The following PostgreSQL documentation contains information on the appropriate `listen_addresses` value in the `postgresql.conf` file as well as the `pg_hba.conf` file:

- [PostgreSQL 8.2 documentation — Connections and Authentication](#)

Once you have modified your `data/postgresql.conf` and `data/pg_hba.conf` files, you will need to restart PostgreSQL for your changes to take effect.

Creating a Bamboo database

```
sudo -s -H -u postgres
# Create the Bamboo user:
/opt/PostgreSQL/8.3/bin/createuser -S -d -r -P -E bamboouser
# Create the bamboo database:
/opt/PostgreSQL/8.3/bin/createdb -O bamboouser bamboo
exit
```



Creating a completely empty Bamboo database is recommended. Avoid using templates to create the database as some may insert default tables which can lead to conflicts when setting up Bamboo.

2. Connecting Bamboo to PostgreSQL

Bamboo provides two ways to connect to a PostgreSQL database — using JDBC or using a datasource. JDBC is generally simpler and is the recommended method.

Run the Setup wizard

For both methods, run the [Setup Wizard](#) and choose the **Custom Installation** option.

On the 'Choose a Database Configuration' page, choose **External Database**, select **PostgreSQL 8.2 and above** from the list and click **Continue**.

Choose one of the following.

Connecting using JDBC

On the 'Database Configuration' page of the Setup Wizard, ensure that **Direct JDBC connection** has been selected and make the following settings:

Setting	Description
Driver Class Name	Type <code>org.postgresql.Driver</code> (if different from the default).
Driver Class Name	Type the URL where Bamboo will access your database (if different from the default). For details about syntax, please refer to the Postgres JDBC driver documentation .
User Name	Type the username that Bamboo will use to access your database.
Password	Type the password (if required) that Bamboo will use to access your database.
Overwrite existing data	Select if you wish Bamboo to overwrite any tables that already exist in the database.

Screenshot 1: Setup JDBC Connection (PostgreSQL)

Database Configuration

Choose how you wish Bamboo to connect to your database

Select Database Connection

- ☒ Direct JDBC connection
☐ Connect via a datasource (configured in the application server)

Driver Class Name:

The class name of the database driver. Ensure that this class is in your class path.

Database URL:

The url to access the database.

User Name:

The username to access the database.

Password:

(optional) Enter the password if the database configuration requires it.

☐ Overwrite existing data

If you wish Bamboo to overwrite any existing tables that may exist in the database.

Continue

Connecting with a datasource

Configure a datasource in your application server (consult your application server documentation for details).

 For details about the syntax to use for the JDBC database URL, please see the [Postgres JDBC driver documentation](#).

On the 'Database Configuration' page of the Setup Wizard, choose **Connect via a datasource (configured in the application server)** and make the following settings:

Setting	Description
JNDI name	Type the JNDI name of your datasource, as configured in your application server.  If <code>java:comp/env/jdbc/DatasourceName</code> does not work, try <code>jdbc/DatasourceName</code> (and vice versa).
Overwrite existing data	Select if you wish Bamboo to overwrite any tables that already exist in the database.

Screenshot 2: Setup Datasource Connection

Database Configuration

Choose how you wish Bamboo to connect to your database

Select Database Connection

- ☐ Direct JDBC connection
☒ Connect via a datasource (configured in the application server)

 If `'java:comp/env/jdbc/DatasourceName'` doesn't work, try `'jdbc/DatasourceName'` (or vice versa)

JNDI name:

☐ Overwrite existing data

If you wish Bamboo to overwrite any existing tables that may exist in the database.

Continue

MySQL 5.1

These instructions will help you connect Bamboo to a [MySQL 5.1](#) database.
(Neither MySQL 4.1 nor 5.0 are supported databases for use with Bamboo.)

 Please note that the JDBC driver for MySQL 5.1 (JDBC Connector/J 5.1) is **no longer** bundled with Bamboo. You **must** [download](#) and

install the driver yourself.

On this page:

1. Creating and Configuring the MySQL database
2. Connecting Bamboo to the MySQL database

Related pages:

[Troubleshooting Databases](#)

1. Creating and Configuring the MySQL database

To connect Bamboo to an external MySQL database, you must first create and configure it. This database must be configured to use:

- utf8 character set encoding, instead of latin1
- utf8_bin collation
- the InnoDB storage engine

If your MySQL database server is configured to use a storage engine other than InnoDB by default (such as MyISAM), then if possible change it to use InnoDB. Otherwise, you can [configure Bamboo's JDBC connection](#) to your MySQL database so that any tables which Bamboo creates in this database will be done using the InnoDB database engine.

A MySQL database administrator can easily create and configure a MySQL database for Bamboo by running the following MySQL commands:

```
mysql> CREATE DATABASE bamboo CHARACTER SET utf8 COLLATE utf8_bin;
mysql> GRANT ALL PRIVILEGES ON bamboo.* TO 'bamboouser'@'localhost' IDENTIFIED BY
'password';
mysql> FLUSH PRIVILEGES;
mysql> QUIT
```

This creates an empty MySQL database for Bamboo named bamboo.



Please Note:

- bamboouser — the user account name for the Bamboo MySQL database
- localhost — the host name of the MySQL database server
- password — the password for this user account

For more information about configuring character set encoding and collation for Bamboo MySQL databases, please refer to the [MySQL 5 documentation — Specifying Character Sets and Collations](#).

2. Connecting Bamboo to the MySQL database

Bamboo provides two ways to connect to a MySQL database — by using either JDBC or a datasource. JDBC is generally simpler and is the recommended method.

Connect using JDBC

1. Download and install the JDBC driver:

The JDBC drivers for MySQL Enterprise Server are *no longer* bundled with Bamboo (due to licensing restrictions). You need to download and install the driver yourself.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).
2. Expand the downloaded zip/tar.gz file.
3. Copy the mysql-connector-java-5.1.XX-bin.jar file from the extracted directory to the <Bamboo installation directory>/lib directory (create the lib/ directory if it doesn't already exist). If you are using the Java Service Wrapper to start your Bamboo instance (Bamboo/wrapper/run-bamboo start), copy the mysql-connector-java-5.1.XX-bin.jar file to <Bamboo installation directory>/wrapper/lib directory.
4. Stop Bamboo, on [Windows](#), [Linux](#) or [Mac](#).
5. Restart Bamboo, on [Windows](#), [Linux](#) or [Mac](#).

2. Connect Bamboo to a MySQL database using JDBC:

1. Run the [Setup Wizard](#) and choose the **Custom Installation** method.
2. On the 'Choose a Database Configuration' page, choose **External Database > MySQL 5.1** and click **Continue**. The 'Database Configuration' page will appear.
3. Ensure that **Direct JDBC connection** is selected and complete the following fields (as shown in the screenshot below):

Driver Class Name	Type <code>com.mysql.jdbc.Driver</code> (if different from the default).
Database URL	Type the URL where Bamboo will access your database (if different from the default). Your URL <i>must</i> include the <code>autoReconnect=true</code> flag. - If you intend to use non-Latin characters in Bamboo, ensure that your URL includes the <code>useUnicode=true</code> and <code>characterEncoding=utf8</code> flags. - If your MySQL database server is configured to use a storage engine other than InnoDB by default, ensure that your URL includes the <code>sessionVariables=storage_engine=InnoDB</code> flag. If you include all of these flags, your Database URL should look similar to: <pre>jdbc:mysql://localhost/bamboo?autoReconnect=true&useUnicode=true&characterEncoding=utf8&sessionVariables=storage_engine=InnoDB</pre>  If the <code>autoReconnect=true</code> flag is not specified, the MySQL JDBC driver will eventually time out and Bamboo will no longer be able to communicate with the database. For more information on the URL syntax, please see the MySQL documentation.
User Name	Type the username that Bamboo will use to access your database. This is <code>bamboouser</code> defined in section 1 (above).
Password	Type the password (if required) that Bamboo will use to access your database. This is <code>password</code> defined in section 1 (above). Leave this field blank if a password for the database user account was not specified.

4. Select **Overwrite existing data** if you wish Bamboo to overwrite any tables that already exist in the database.
5. Click **Continue**.

Screenshot: Setup JDBC Connection (MySQL)

Database Configuration

Choose how you wish Bamboo to connect to your database

Select Database Connection

☒ Direct JDBC connection
 ☐ Connect via a datasource (configured in the application server)

 If you are using MySQL, the JDBC URL for the connection *must* have `'autoReconnect=true'` set. For example:

```
jdbc:mysql://localhost/bamboo?autoReconnect=true
```

If the `autoreconnect` flag is not set, the MySQL JDBC driver will eventually time out, and bamboo will no longer be able to communicate with the database.

- If you plan to use non-latin characters, you will also need to add `&useUnicode=true&characterEncoding=utf8` on the end of the above URL. These options are not required for any database other than MySQL.

Driver Class Name:

The class name of the database driver. Ensure that this class is in your class path.

Database URL:

The url to access the database.

User Name:

The username to access the database.

Password:

(optional) Enter the password if the database configuration requires it.

☐ **Overwrite existing data**

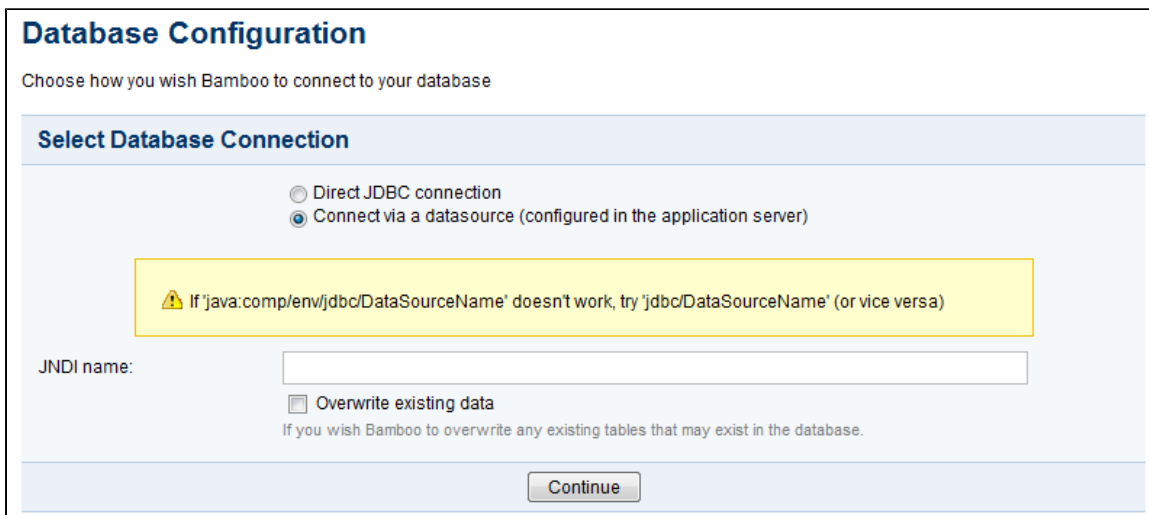
If you wish Bamboo to overwrite any existing tables that may exist in the database.

Connect using a datasource

To connect Bamboo to a MySQL database using a datasource:

1. Configure a datasource in your application server (consult your application server documentation for details). Please note the following:
 - Ensure that the JDBC URL which you configure in your application server includes the `autoReconnect=true`, `useUnicode=true` and `characterEncoding=utf8` flags, such that your database URL should look similar to: `jdbc:mysql://localhost/bamboo?autoReconnect=true&useUnicode=true&characterEncoding=utf8`
 - If your MySQL database server is configured to use a storage engine other than InnoDB by default, also include the `sessionVariables=storage_engine=InnoDB` flag in this URL.
 - If the `autoReconnect` flag is not set, the MySQL JDBC driver will eventually time out and Bamboo will no longer be able to communicate with the database.
For more information on the URL syntax, please see the [MySQL documentation](#).
 - Datasource example: You can see an example of using Tomcat with a MySQL database as a datasource in the following document: [Tomcat and External MySQL Datasource Example](#).
2. Run the [Setup Wizard](#) and choose the **Custom Installation** method.
3. On the 'Choose a Database Configuration' page, choose **External Database > MySQL 5.1** from the list and click **Continue**. The 'Database Configuration' page appears.
4. Choose **Connect via a datasource (configured in the application server)** (as shown in the screenshot below).
5. In the **JNDI name** field, type the JNDI name of your datasource, as configured in your application server.
 If `java:comp/env/jdbc/DatasourceName` does not work, try `jdbc/DatasourceName` (and vice versa).
6. Select **Overwrite existing data** if you wish Bamboo to overwrite any tables that already exist in the database.
7. Click **Continue**.

Screenshot 2: Setup Datasource Connection



Tomcat and External MySQL Datasource Example

Within the Context tags of your context descriptor (bamboo.xml), directly after the opening `<Context...>` line, insert the DataSource Resource tag:

```
<Context ....>
<Resource name="jdbc/bamboo" auth="Container" type="javax.sql.DataSource"
    username="yourusername"
    password="yourpassword"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/bamboo?autoReconnect=true"
    maxActive="15"
    maxIdle="7"
    validationQuery="Select 1" />
</Context>
```

Oracle 11g

These instructions will help you connect Bamboo to an [Oracle 11g](#) database. Oracle 10g is no longer a supported database for use with Bamboo and the 11.2.x drivers from Oracle do not support 9i.

Bamboo provides two ways to connect to an Oracle database — using JDBC or using a datasource. JDBC is generally simpler and is the

recommended method.



Important

- For JDBC or JNDI connections, please ensure that the user connecting to the database will have total permissions over it. This includes **DBMS_LOB package** and other resources available.
- Note that the JDBC driver for Oracle 11g (Oracle 11.2.0.1.0) is bundled with stand alone Bamboo. You do not have to download and install the driver if you are using a stand alone Bamboo distribution.

On this page:

- [Configuring Oracle](#)
- [Connecting using JDBC](#)
- [Connecting using a datasource](#)

Related pages:

[Troubleshooting Databases](#)

Configuring Oracle

1. Ensure that you have a database instance available for Bamboo (either create a new one or use an existing one).
2. Within that database instance, create a user which Bamboo will connect as (e.g. **bamboo-user**). Remember this database user name, as it will be used to configure Bamboo's connection to this database.



When you create a user in Oracle, Oracle will create a 'schema' automatically.

```
create user bamboo-user identified by password;
```

3. Ensure that the user has the following permissions:

```
grant connect, resource, create table to bamboo-user;
```

Connecting using JDBC

To connect Bamboo to a Oracle database, using JDBC:

1. [Run the Setup Wizard](#) and choose the **Custom Installation** method.
2. At the 'Choose a Database Configuration' step, choose **External Database > Oracle 11g**. The 'Select Database Connection' screen will appear.
3. Select **Direct JDBC connection**. The 'Setup JDBC Connection' screen will appear as shown in the screenshot below.

Setting	Description
Driver Class Name	Type: <code>oracle.jdbc.driver.OracleDriver</code>
Database URL	Type the URL where Bamboo will access your database, e.g. <code>jdbc:oracle:thin:@localhost:1521:SID</code> . For syntax, please see the Oracle documentation .
Username	Type the username that Bamboo will use to access your database.
Password	Type the password that Bamboo will use to access your database.

4. Select **Overwrite existing data** if you wish Bamboo to overwrite any tables that already exist in the database.
5. Click **Continue**.

Screenshot: *Setup JDBC Connection (Oracle)*

Setup JDBC Connection

Driver Class Name:

The class name of the database driver. Ensure that this class is in your class path.

Database URL:

The url to access the database

User Name:

The username to access the database

Password:

(optional) Enter the password if the database configuration requires it

☐ Overwrite existing data

If you wish Bamboo to overwrite any existing tables that may exist in the database.

Continue

Connecting using a datasource

To connect Bamboo to a Oracle database, using a datasource:

1. Configure a datasource in your application server (consult your application server documentation for details). For the syntax of the JDBC URL to use, please see the [Oracle documentation](#).
2. Run the [Setup Wizard](#) and choose the **Custom Installation** method.
3. At the 'Choose a Database Configuration' step, choose **External Database > Oracle 11g**.
4. Select **Connect using a datasource (configured in the application server)**. The 'Setup Datasource Connection' screen will appear as shown in the screenshot below.
5. In the **JNDI name** field, type the JNDI name of your datasource, as configured in your application server.
6.  If `java:comp/env/jdbc/DataSourceName` doesn't work, try `jdbc/DataSourceName` (and vice versa).
7. Select **Overwrite existing data** if you wish Bamboo to overwrite any tables that already exist in the database.
7. Click **Continue**.

Screenshot Setup Datasource Connection

Setup Datasource Connection

JNDI name:

☐ Overwrite existing data

If you wish Bamboo to overwrite any existing tables that may exist in the database.

Continue

Microsoft SQL Server 2005 and 2008

These instructions will help you connect Bamboo to a [Microsoft SQL Server 2005](#) or a [Microsoft SQL Server 2008](#) database. (Microsoft SQL Server 2000 is not a supported database for use with Bamboo.)

 **Express Editions:** *SQL Server Express 2005 and 2008 are not recommended databases due to CPU, memory and database size limitations (please see these pages for full details: [SQL Server Express 2005 feature comparison](#), [SQL Server Express 2008 feature comparison](#)). However, the instructions below will allow you to connect Bamboo to SQL Server Express 2005/2008.*

 Please note, the JDBC driver for SQL Server 2005/2008 (JTDS 1.2.2) is bundled with Bamboo. You do not have to download and install the driver.

On this page:

1. [Configuring SQL Server](#)
2. [Creating Your Database](#)
3. [Connecting Bamboo to SQL Server](#)
 - [Connecting using JDBC](#)
 - [Connecting using a datasource](#)
 - [Unicode Characters Not Supported By Default](#)

Related pages:

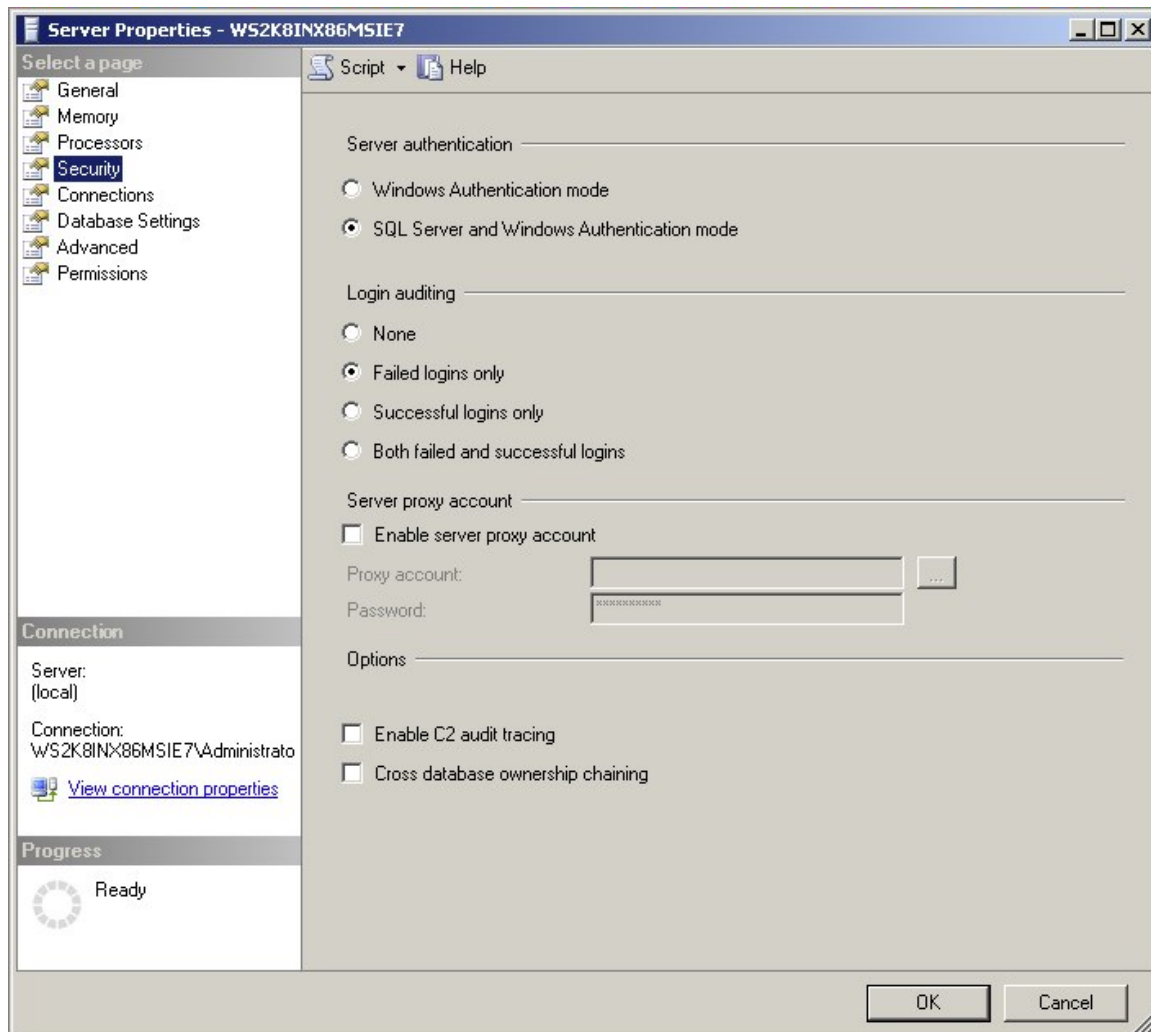
- [Bamboo installation guide](#)
- [Connecting Bamboo to an external database](#)
- [Troubleshooting Databases](#)

1. Configuring SQL Server

Before you connect Bamboo to a SQL Server, you need to configure the SQL Server appropriately.

- **Change server authentication to 'SQL Server and Windows Authentication mode'** — On a typical SQL Server installation, **Windows Authentication mode** is the default security mode. However, if you try to connect to the database with a database user using this authentication mode, SQL Server will throw an error. You need to change the server authentication mode to **SQL Server and Windows Authentication mode** in SQL Server before you can connect Bamboo to SQL Server. Please see [this MSDN article](#) for instructions on how to do this.

Screenshot: Changing the SQL Server authentication mode



- **Configure your firewall to allow SQL Server access** — If you need to access SQL server through a firewall, you will need to configure your firewall appropriately. The following MSDN article describes how to configure a Windows firewall to allow SQL Server access, however the instructions are applicable to other firewalls: [Configuring the Windows Firewall to Allow SQL Server Access](#).
- **Enable the TCP/IP protocol for your database instance** — You must enable the TCP/IP protocol for your SQL Server database instance by following the instructions in [this MSDN article](#).

2. Creating Your Database

After configuring the SQL Server, you need to create the SQL database.

- **Create the database for Bamboo** — see [this MSDN article](#) for instructions.
- **Assign the 'db-owner' role on the database for the user that will access the Bamboo database** — the 'db_owner' fixed database role allows the user to perform all configuration and maintenance activities on the database. You need to add this role to the Bamboo user used to access your database by updating the login properties for your database user in SQL Server. Read more about [login properties for SQL Server](#).

Screenshot: Adding the 'db_owner' database role to a database user in SQL Server

Login Properties - foo

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Script Help

Users mapped to this login:

Map	Database	User	Default Schema
☐	master		
☐	model		
☐	msdb		
☒	mydb	foo	dbo
☐	ReportServer		
☐	ReportServerTempDB		
☐	tempdb		

☐ Guest account enabled for: mydb

Database role membership for: mydb

- ☐ db_accessadmin
- ☐ db_backupoperator
- ☐ db_datareader
- ☐ db_datawriter
- ☐ db_ddladmin
- ☐ db_denydatareader
- ☐ db_denydatawriter
- ☒ db_owner
- ☐ db_securityadmin
- ☒ public

Connection

Server:
(local)

Connection:
WS2K8IN\X86MSIE7\Administrato

[View connection properties](#)

Progress

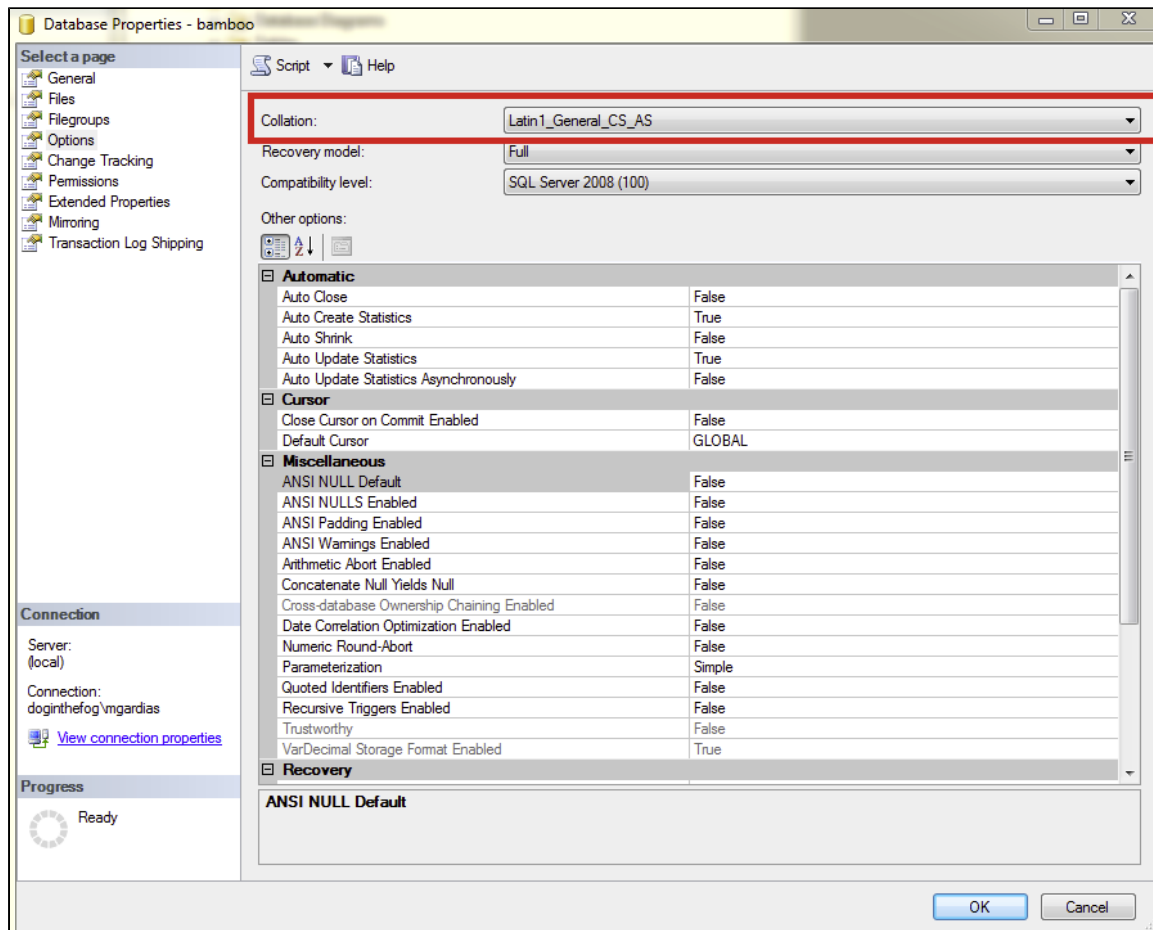
Ready

OK Cancel

⚠ Please ensure that you use a **SQL Server** user account to log into your database, not a Windows user account.

- **Configure the database to use case-sensitive collation** — to make the SQL Server database respect case differences in the data it stores (which is required for Bamboo), ensure that you configure it using a case-sensitive collation option such as 'Latin1_General_CS_AS'. To access this feature in SQL Server Management Studio, right-click on the database name, select **Properties** from the resulting menu, then select the **Options** page.

Screenshot: Configuring the Bamboo database to use 'Latin1_General_CS_AS' collation



- **Configure the database to use the correct isolation level**— Ensure that the new database was set to use Read Committed with Row Versioning as its isolation level. You can apply the new isolation by executing the following query:

```
ALTER DATABASE <database name>
SET READ_COMMITTED_SNAPSHOT ON
WITH ROLLBACK IMMEDIATE;
```

To verify the changes, use this query which should result in '1':

```
SELECT sd.is_read_committed_snapshot_on
FROM sys.databases AS sd
WHERE sd.[name] = '<database name>';
```

3. Connecting Bamboo to SQL Server

Bamboo provides two ways to connect to a Microsoft SQL Server database — using JDBC or using a datasource. JDBC is generally simpler and is the recommended method.

i If you are planning to support Unicode in Bamboo please [enable unicode settings for SQL Server](#)

Connecting using JBDC

To connect Bamboo to a Microsoft SQL Server database, using JDBC:

1. Run the [Setup Wizard](#) and choose the **Custom Installation** method.
2. On the **Choose a Database Configuration** page, choose **External Database > Microsoft SQL Server 2005/2008** and click **Continue**. The 'Database Configuration' page will appear.
3. Ensure that **Direct JDBC connection** has been selected and complete the following fields (as shown in the screenshot below):

Setting	Description

Driver Class Name	Type <code>net.sourceforge.jtds.jdbc.Driver</code> (if different from the default)
Database URL	The URL where Bamboo will access your database, e.g. <code>jdbc:jtds:sqlserver://localhost:1433/<database></code> . If you are connecting to a Named Instance , you will need to append <code>:instance=mssqlnamehere</code> to the connection string, where <i>mssqlnamehere</i> is the name of your named instance. For more details about syntax, please refer to the Microsoft SQL Server documentation .
Username	The username that Bamboo will use to access your database.
Password	The password that Bamboo will use to access your database.

4. Select **Overwrite existing data** if you wish Bamboo to overwrite any tables that already exist in the database.
5. Click **Continue**.

Screenshot: Set Up JDBC Connection SQL Server 2005/2008

Database Configuration

Choose how you wish Bamboo to connect to your database

Select Database Connection

☒ Direct JDBC connection
 ☐ Connect via a datasource (configured in the application server)

Driver Class Name:

The class name of the database driver. Ensure that this class is in your class path.

Database URL:

The url to access the database.

User Name:

The username to access the database.

Password:

(optional) Enter the password if the database configuration requires it.

☐ Overwrite existing data

If you wish Bamboo to overwrite any existing tables that may exist in the database.

Connecting using a datasource

To connect Bamboo to a MS SQL Server, using a datasource:

1. Configure a datasource in your application server (consult your application server documentation for details).
 -  For details about the syntax to use for the SQL Server database URL, please refer to the [Microsoft SQL Server documentation](#).
2. Run the **Setup Wizard** and choose the **Custom Installation** method.
3. On the 'Choose a Database Configuration' page, choose **External Database > Microsoft SQL Server 2005/2008** and click **Continue**. The 'Database Configuration' page will appear.
4. Choose **Connect via a datasource (configured in the application server)**, as shown in the screenshot below.
5. In the **JNDI name** field, type the JNDI name of your datasource, as configured in your application server.
 -  If `java:comp/env/jdbc/DataSourceName` does not work, try `jdbc/DataSourceName` (and vice versa).
6. Select **Overwrite existing data** if you wish Bamboo to overwrite any tables that already exist in the database.
7. Click **Continue**.

Screenshot: Set up Datasource Connection

Database Configuration

Choose how you wish Bamboo to connect to your database

Select Database Connection

- ☐ Direct JDBC connection
☒ Connect via a datasource (configured in the application server)

⚠ If 'java:comp/env/jdbc/DataSourceName' doesn't work, try 'jdbc/DataSourceName' (or vice versa)

JNDI name:

☐ Overwrite existing data

If you wish Bamboo to overwrite any existing tables that may exist in the database.

Continue

Unicode Characters Not Supported By Default

Problem

Non-ASCII characters will not be displayed by Bamboo.

Reason

The default SQL Server dialect uses column types that do not support Unicode, specifically the **char**, **varchar** and **text** column types. See [C ONF-4786](#) for details.

Solution

To add Unicode support, use the Unicode SQL Server dialect which uses **nchar**, **nvarchar** and **ntext** column types. Unicode SQL Server dialect has the downside of halving the maximum length of each column from 8000 characters to 4000, as every char is stored in two bytes.

Enable Unicode SQL Server dialect on a new setup, perform these steps prior to 'Step 3 - Database Connection Setup'.

1. Open the `<bamboo-install>\webapp\WEB-INF\classes\database-defaults\mssql.properties` file within your Bamboo installation folder.
2. Comment the line: `dialect=net.sf.hibernate.dialect.SQLServerDialect`
3. Uncomment the line: `#dialect=net.sf.hibernate.dialect.SQLServerIntlDialect`
4. Start the Bamboo Setup Wizard

How do I connect Bamboo to an unsupported database

We *strongly recommend* that you use Bamboo with one of the databases that we support (see [Supported platforms](#) for details). However, if you wish to connect Bamboo to an unsupported database, you can do so using the instructions below.

First, choose one of the following methods by which you will connect to your database:

- Connecting using JDBC
- Connecting using a datasource.

Then follow the instructions for that method. Note that using JDBC is generally simpler, and is therefore the recommended method.

On this page:

- [Connecting using JDBC](#)
 - [Hibernate database dialects](#)
- [Connecting using a datasource](#)

Related pages:

- [Bamboo installation guide](#)
- [Connecting Bamboo to an external database](#)
- [Moving your Bamboo data to a different database](#)
- [Troubleshooting Databases](#)

Connecting using JBDC

To connect Bamboo to an unsupported database, using JDBC:

1. Put the appropriate JDBC driver `jar` file into your application server's classpath:
 - For the Bamboo distribution, copy the `jar` file into the **webapp/WEB-INF/Lib** directory.
 - For the Bamboo EAR-WAR distribution, the location will depend on which application server you are using.
2. [Set the following system property](#) before starting your upgraded Bamboo server to enable "Unsupported Database" as a selectable option in the Setup Wizard:

-Dbamboo.enable.unsupported.db=true

3. At Step 2 of the [Bamboo Setup Wizard](#), choose **External Database > Unsupported Database**.
4. In the 'Select Database Connection' screen, choose **Direct JDBC connection**.
5. In the 'Setup JDBC Connection' screen, make the following settings:

Setting	Description
Driver Class Name	The classname of your JDBC driver (consult your JDBC driver documentation for details).
Database URL	The URL where Bamboo will access your database (consult your JDBC driver documentation for details).
UserName	The username that Bamboo will use to access your database.
Password	The password (if required) that Bamboo will use to access your database.
Hibernate Dialect	The Hibernate dialect for your particular database. See the table of dialects below.  NOTE: the databases in this list are <i>not supported by Atlassian</i> . Using these databases is not recommended as there is no guarantee that they will operate correctly with Bamboo. Please consider using a supported database instead.

6. Select **Overwrite existing data** if you wish Bamboo to overwrite any tables that already exist in the database.
7. Go to Step 3 of the [Setup Wizard](#).

Hibernate database dialects

This table lists the [Hibernate](#) dialects that are available for particular databases.

Database	Dialect
DB2	net.sf.hibernate.dialect.DB2Dialect
DB2 AS/400	net.sf.hibernate.dialect.DB2400Dialect
DB2 OS390	net.sf.hibernate.dialect.DB2390Dialect
Oracle 9/10g	net.sf.hibernate.dialect.Oracle9Dialect
Oracle (other versions)	net.sf.hibernate.dialect.OracleDialect
Sybase	net.sf.hibernate.dialect.SybaseDialect
Sybase Anywhere	net.sf.hibernate.dialect.SybaseAnywhereDialect
Microsoft SQL Server	net.sf.hibernate.dialect.SQLServerDialect
SAP DB	net.sf.hibernate.dialect.SAPDBDialect
Informix	net.sf.hibernate.dialect.InformixDialect
Ingres	net.sf.hibernate.dialect.IngresDialect
Progress	net.sf.hibernate.dialect.ProgressDialect
Mckoi SQL	net.sf.hibernate.dialect.MckoiDialect
Interbase	net.sf.hibernate.dialect.InterbaseDialect
Pointbase	net.sf.hibernate.dialect.PointbaseDialect
FrontBase	net.sf.hibernate.dialect.FrontbaseDialect
Firebird	net.sf.hibernate.dialect.FirebirdDialect

Connecting using a datasource

To connect Bamboo to an unsupported database, using a datasource:

1. Configure a datasource in your application server (consult your application server documentation for details). For the syntax of the JDBC URL to use, please see your JDBC driver documentation.
2. [Set the following system property](#) before starting your upgraded Bamboo server to enable "Unsupported Database" as a selectable option in the Setup Wizard:
-Dbamboo.enable.unsupported.db=true
3. At Step 2 of the [Bamboo Setup Wizard](#), choose **External Database > Unsupported Database** from the list.

4. In the 'Select Database Connection' screen, select **Connect via a datasource (configured in the application server)**.
5. The 'Setup Datasource Connection' screen is displayed. In the **JNDI name** field, type the JNDI name of your datasource, as configured in your application server.
 If `java:comp/env/jdbc/DatasourceName` doesn't work, try `jdbc/DatasourceName` (and vice versa).
6. Select **Overwrite existing data** if you wish Bamboo to overwrite any tables that already exist in the database.
7. Go to Step 3 of the [Setup Wizard](#).

Add-ons

An add-on is an installable component that supplements or enhances the functionality of Bamboo in some way. For example, the [JIRA Bamboo Plugin](#) is an add-on that integrates JIRA and Bamboo. Other add-ons are available for integrating Bamboo into the Visual Studio IDE, running arbitrary commands before or after builds, and accessing Atlassian support from the Bamboo interface.

Bamboo comes with many pre-installed add-ons (called system add-ons). You can install more add-ons, either by acquiring the add-on from the [Atlassian Marketplace](#) or by uploading it from your file system. This means that you can install add-ons that you have developed yourself. For information about developing your own add-ons for Bamboo, see the [Bamboo Developer documentation](#).

On this page:

- [About the Universal Plugin Manager \(UPM\)](#)
- [Administering Add-ons in Bamboo](#)



You may notice that the terms 'add-on' and 'plugin' both appear in the Atlassian documentation and tools. While the terms are often used interchangeably, there is a difference. A plugin is a type of add-on that can be installed into an Atlassian host application. Plugins are what developers create with the Atlassian SDK. But there are other types of add-ons as well. For example, the JIRA client is an add-on that runs as a separate program rather than as a plugin to JIRA. This documentation uses the term 'add-on' most often.

About the Universal Plugin Manager (UPM)

You administer add-ons for Bamboo using the Universal Plugin Manager (UPM). The UPM is itself an add-on that exposes add-on administration pages in the Bamboo Administration Console. UPM works across Atlassian applications, providing a consistent interface for administering add-ons in Bamboo, JIRA, Confluence, Crucible, Fisheye or Stash.

UPM comes pre-installed in recent versions of all Atlassian applications, so you do not normally need to install it yourself. However, like other add-ons, the UPM software is subject to regular software updates. Before administering add-ons in Bamboo, therefore, you should [verify your version](#) of the UPM and update it if needed.

Administering Add-ons in Bamboo

You can update UPM, or any add-on, from the UPM's own add-on administration pages. Additionally, you can perform these tasks from the UPM administration pages:

- Install or remove add-ons
- Configure add-on settings
- Discover and install new add-ons from the [Atlassian Marketplace](#)
- Enable or disable add-ons and their component modules

For information on performing these add-on administration tasks, see the [Universal Plugin Manager documentation](#).

For add-on information specific to Bamboo, see these pages:

- [Add-on blacklist](#)
- [Enabling the Clover add-on](#)

Add-on blacklist

Outdated add-ons may break certain functionality in Bamboo. If Bamboo detects the presence of a non-working add-on it will print a warning to its logs during startup and ask you to refer to this page.

For more information about why Bamboo printed a particular warning, please refer to a section below that is relevant to the add-on in question.

Experimental Bamboo Git Plugin

Since version 3.0, Bamboo is distributed with a fully supported version of the Bamboo Git Plugin.

The experimental Bamboo Git Plugin that was available before Bamboo 3.0 (and was not distributed with Bamboo) does not work with Bamboo 3.0 and later.

If you were using the experimental Bamboo Git Plugin, please remove the add-on from your Bamboo installation, and manually reconfigure each plan that was using it to use the Bamboo Git Plugin that is distributed with Bamboo.

Enabling the Clover add-on

This page contains instructions about enabling and configuring Atlassian's [Clover](#) add-on for a job in Bamboo.

When Bamboo is integrated with Clover, you can:

- View code-coverage details (i.e. the percentage of code covered by tests) for each build result
- View code-coverage trends for a job over a period of time
- View the code-coverage summary for the job.

To enable the Clover add-on on a job:

1. Navigate to the desired job, as described on [Configuring jobs](#).
2. Choose **Actions > Configure Job**.
3. Click the **Miscellaneous** tab.
4. Select **Use Clover to collect Code Coverage for this build** and complete the following settings:

On this page:

- [Automatic Clover integration](#)
- [Manual Clover integration](#)
- [Browsing Clover results](#)
- [Troubleshooting](#)

Related pages:

- [Using Bamboo with Clover](#)
- [Viewing the Clover code-coverage for a build](#)
- [Viewing the Clover code-coverage for a plan](#)

Atlassian Blogs:

- [Aggregated code coverage using Maven, Clover and Bamboo](#)

Setting	Description
Automatically integrate Clover into this build	You will need to provide a Clover license (evaluation licenses are available), unless this has been configured globally in the Administration panel (Administration > Plugins > Clover Plugin).
Generate a Clover Historical Report	Displays the current coverage results compared with previous Clover code coverage reports.
Generate a JSON report	Provides the Clover results in a format ready for embedding into applications or external report views.
Use plan-defined Clover license key	Override the global Clover license for this particular plan.
Clover is already integrated into this build	Use this option when you already have Clover-for-Ant or Clover-for-Maven configured to generate a report.
Clover XML Location	Specify the location where Bamboo will look for the XML report file from Clover. Please specify the file path <i>relative to your plan's root directory</i> (e.g /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. do not specify an absolute path. <pre>target/site/clover/clover.xml</pre>

Screenshot: Settings to enable Clover for a job

Would you like to view Clover Code Coverage for this plan?

☒ Use Clover to collect Code Coverage for this build.

Clover is a code coverage tool reports how well tested your code is and also highlights parts of code that require more testing. For more information visit: [Atlassian Clover](#)

Integration Options ☒ Automatically integrate Clover into this build.

☐ Clover is already integrated into this build and a clover.xml file will be produced. For more information visit: [Atlassian Clover](#) or view the [Online Documentation](#).

☐ Generate a Clover Historical Report

Include coverage trends and class movers in the Clover HTML report. [More info](#).
(Will only work consistently if this plan is run on a single agent and no clean checkout is performed.)

☐ Generate a JSON report

JSON makes it very easy to integrate Clover data into a web-page. [Learn how](#).

 Global Clover license has been configured in administration panel. To override use option below.

☐ Use plan-defined Clover license key

Override globally defined Clover license and provide dedicated license for this plan.

Automatic Clover integration

Automatic integration works with Ant, Maven 2.x, Maven 3.x and Grails tasks.

To enable automatic integration:

1. Check **Use Clover to collect Code Coverage for this build** in the Clover Code Coverage settings.
2. Select **Automatically integrate Clover into this build**.
3. Enter the global license key for Clover (go to **Administration > Plugins > Clover Plugin**), or enable **Use plan-defined Clover license key** and paste the key into the text field that appears.

Additionally, you can:

- Select **Generate a Clover Historical Report** to compare the current coverage results with previous Clover code coverage reports.
- Select **Generate a JSON report** to get the Clover results in a format ready for embedding into applications or external report views.

When automatic Clover integration is enabled, Bamboo will:

Initially:

- Create an artifact named **Clover Report (System)**, which will be visible on the 'Artifacts' tab for the job.

and during every build:

- Extract the Clover licence (either the global or plan license key) into a temporary file and pass it to:
 - an Ant task as `-Dclover.license.path=/<bamboo-home>/xml-data/build-dir/<your-job>/clover/clover.license`
 - a Maven task as `-Dmaven.clover.licenseLocation=/<bamboo-home>/xml-data/build-dir/<your-job>/clover/clover.license`
- Enhance tasks by adding
 - Ant targets
 - Maven goals like `"clover2:setup verify clover2:aggregate clover2:clover"`
- Generate Clover XML and HTML reports (by default)
- Generate statistics and charts for plan summary

Manual Clover integration

Manual Clover integration works with any kind of task in which Clover can be called (Ant, Maven 2.x, Maven 3.x, Command, Grails).

To enable manual integration:

1. Check **Use Clover to collect Code Coverage for this build**, in the Clover Code Coverage settings.
2. Check **Clover is already integrated into this build ...**
3. Specify in **Clover XML Location** where Bamboo will look for the XML report file generated by Clover.
4. On the 'Artifacts' tab, click **Create Definition** and complete the form as follows:

Name	This should begin with with "Clover Report".
Location	This should point to the HTML report directory (e.g. target/site/clover)
Copy Pattern	Use <code>**/*.*</code>

Plan Configuration

Job Details
Tasks
Requirements
Artifacts
Miscellaneous

Run

Stages & Jobs
1

Default Stage

- Default Job

Branches
0

No Branches are configured. Add one through the [configuration](#).

Artifact Definitions

Jobs may create Artifacts you want to keep or share with others Jobs (e.g. Clover, Agilifier, JAR files, Definitions allow you to define which Artifacts you care about.

Name	Location	Copy Pattern
Clover Report (System)	target/site/clover	**/*.*

- Configure Clover in your build script so that it generates both XML and HTML reports.

Ant:

```
<clover-report initstring="target/clover/database/clover.db">
  <current outfile="target/site/clover/clover.xml" />
  <current outfile="target/site/clover">
    <format type="html"/>
  </current>
</clover-report>
```

Maven:

```
<plugin>
  <groupId>com.atlassian.maven.plugins</groupId>
  <artifactId>maven-clover2-plugin</artifactId>
  <configuration>
    <generateHtml>true</generateHtml>
    <generateXml>true</generateXml>
  </configuration>
</plugin>
```

- Configure the Clover license in your build script or pass it as a proper task parameter in the job configuration:
 - Save the Clover license key in a file (for example in `/opt/bamboo/clover.license`).
 - Pass the location of the license key to the build task:
 - Define it in the build script, or
 - Pass it as a Java property for the Ant/Maven task in the plan configuration.

Example - declare license location in pom.xml (Maven)

```
<plugin>
  <groupId>com.atlassian.maven.plugins</groupId>
  <artifactId>maven-clover2-plugin</artifactId>
  <version>3.1.8</version>
  <configuration>
    <licenseLocation>/opt/bamboo/clover.license</licenseLocation>
    <generateXml>true</generateXml>
    <generateHtml>true</generateHtml>
  </configuration>
</plugin>
```

Example - declare license location in build.xml (Ant)

```
<project>
  <property name="clover.license.path" location="/opt/bamboo/clover.license"/>
  <!-- ... -->
</project>
```

Example - pass license location for Ant task:

```
clean with:clover test clover:report
-Dclover.license.path=/opt/bamboo/clover.license
```

Example - pass license location for Maven task:

```
mvn clean clover2:setup verify clover2:aggregate clover2:clover
-Dmaven.clover.licenseLocation=/opt/bamboo/clover.license
```

Tasks

A Task is a piece of work that is being executed as part of the Build. The execution of a script, a shell command, an Ant Task or a Maven goal are only few. You can use [Runtime](#), [Plan](#) and [Global variables](#) to parameterize your Tasks.

After every build, Bamboo will parse the Clover XML file and generate statistics and charts for a plan summary. The Plan summary and job summary pages will contain a "Clover" tab.

Browsing Clover results

Clover HTML report and Clover statistics for a job: see [Viewing the Clover code-coverage for a plan](#).

Clover code coverage summary for a plan: see [Viewing the Clover code-coverage for a build](#).

Clover code coverage statistics across multiple plans: see [Generating reports across multiple plans](#).

Troubleshooting



Using automatic Clover integration or adding dependency to the *maven-clover2-plugin* manually is usually sufficient.

However, if your build spawns another JVM process (for example: unit tests executed in a forked JVM, tests in the container instantiated on the fly, tests calling code deployed on another server), you must manually add dependency to the Clover JAR for these spawned processes.

See [NoClassDefFoundError com_cenqua_clover/CoverageRecorder](#) KB article.

Data and backups

For information on managing data and backups, see the following topics:

- [Locating important directories and files](#)
- [Specifying Bamboo's working directory](#)
- [Viewing your database connection details](#)
- [Moving your Bamboo data to a different database](#)
- [Optimising or re-indexing data](#)
- [Specifying a backup schedule](#)
- [Exporting data for backup](#)
- [Importing data from backup](#)
- [Configuring global build results expiry](#)
- [Importing data from Jenkins](#)

Locating important directories and files

The information on this page describes how to find important Bamboo directories and files.

On this page:

- [Bamboo server installation directory](#)
- [Bamboo server home directory](#)
- [Bamboo agent home directory](#)

Bamboo server installation directory

When you [installed](#) your Bamboo server, you specified the location for the *Bamboo installation directory* — this is the directory where the Bamboo application files are installed. (The default location depends on your operating system: [Windows](#), [Unix/Linux](#), [Solaris](#) or [Mac OS](#).)

webapp/WEB-INF/classes/bamboo-init.properties	This file tells Bamboo where to find the <i>Bamboo home directory</i> . The location of this directory is specified by the Bamboo administrator as described in the Bamboo installation guide, and can be viewed as described in Viewing Bamboo's System Information .
bamboo.sh *	This is the startup file for Bamboo under Unix/Linux , Solaris and Mac OS .
bamboo.bat *	This is the startup file for the Bamboo under Windows .
bamboo.pid *	This file, under Linux, contains the Process ID for the running instance of Bamboo.
conf/wrapper.conf *	This file is used to configure Bamboo on startup, when using the Java Service wrapper under Linux or Windows .
scripts/	This directory contains operational scripts, including scripts for CVS and SVN triggers.
wrapper/*	This directory contains the necessary files to start Bamboo using the Java Service wrapper (see the Mac and Linux installation guides).
logs/*	This directory contains logs (atlassian-bamboo.log) written by the Java Service wrapper, unless you have used the Installer for Windows. (Note: The Bamboo server logs are written to the root of the installation directory. Build logs are stored in the xml-data/builds/ sub-directories.)  If you used the Installer for Windows, log files will be located at %USERPROFILE%\bamboo.log. For Bamboo running as a Windows service it can be found at %WINDIR%\System32\Config\systemprofile\bamboo.log.
webapp/	This directory contains all the Bamboo server application files.
webapp/WEB-INF/lib/	This directory is used when deploying Bamboo add-ons . It also contains other libraries required by Bamboo.
webapp/WEB-INF/classes/log4j.properties	This is Bamboo's logging configuration file.
webapp/WEB-INF/classes/jetty.xml *	This is the configuration file for Jetty, the application server that is bundled with Bamboo distribution .

* This applies to the [Bamboo distribution](#). The configuration may differ for the [Bamboo EAR-WAR distribution](#).

Bamboo server home directory

When you [installed](#) your Bamboo server, you specified the location for the *Bamboo home directory* — This is the directory where your Bamboo configuration data and build results are stored. (The default location depends on your operating system: [Windows](#), [Unix/Linux](#), [Solaris](#) or [Mac OS](#).) This directory can grow quite large when managing large quantities of plans and builds.

artifacts/ <i>PLAN_KEY</i> /shared/build- <i>BUILD_NUMBER</i> /	This is a folder shared by all the stages of a certain plan. Stages will place Artifacts here so that other stages from the same plan can have access to them. The <i>BUILD_NUMBER</i> will always have a minimum of 5 digits, having the number completed with zeros when necessary. For instance, for build "42" the number will be "00042".
bamboo.cfg.xml	This is Bamboo's core configuration file. It includes the configuration information for connecting to Bamboo's database.
xml-data/	This directory contains all files relating to source repositories and build results .

xml-data/build-dir/ <i>JOB_KEY</i>	This is known as the <i>Working Directory</i> . This is where Bamboo temporarily puts the checked-out files it is building. The location of this directory was specified using the Setup Wizard , can be viewed as described in Viewing Bamboo's System Information , and can be changed as described in Specifying Bamboo's Working Directory .
xml-data/builds/	This is known as the <i>Build Directory</i> . This is where Bamboo stores build results (note that they will be deleted as described in Configuring global build results expiry). The location of this directory was specified using the Setup Wizard , and can be viewed as described in Viewing Bamboo's System Information . Its contents can be backed up as per Exporting data for backup .
xml-data/builds/ <i>JOB_KEY</i> /results	Contains the build results for all the builds belonging to the <i>JOB_KEY</i> plan. Each build result is an individual XML file. Do not edit these files or the corresponding information in the database may become corrupt.
xml-data/builds/ <i>JOB_KEY</i> /download-data	Contains the logs for each build belonging to the <i>JOB_KEY</i> plan.
xml-data/configuration/	This is known as the <i>Configuration Directory</i> . It contains server-wide configuration information. The location of this directory was specified via the Setup Wizard , and can be viewed as described in Viewing Bamboo's System Information . Its contents can be backed up as per Exporting data for backup .
database/	This directory contains Bamboo's embedded HSQL database. The database contains plan configurations and some build results data.
index/	This directory contains the build results index. Removing or modifying files in this directory may corrupt build history. Rebuilding the search index from Bamboo's global administration screen (see Optimising or re-indexing data) will completely regenerate the contents of this directory.
logs/*	This directory contains logs (bamboo.log) written by the Java Service wrapper, unless you have used the Installer for Windows. (Note: The Bamboo server logs are written to the root of the installation directory. Build logs are stored in the xml-data/builds/ sub-directories.)  If you used the Installer for Windows, log files will be located at %USERPROFILE%\bamboo.log. For Bamboo running as a Windows service it can be found at %WINDIR%\System32\Config\systemprofile\bamboo.log.  The difference between ../logs/atlassian-bamboo.log (located in the Bamboo home directory) and the ../logs/bamboo.log (located in the Bamboo Installation directory) is that the last one (bamboo.log) gets recreated when the Bamboo instance gets restarted.

Bamboo agent home directory

When you [installed](#) your remote agents (if any), you specified the location for the *Agent home directory* — this is the directory where the agent's configuration data is stored. The default name of this directory is `bamboo-agent-home`. (The default location depends on your operating system: [Windows](#), [Unix/Linux](#), [Solaris](#) or [Mac OS](#).) This directory can grow quite large when managing large numbers of plans and builds.

The contents of the agent home directory are:

bamboo-agent.cfg.xml	This contains configuration information about this remote agent. Most notably, it stores the agent id, which gets generated the first time this agent connects to the Bamboo server.
xml-data/build-dir/	This is where the agent will check out the files and perform builds (similar to the Bamboo server's xml-data/build-dir/ directory)

Specifying Bamboo's working directory

The *Working Directory* is where Bamboo temporarily puts the checked-out files it is building. The location of this directory was specified using the [Setup Wizard](#), can be viewed as described in [Viewing Bamboo's System Information](#), and can be changed as described below.

By default, this directory is located under the `xml-data` directory in the [Bamboo home directory](#).

Each build's jobs have their own working directory relative to this configured *working directory*:

```
xml-data/build-dir/JOB_KEY
```


To change the location of Bamboo's working directory:

1. Shut down Bamboo.
2. Open the <Bamboo-Home>/bamboo.cfg.xml file in a text editor. Find the following line -

```
....
    <property
name="buildWorkingDir">/home/Bamboo-home/xml-data/build-dir</property>
....
```

3. Edit the Bamboo working directory to point to a new folder on disk.
 4. **Save** the changes and restart Bamboo.
- Note: Bamboo will do a fresh checkout and perform a clean build of *all your plans*, once the directory is changed.

Viewing your database connection details

When you installed Bamboo, you would have set up a database connection by following one of these processes:

Once Bamboo is running, you can view the database configuration details as follows.

Related pages:

- [Data and backups](#)

To view your database connection details:

- Click **Administration** in the top navigation bar.
- Click **Database Configuration** in the left navigation column, under 'System'.

Screenshot: Database Configuration

Database Configuration

Database Details

Driver Class Name	org.postgresql.Driver
Database URL	jdbc:postgresql://localhost:5432/prodpanda
User Name	postgres
Hibernate Dialect	net.sf.hibernate.dialect.PostgreSQLDialect

Moving your Bamboo data to a different database

When you installed Bamboo, you would have set up a database connection by following one of these processes:

You may later wish to use a different database. For example, the embedded HSQL database is suitable for evaluation purposes only — you would typically move to an external database before deploying Bamboo in production.

Related pages:

- [Data and backups](#)

To move your Bamboo data to a different database:

1. Back up your Bamboo data as described in [Exporting data for backup](#). Note the filename and path of the exported file for use in Step 8 below.
2. Shut down your old instance of Bamboo.
3. If your old instance of Bamboo was configured to start automatically (e.g. as a Windows service), disable it.
4. Install a new instance of Bamboo as described in the [Bamboo installation guide](#). Specify a *different* **Home Directory*** and **Installation Directory*** from the directories used by your old instance of Bamboo. (If you use the same locations, your existing data will be deleted.)
5. Launch your new instance of Bamboo. You will see the Setup Wizard.
6. At **Step 1** of the [Setup Wizard](#), ensure that your new **Configuration Directory***, **Build Data Directory*** and **Build Working Directory*** are in *different* locations to your old instance of Bamboo.
7. At **Step 2** of the [Setup Wizard](#), select your *new database* and follow the appropriate instructions for your chosen database:
 - [PostgreSQL 8.2](#)
 - [MySQL 5.1](#)
 - [Tomcat and External MySQL Datasource Example](#)
 - [Oracle 11g](#)
 - [Microsoft SQL Server 2005 and 2008](#)
 - [How do I connect Bamboo to an unsupported database](#)

- At **Step 3** of the [Setup Wizard](#), select **Import existing data** and specify the export file created in Step 1 above.
- Wait while Bamboo imports your data. (You will not need to complete any more steps of the Setup Wizard.)
- When the data import has finished, restart your new instance Bamboo.
- Re-index your Bamboo data as described in [Optimising or re-indexing data](#).
- Verify that your build results and [system settings](#) look the same as before.

*For information about the contents of these directories, please see [Locating important directories and files](#)

Optimising or re-indexing data

About optimising

You may want to *optimise* your Bamboo build results data if you notice that search-intensive operations (e.g. [reporting](#)) are becoming slow. Bamboo will still be accessible while the optimisation process is running.

About re-indexing

You will need to *re-index* your Bamboo build results data whenever you perform a [data import](#). Re-indexing your data can also be helpful if your reports appear to be out-of-sync with your data. Bamboo *will not* be accessible while the re-indexing process is running. This may take a few minutes to complete (see [System Information](#) for an estimate of how long it will take).

Related pages:

- [Data and backups](#)

To re-index or optimise Bamboo's build results data:

- Click **Administration** in the top navigation bar.
- Click **Indexing** in the left navigation column, under 'System'.
- Select either **Full reindex** or **Optimise current index**, and click **Perform**.

Specifying a backup schedule

You can configure Bamboo to automatically create a backup each night, rather than doing a [manual export](#) every time.

Before you begin,

- Bamboo will be unavailable while the backup process completes. The export itself may take a long time to complete, depending on the number of builds and test. We recommend running your backups at a time of day or night when usage is low.
- Backups may require large amounts of disk space, depending on the number of builds and tests. Please make sure you have enough disk space in your desired backup location before proceeding.
- Bamboo will not export if plans are currently being built (see [Using the Bamboo dashboard](#)).

On this page:

- [Specifying a backup schedule](#)
- [Disabling a backup](#)

Related pages:

- [Data and backups](#)
- [Exporting data for backup](#)
- [Importing data from backup](#)

Specifying a backup schedule

To specify a backup schedule:

- Click **Administration** in the top navigation bar.
- Click **Scheduled Backups** in the left navigation column (under 'System').
- Click **Edit** to modify the schedule settings:

Setting	Notes
Disable scheduled backups	This check box must be cleared for automatic backups to be performed.
Backup Artifacts	Select if you want to include build artifacts in your scheduled backups.

Backup path	Specify the directory where you want to store your backups. Each backup will be stored as a single file. It may be necessary to modify the Bamboo <code>bamboo.paths.set.allowed</code> system property to do this. <i>Note that:</i> Bamboo restricts the editing of certain file path settings for security reasons (see Bamboo Security Advisory 2010-05-04). If you must configure Bamboo to permit modification to its file path settings, start Bamboo with the system property <code>bamboo.paths.set.allowed=true</code>. The procedure for configuring a Bamboo system property is described on Configuring system properties. Once you have configured your file path setting, we recommend removing or disabling the <code>bamboo.paths.set.allowed</code> system property and restarting Bamboo. If your Bamboo instance is accessible to anyone outside your organisation, then this will minimise the risk of Bamboo being compromised by security-related attacks.
Backup file prefix	Specify the first part of the filename for all your backup files.
Backup file date pattern	Specify the date/time format for identifying your individual backup files. This will be appended to Backup file prefix to form the complete filename for your backup files.
Schedule	Use the Schedule Editor to choose the frequency with which backups will be performed. See Cron-based scheduling for more information about the Schedule Editor.

- Click **Save**. Your first backup will run when your server's clock matches the specified time.

Disabling a backup

If you disable schedule backups, your schedule details will be retained but no automatic backups will be performed.

To disable a scheduled backup:

- Click **Administration** in the top navigation bar.
- Click **Scheduled Backups** in the left navigation column. The 'Scheduled Backup Details' page will be displayed, showing details about the status of scheduled backups or any currently configured backup.
- Click **Edit** to edit the current 'Scheduled Backup Details'.
- Select the **Disable scheduled backups** check box.
- Click **Save**.

Screenshot: Scheduled backups

Scheduled Backups

Configure Bamboo to generate a backup at specific time intervals.

Scheduled Backup Details

 Please choose the schedule for your backups carefully. Bamboo will be unavailable for the duration of the backup process.

☐ Disable scheduled backups
Switch scheduled backups on and off.

☐ Backup Artifacts
Should Bamboo backup build artifacts

Backup path **/Volumes/Pharlap/opt/dogfood/panda/home/backups/**

Backup file prefix
The prefix for the generated backup file

Backup file date pattern
The date/time format you specify will be added to the file prefix to differentiate between backup files

Schedule* **Each Saturday at 12:00 am** 

Exporting data for backup

The instructions on this page describe how to export Bamboo data for backup.

Before you begin:

- Bamboo will be unavailable while the backup process completes. The export itself may take a long time to complete, depending on the number of builds and tests. We recommend running your backups at a time of day or night when usage is low.
- Backups may require large amounts of disk space, depending on the number of builds and tests. Please make sure you have enough disk space in your desired backup location before proceeding.
- Bamboo will not export if plans are currently being built.
- Bamboo uses the third party TrueZip library to create zip archives. TrueZIP currently implements the ZIP32 specification only. This limits the maximum ZIP file length to 4GB. Unfortunately, Bamboo exports will fail if the resulting ZIP file is over 4GB. As a workaround, please export Bamboo without artifacts or [use a different backup strategy](#)
- User management settings for Bamboo will be saved as part of the export. For information on user management in Bamboo, see [Connecting to external user directories](#).
- **Export Directory Path** setting: Bamboo restricts the editing of certain file path settings for security reasons (see [Bamboo Security Advisory 2010-05-04](#)). If you must configure Bamboo to permit modification to its file path settings, start Bamboo with the system property `bamboo.paths.set.allowed=true`. The procedure for configuring a Bamboo system property is described on [Configuring system properties](#).
Once you have configured your file path setting, we recommend removing or disabling the `bamboo.paths.set.allowed` system property and restarting Bamboo. If your Bamboo instance is accessible to anyone outside your organisation, then this will minimise the risk of Bamboo being compromised by security-related attacks.

Related pages:

- [Data and backups](#)
- [Specifying a backup schedule](#)
- [Importing data from backup](#)

To export data for backup:

1. Click **Administration** in the top navigation bar.
2. Click **Export** in the left navigation column (under 'System').
3. Complete the following settings:

Export Directory Path	This can be configured – see the note above.

File Name	Edit the default name of the file to which Bamboo will export, if necessary.
Export Results	Clear this to export only the plan configurations.
Export Artifacts	Select to have Bamboo export build artifacts .
Export Build Logs	Select to have Bamboo export build logs.

- Click the **Export**. Bamboo creates the export file in the location shown for **Export Directory Path**.

Screenshot: Exporting data for backup

Export

Bamboo will be unavailable until the export process completes.

Specify Export Paths

Depending on the number of builds and tests, the export may take a long time to complete and may require large amounts of disk space. Please make sure you have enough disk space before proceeding.

Export Directory Path **C:\Documents and Settings\All Users\Bamboo43\exports**

File Name

Specify the file name for Bamboo export

- ☒ **Export Results**
Uncheck this box if you only want to export the plan configuration
- ☐ **Export Artifacts**
Should Bamboo export build artifacts
- ☒ **Export Build Logs**
Should Bamboo export build logs

Export

Importing data from backup

The instructions on this page describe how to import data from a [Bamboo backup](#).

Before you begin:

- Bamboo will be unavailable until the import process is complete, which may take some time.
- The import process will *delete* your Bamboo installation and restore data from a previous export of Bamboo. This includes login data, so you will need to know an administration login in the Bamboo data to be imported.
- If you created your backup file using Bamboo 3.2 or later, importing the file will restore your [user management settings](#). If you created your backup file using Bamboo 3.1 or earlier, importing the file will default your [user management settings](#) to 'Local users and groups' (i.e. user/group management in Bamboo). You may need to change your settings after the import.
- If you manage users externally (using LDAP or Crowd) and the Bamboo *internal* user repository (in the backup file) contains user names that duplicate user names in the external repository, you will not be able to import from the backup file.
- Backup Directory Path:** Bamboo restricts the editing of certain file path settings for security reasons (see [Bamboo Security Advisory 2010-05-04](#)). If you must configure Bamboo to permit modification to its file path settings, start Bamboo with the system property `bamboo.paths.set.allowed=true`. The procedure for configuring a Bamboo system property is described on [Configuring system properties](#).

Once you have configured your file path setting, we recommend removing or disabling the `bamboo.paths.set.allowed` system property and restarting Bamboo. If your Bamboo instance is accessible to anyone outside your organisation, then this will minimise the risk of Bamboo being compromised by security-related attacks.

Related pages:

- [Data and backups](#)
- [Specifying a backup schedule](#)
- [Exporting data for backup](#)

To import data from backup:

- Click **Administration** in the top navigation bar.
- Click **Import** in the left navigation column (under 'System').
- Complete the following settings:

File Path	The absolute path to the data file that Bamboo should import. For example, "/opt/bamboo/bamboo-home/export.zip" on UNIX-based operating systems.
Backup data	<i>Highly recommended.</i> Bamboo will not import data unless it is able to successfully export data first.
Backup Directory Path	This can be configured – see the note above.
File Name	The file to which Bamboo will export its data.
Clear artifact directory	Delete all existing build artifacts before the import.
Apply imported data without server shutdown	<i>Not recommended in a production environment.</i>

- Click **Import**.
- After the import is complete,
 - check the paths of your builders and JDK.
 - [index your data](#).

Screenshot: Importing data from a backup

Import

 You need an administration login in the import file to be able to make changes after import.

Specify Import Paths

The import process will DELETE this instance and restore data from a previous export of Bamboo. This includes login data hence you will need an administration login that is contained in the Bamboo data to be imported. Bamboo will be unavailable until the import process is complete which may take some time. Please check the paths of your executables and JDK after importing.

File Path	
	Specify the absolute path to the file on the server from which Bamboo is to import from. For example "C:\Documents and Settings\All Users\Bamboo43\export.zip".
	☒ Backup data? Although this will make the import process longer, backup is strongly recommended. Import will not proceed unless Bamboo successfully backs up.
Backup Directory Path	C:\Documents and Settings\All Users\Bamboo43\backups
File Name	backup.zip
	File name for backup file.
	☐ Clear artifact directory Remove all the existing artifacts before the import
	☐ Apply imported data without server shutdown Not recommended in production environment
Import	

Configuring global build results expiry

Global build expiry allows you to choose when build result data and artifacts will be deleted from your Bamboo system.

Build result data is used for such things as [reporting](#). If global build expiry is disabled, the build result data for your plans will never be deleted from your Bamboo server. This could lead to a large portion of your Bamboo server's storage space being used to store this data.

You can configure build results expiry for:

- all plans** (i.e. 'global', described below). This is generally the easiest way to manage build expiry in Bamboo. The global configuration applies to all plans that do not override the global build expiry settings.
- individual plans** (see [Configuring build results expiry for a plan](#)). You would generally only do this if there is a specific reason to keep/delete a particular plan's build result data.

You can also delete the results of a plan build manually — see [Deleting the results of a plan build](#).

You must be Bamboo administrator to configure global build results expiry.

On this page:

- [Configuring global expiry](#)
- [Disabling global expiry](#)

Related pages:

- [Data and backups](#)
- [Configuring build results expiry for a plan](#)
- [Configuring automatic labelling of job build results](#)

Atlassian Blogs:

- [Get to know Bamboo's build expiry and labels](#)

Configuring global expiry

Before you begin:

- If you enable build expiry, ensure that you [back up](#) your build results data before its expiry date is reached.

To enable and configure global expiry of build result data:

1. Click **Administration** in the top navigation bar.
2. Click **Build Expiry** (under 'Plans') in the left-hand panel.
3. Click **Edit**.
4. Click the icon on the right of the scheduled expiry to set when the build expiry event will be triggered. You can specify a cron expression if required. See [this FAQ](#) for help constructing cron expressions.
5. Clear the **Disable Global Build Expiry** checkbox, if necessary, to enable build expiry. The controls described below will be displayed.

Setting	Result
Build results	All build results data (including artifacts and build logs) are deleted.
Artifacts	Only user-defined artifacts are deleted from the build results.
Build logs	Only build logs are deleted from the build results.
Expiry period	Specifies the period (days, weeks or months) for which you want to keep build results. E.g. specify '24 months' to keep all build results for the last two years.
Minimum builds to keep	Specifies the minimum number of build results you want to keep. E.g. specify '50' to keep the latest 50 build results, even if they are older than the period specified with Expiry period .
Labels to keep	Specifies the build labels (not plan labels or job labels) applied to builds for which you want to keep build results, regardless of the Expiry period and Minimum builds to keep settings. Note that builds can be labelled either manually or automatically .

6. Click **Save**.

i Note that the **Disable Global Build Expiry** checkbox, described above, only controls the deletion of build results. The build expiry event (triggered by the expiry **Schedule**) will still run. For example, you may wish to disable build expiry globally, but still schedule a global build expiry event that triggers the deletion of build data from individual plans. See [Configuring build results expiry for a plan](#) for details on how to override the global build expiry settings.

Disabling global expiry

To disable global expiry of build result data:

1. Click **Administration** in the top navigation bar.
2. Click **Build Expiry** (under 'Plans') in the left-hand panel.
3. Click **Edit**, then check **Disable Global Build Expiry**.

Screenshot: Configuring build expiry

Build Expiry



The build expiry schedule determines when build results and/or artifacts will be deleted from your Bamboo system. On this page, you can configure when the build expiry schedule will run and what will happen to the build results globally when it runs. You can override what will happen to build results at the build plan level.

Build Expiry Settings

Bamboo will check for expired data based on the cron expression defined below.

Schedule* **Daily at 12:00 am**

Global Configuration

Bamboo will remove expired data based on the settings below. This global configuration can be overridden for individual plans by updating the Post Actions for a plan. Refer to the [documentation](#) for help configuring the settings below.

- ☐ **Disable Global Build Expiry**
Check this box to prevent the automatic deletion of build results from Bamboo.

What should be expired?

- ☐ **Build results**
The entire result will be removed (including artifacts)
- ☒ **Artifacts**
User defined artifacts will be expired
- ☒ **Build logs**
Build log will be expired

Expiry criteria

Expiry period weeks
Expire builds that completed before the above time period. Use 0 to ignore this option.

Minimum builds to keep
Minimum number of builds per plan to keep (e.g. 1 will keep the last build only), the rest will expire. Use 0 to ignore this option.

- ☒ **Don't expire builds with certain labels**

Labels to keep
The labels (separated by spaces) that you don't want to expire.

Importing data from Jenkins

The Jenkins Importer helps you to migrate projects deployed via the Jenkins continuous integration tool to Bamboo. On this page:

- [Requirements & supported configurations](#)
- [Using the Jenkins importer](#)
- [Getting Help](#)

Related pages:

- [Getting started with Java and Bamboo](#)
- [Getting started with .NET and Bamboo](#)
- [Using Bamboo](#)
- [Installing and upgrading Bamboo](#)

Requirements & supported configurations

While the importer assists and supports the migration of projects from Jenkins to Bamboo, a small amount of manual configuration may also be required.

Jenkins Version	Required Dependencies*	Jenkins Project Types	Repository Types	Build Steps	Notifications	Parameter Types	Other plugins

✓ Jenkins 1.494 and later	✓ Jenkins built-in job dependencies ✓ Pipelines plugin 1.3.3 ✓ Join plugin 1.15	✓ Freestyle ✓ Maven ✗ Matrix	✓ Git 1.1.25 ✓ SVN 1.43 ✗ Mercurial ✗ CVS ✗ Perforce	✓ Maven ✓ Ant 1.1 ✓ Script	✓ Email	✓ Boolean ✓ Choice ✓ String ✓ Password	✓ EnvInject plugin 1.73 ✓ JavaDoc 1.0 ✓ Wall Display plugin



The Required Dependencies are the functional Jenkins plugins that the Bamboo Jenkins Importer requires to be present. These must be present in your Jenkins instance.

Using the Jenkins importer

1. Start the importer



You must have administration privileges to run the Jenkins Importer

The Jenkins Importer is accessed from the welcome screen or Administration panel. To start the Jenkins Importer:

If running Bamboo for the first time

From the Bamboo Welcome screen:

1. Click **Import from...**
2. Select **Import from Jenkins.**

If already running Bamboo

From anywhere within the Bamboo interface:

1. Click on the **Administration** tab at the top of the Bamboo interface
2. Scroll down to the **System** side panel
3. Click on **Import from Jenkins.**

The Locate Jenkins screen will appear.

2. Select Jenkins data for importing

You may either import Jenkins data from its home location on the Bamboo server, or you may import from a zipped archive of your Jenkins home:

Importing from the Jenkins home location

From the Locate Jenkins interface:

1. Click the **Source of Jenkins home** dropdown menu
2. Select **Location on the Bamboo server**
3. Enter the path to your Jenkins home directory in the text field
4. Click **Next.**



You must specify the path to your Jenkins **home** directory.

Importing from an archived Jenkins home

From anywhere within the Bamboo interface:

1. Click the **Source of Jenkins home** dropdown menu
2. Select **Upload a zip archive**
3. Click **Choose Files**. A file manager window will open. Use it to locate your zipped Jenkins home directory
4. Click **Next**.



When creating your Jenkins zip archive, you need to remove or exclude the **userContent** and **builds** directory for each job from the archive before zipping:

1. Make a copy of Jenkins home
2. Remove/exclude the userContent directory
3. Remove/exclude the builds directory
4. Zip the archive.



Bamboo Jenkins Importer supports only ZIP file archives. Other archive formats such as tar and tar.gz are not currently supported.

Once you have selected your Jenkins data and clicked **Next**, the Jenkins job and pipeline selector will open.

3. Configure Jenkins data for import

The Job and Pipeline selector screen allows you to select and configure which Jenkins import items you would like to import into Bamboo. Import items include Jenkins pipelines and jobs, and the importer will identify how many items were found for processing:

Choose Jenkins jobs and pipelines to import as Bamboo Plans

Bamboo has found **one pipeline** and **13 jobs** that can be imported from Jenkins.

Jenkins Pipelines

Importing a Jenkins pipeline converts it into a Bamboo Plan, and all associated jobs into Bamboo jobs inside the Plans.

Import Pipelines	Bamboo Plan Name	Enabled in Bamboo?
☒ Build Pipeline Dashboard View	Build Pipeline Dashboard View	☒ Enabled
Import Jobs		
Bamboo Job Name		
☒ DLINK_COMPILE	SPLAN_COMPILE	☒ Enabled
☒ DLINK_TEST	SPLAN_TEST	☒ Enabled

The Jenkins Importer processes both Jenkins pipelines and jobs, but handles each differently:

Jenkins Import Item	Description	Bamboo Equivalent	Jenkins Importer Process
Pipeline	A group of associated jobs linked using a Jenkins pipeline	Plan	• Creates a new Bamboo plan • Imports individual pipeline jobs as Bamboo jobs into the new plan
Job	A stand alone build job	Job	• Imports job into a Bamboo plan with a single associated job within the plan

Importing a Jenkins job

To import a Jenkins job:

1. Locate the relevant Jenkins Import item on the selector screen
2. Ensure the **Import Jobs** check box is checked
3. Enter a **Bamboo Job Name** (or leave to accept the default name from Jenkins)
4. Check the **Enabled in Bamboo?** check box. To disable an imported job, leave the box unchecked. Bamboo will not automatically run a disabled job.

By default, all jobs are selected for importing. To reject a Jenkins job for importation:

1. Locate the relevant Jenkins Import item on the selector screen
2. Uncheck the **Import Jobs** check box

Importing a Jenkins pipeline

To import a Jenkins pipeline, all of the associated jobs must be imported. The importing of individual jobs is described above.

Changing the imported Bamboo plan or job name

By default, the Jenkins Importer uses the Jenkins import item name as the default for the Bamboo name. To change the default name:

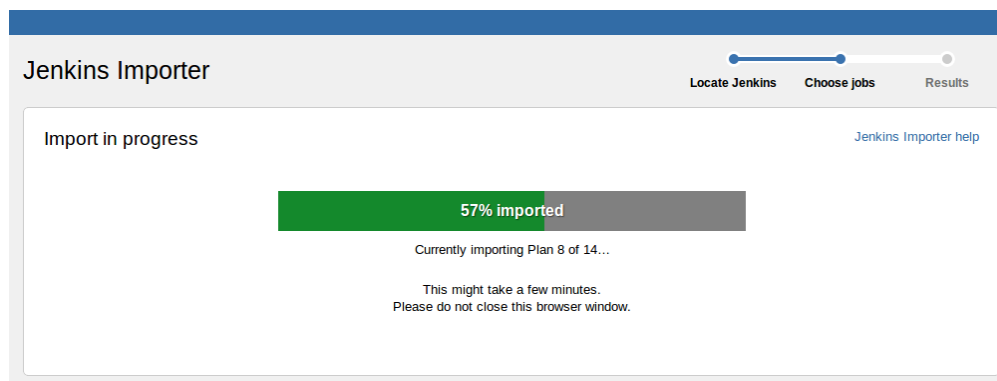
1. Locate the relevant Jenkins Import item on the selector screen
2. Enter a new name in the **Bamboo Plan Name** text field

Existing Bamboo plans

Existing Bamboo plans are not overwritten when you use the Jenkins Importer. When importing Jenkins data, Bamboo creates a new project called 'Imported from Jenkins' to contain all of the newly imported plans.

Starting the import

When you have identified and selected all of the Jenkins import items that you require, click **Next** at the bottom of the screen. Bamboo will start to import the specified plans and a progress indicator screen will display:



Once importing has finished, the importer results screen will display.

4. Review the importer results

The Importer Results screen shows the success outcome of the import activity for each import item. The three possible success outcomes are:

- Success
- Partial
- Fail

An example of Importer results is seen below.

SPLAN_DOCS	PARTIAL	Imported with warnings. See Import log for details.]
DLINK_CODE_CHECK	FAIL	Could not import due to fatal error. See Import log for details.
SPLAN_DEPLOY	SUCCESS	

Success

A '**success**' result indicates that the Jenkins item was successfully imported into Bamboo. No additional work is required.

Partial

A '**partial**' result indicates that the import was partially successful, however there may be unmapped configuration or other issues that require

attention.

Fail

A **'fail'** result indicates that the importer was unable to import the Jenkins job or pipeline if the repository type is unsupported, none of the build steps could be converted to tasks or another unknown error occurred.

The Import log

In the case of a fail or partial, additional information can be obtained from the Import log, which provides the following:

- The name of the attempted job import
- The severity if the problem/issue. Severity is rated as:
 - Low – warning that might be interesting to the administrator such as how dependencies were imported
 - Medium – unsupported publishers or other configurations that are non-critical to running the build
 - High – unsupported repository or none of the build steps could be imported
 - Fatal – un-handled error that prevents the job from importing at all
- A brief description of the problem/issue.

An example Import Log entry can be seen below:

```
-----  
-----  
Job name: SPLAN_DOCS  
SEVERITY: HIGH  
Unsupported configuration for plugin 'ClearCase  
UCM Plugin'  
-----  
-----
```

It is possible that an import item has multiple problems/issues. Where this is the case, the Import Log will identify the severity and brief description for each problem/issue associated with an import item. An example Import Log entry detailing multiple problems/issues can be seen below:

```
-----  
-----  
Job name: DLINK_CODE_CHECK  
SEVERITY: HIGH  
Unsupported configuration for plugin 'ClearCase  
Plugin'  
SEVERITY: HIGH  
Requested plugin parameterized-trigger but we  
don't support it  
SEVERITY: HIGH  
Requested plugin downstream-ext but we don't  
support it  
SEVERITY: FATAL  
Scm cannot be imported  
-----  
-----
```

The import log is accessed by clicking on the Import Log link associated with an import issue, or by clicking on the **Download import log file** button at the base of the Import Results screen.

Missing or incompatible plugins

From time to time, Bamboo may not support particular Jenkins functionality. When this occurs, a great place to look is the [Atlassian Marketplace](#). The Marketplace contains over 120 add-ons and plugins for Bamboo, and you will more than likely find a plugin for your functionality there. If you can't find what you need in the Marketplace, then consult the 'Getting help' section below.

5. View the imported plans

Once you have completed examining the importer results, click on **View Plans** to examine the imported plans in the Bamboo dashboard. An example of plans imported from Jenkins is seen below:

Plan	Build	Completed	Tests	Reason	
BSP_DEMO_SNAPSHOT_SONAR_BUILD	Never Built				
BSP_PARC_ALL_DASHBOARD_BUILD	#26	13 minutes ago	No tests found	Scheduled	
BSP_PARC_ARTIFACTORY_DASHBOARD_BUILD	Never built				
BSP_PARC_NIGHTLY_DASHBOARD_BUILD	#2	11 hours ago	No tests found	Scheduled	
BSP_PARC_RELEASE_BUILD_CLEARCASE_DASHBOARD_BUILD	Never built				
BSP_PARC_RELEASE_BUILD_GIT_DASHBOARD_BUILD	Never built				
Build Pipeline Dashboard View	#1	1 day ago	No tests found	First build for this plan	
PARC_RELEASE_BUILD_GIT	Never Built				
SCM_CONFIG_SAVE	Never built				

Imported plans can now be configured and managed using existing Bamboo methods.

Getting Help

Support

Help with the Jenkins importer is never far away. The best way to get help is to raise a support ticket directly via the Atlassian support site.

To create a support ticket:

1. Download the import log
2. Go to <https://support.atlassian.com> and select **Create New Issue**
3. Enter a detailed description of your problem within the support ticket
4. Attach the import log and lodge your support ticket
5. Wait to be notified of updates by Email

Security

As a distributed application, Bamboo's security is important. This page contains links to security-related information in the Bamboo documentation.

Security advisories

For information on how to report a security vulnerability in Bamboo and our policy on security advisories and patches, please read [Bamboo security advisories](#). A full list of security advisories that we have previously issued is also available on that page.

Bamboo permissions

For information on Bamboo's internal security model, i.e. user management and permissions, please see [Users and permissions](#).

Remote agent security considerations

Please note the following security implications when enabling remote agents for Bamboo:

- No encryption of data passed between server and agent — this includes data such as:
 - login credentials for version control repositories
 - build logs
 - build artifacts
- No authentication of the agent or server — this could result in unauthorised actions being taken on your system, such as:
 - Unauthorised parties installing new remote agents — version control repository login credentials could be stolen.
 - Unauthorised parties masquerading as a Bamboo server — the unauthorised server could pass malicious code to the agent to run.
 - See [Agent authentication](#) for more information.

We **strongly recommend that you do not enable remote agent installation** on any Bamboo instance accessible from a public or untrusted network. Creating remote agents is [Disabling and enabling remote agents support](#) by default.

Bamboo configuration

The following pages contain information on how to configure Bamboo features that can permit/forbid access to the Bamboo application.

- [Agent authentication](#)
- [Elastic Bamboo Security](#)
- [Bamboo cookies](#)
- [Best practices for Bamboo security](#)
- [Securing your remote agents](#)

Other security resources

-  [Configuring Tomcat to Use HttpOnly Session ID Cookies](#)
-  [Security](#)
-  [Securing your remote agents](#)
-  [Users and permissions](#)
-  [Securing your repository connection](#)
-  [Bamboo security advisories](#)
-  [Severity Levels for Security Issues](#)
-  [How to Report a Security Issue](#)
-  [Security Advisory Publishing Policy](#)
-  [Security Patch Policy](#)
-  [Elastic Bamboo Security](#)
-  [Configuring a plan's permissions](#)

Agent authentication

Bamboo provides a way to verify that remote agents are allowed to connect to the Bamboo server. This provides improved security for sensitive information in Bamboo.

- Bamboo prevents unknown remote agents from connecting to the Bamboo server.
- Remote agents need to be manually approved by an administrator before they can communicate with the Bamboo server in any way.

 Note that Elastic agents do not have to be approved.

On this page:

- [Authenticating remote agents](#)
- [Notes](#)

Related pages:

- [Bamboo remote agent installation guide](#)
- [Disabling and enabling remote agents support](#)
- [Configuring agents](#)

Authenticating remote agents

There are 2 aspects to the authentication of remote agents. Both of these are actioned in the administration 'Agents' screen.

To go there, click **Administration** and then **Agents** (under 'Build Resources').

1. Enable remote agent authentication on the Bamboo server

To do this, click **Enable Remote Agent Authentication**, and then **Confirm**.

2. Approve access for a particular remote agent

To do this, click on the **Agent Authentication** tab (under 'Remote Agents').

See [Bamboo remote agent installation guide](#) for details about installing a remote agent.

Screenshot: Approving access for a remote agent

Online Remote Agents
Offline Remote Agents
Agent Authentication

Below is a list of IP addresses and corresponding unique IDs. You may approve agents pending approval or revoke approval from previously approved agents.

Select: All, None, Waiting, Approved Action: Approve Access Revoke Access

Agent IP	Agent Unique ID (UUID)	Status	Operations
☐ 172.20.7.206	d549e866-faec-4787-8b17-3773e49cf3dc	Waiting	Approve Access

Notes

- If the agent's IP address changes, perhaps because DHCP is being used, then you will have to reapprove the agent when it next tries to connect using that different IP address.
- If you revoke access for a connected agent, the agent will remain connected and will continue to run. However, if the agent is subsequently restarted, it will not be able to connect.
- If you enable remote agent authentication, having previously revoked access for connected agents and disabled remote agent authentication, then you get the option to approve access for all connected agents at once. If you don't approve this, the agents stay connected and continue to run, but you will need to manually approve them when they next try to connect.

Elastic Bamboo Security

Elastic Bamboo is a feature in Bamboo that allows Bamboo to dynamically source computing resources from the Amazon Elastic Compute Cloud (EC2).

All traffic sent between the agents located in EC2 and the Bamboo server is tunnelled through an SSL-encrypted tunnel. The tunnel will be initiated from the Bamboo Server to the EC2 instance, which means that you don't need to allow any inbound connections to your server. You will need to permit outbound traffic from the server on the tunnel port, however - the default port number is 26224. On the EC2 instance, only the tunnel port needs to be open for inbound traffic.

SSL tunneling is not implemented for VCS (Version Control System) to EC2 traffic though. You will need to make your VCS available for access from EC2 to use Elastic Bamboo. Please see the section on setting up your VCS for Elastic Bamboo, which contains guidelines on securing your VCS.

Please be warned that just as with a regular host accessible from the Internet, if one of your remote agent instances is compromised, your Bamboo installation may be exposed to number of security vulnerabilities. These include confidential data (e.g. source code, VCS credentials) being stolen, malicious code being injected into elastic agents, unauthorised access to build queues and false information being submitted to Bamboo servers. Given that all Bamboo-related traffic is sent through a single encrypted connection, the risk of that happening is not high and can be further mitigated by setting up a VPC (Amazon Virtual Private Cloud). In a VPC, your elastic instances typically have no public IPs which means they are inaccessible from the internet other than through a regular, industry-standard VPN connection.

The sections below explain the default access rules for remote agent instances and how to change these rules, if desired.

On this page:

- [Default EC2 Access Rules](#)
- [Changing the Default EC2 Access Rules](#)
- [Using VPCs with Elastic Bamboo](#)
- [Setting up your Version Control System \(VCS\) for Elastic Bamboo](#)

Related pages:

- [Configuring Elastic Bamboo](#)

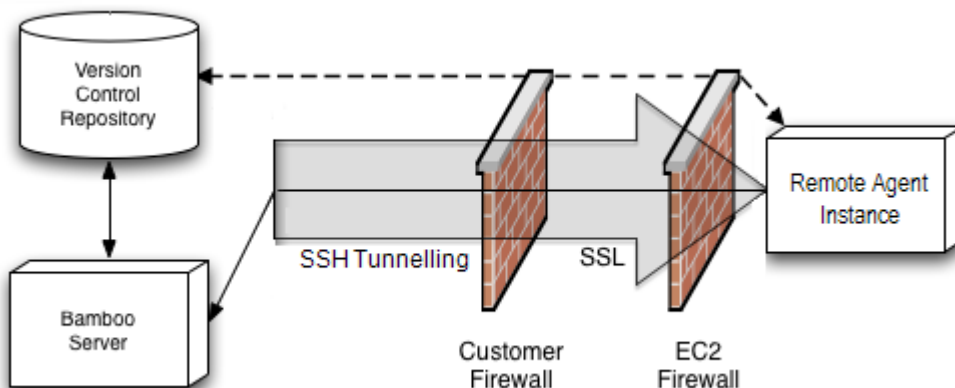
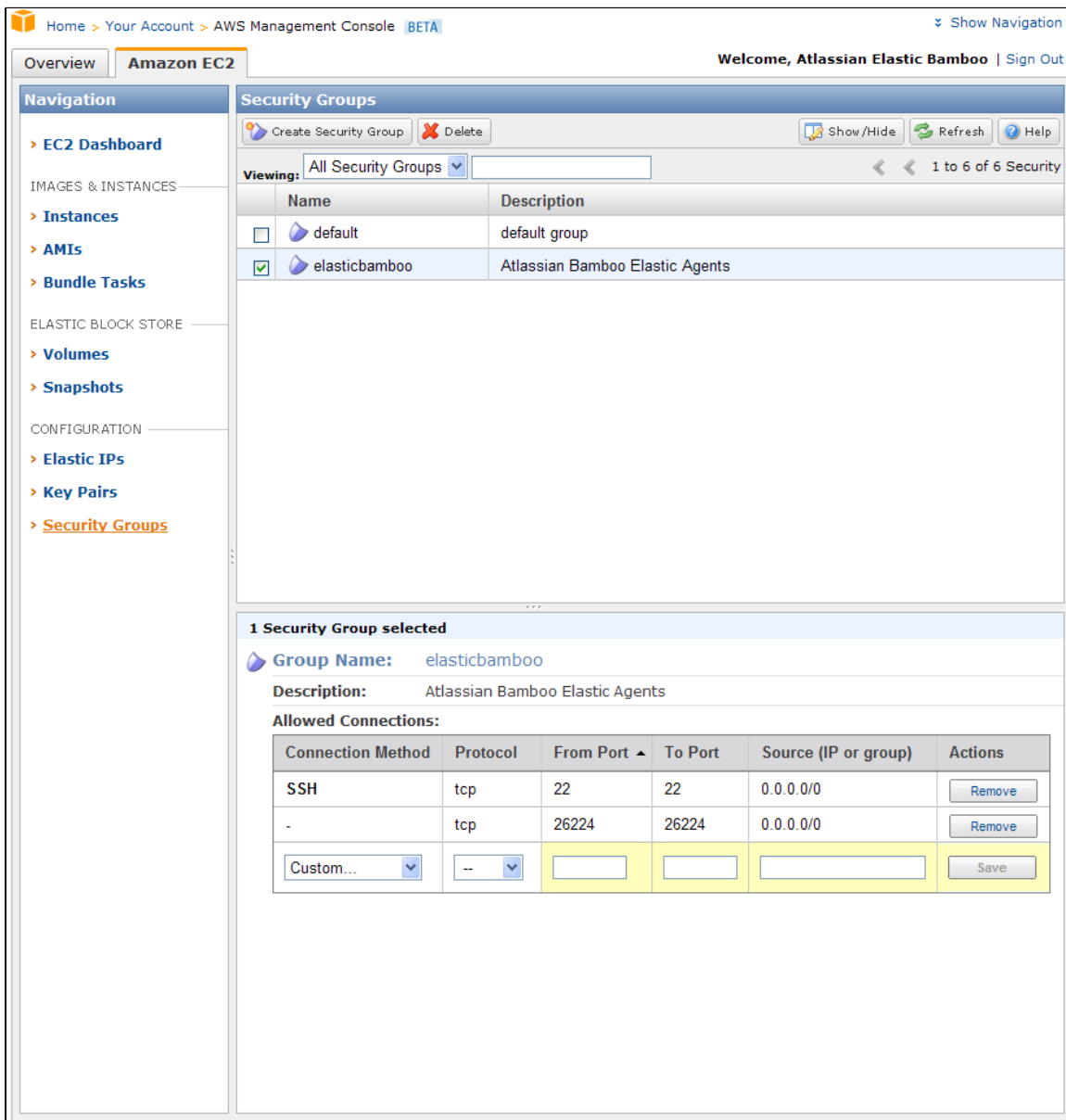


Diagram above: Elastic Bamboo security architecture

Default EC2 Access Rules

When you first use Elastic Bamboo, i.e. start an elastic instance, an 'elasticbamboo' security group will be set up for you on your AWS account. This security group is essentially a set of IP addresses that are permitted access to the EC2. By default, the security group will contain two rules — one to allow connections for Elastic Bamboo itself, and another to allow connections via SSH.

The EC2 security groups can be accessed via the [AWS management console](#) (see 'Security Groups' in the left-hand menu under 'Configuration').



The screenshot shows the AWS Management Console interface. The left-hand navigation menu is open, showing the 'Security Groups' link under the 'CONFIGURATION' section. The main content area displays the 'Security Groups' page. At the top, there are buttons for 'Create Security Group', 'Delete', 'Show/Hide', 'Refresh', and 'Help'. Below this, a table lists the security groups. The 'elasticbamboo' group is selected, and its details are shown below the table. The details include the group name 'elasticbamboo', the description 'Atlassian Bamboo Elastic Agents', and a table of 'Allowed Connections'.

Connection Method	Protocol	From Port	To Port	Source (IP or group)	Actions
SSH	tcp	22	22	0.0.0.0/0	Remove
-	tcp	26224	26224	0.0.0.0/0	Remove
Custom...	--				Save

Screenshot above: AWS Console - Security Groups

Changing the Default EC2 Access Rules

If you wish to permit additional connections to your EC2 instance, you can do this by adding entries to the 'Allowed Connections' section for the 'elasticbamboo' security group. See the previous section on 'Default EC2 Access Rules' for instructions on how to access your EC2 security groups.

Using VPCs with Elastic Bamboo

VPC functionality is available with Bamboo 4.3. Amazon Virtual Private Cloud (Amazon VPC) lets you provision a private, isolated section of the Amazon Web Services (AWS) Cloud where you can launch AWS resources in a virtual network. By default, the instances running in that network will have no public IPs and will not be accessible to the computers outside of your VPC. You can also create a Hardware Virtual Private Network (VPN) connection between your company datacenter and your VPC and leverage the AWS cloud as an extension of your company datacenter. You can read more about VPCs on [Amazon Web Services VPC page](#).

Using a VPC means that your agents (and other instances launched in the VPC) will not be available on the Internet. There are several basic scenarios that can be realised using a VPC:

- Secure access to your company datacenter - agents can securely access resources from your internal network through a VPN

connection. In this way, you can safely use your Version Control System or other internal resources such as databases from your Elastic Agents - without making them publicly accessible.

- Hiding some EC2 instances from the Internet - agents can communicate with your other hosts on the VPC using the internal network. This lets you e.g. set up an agent with a Windows-based DBMS and another one that runs tests against that DBMS from a different platform. Computers from outside of the VPC will not be able to access the DBMS because it will have no external IP. You don't need to use VPN for that use case, it's enough to assign an Elastic IP to the agent.
- Full-cloud deployment - you can host your Bamboo server in an Amazon's VPC and hide all your agents in a VPC. This will also let you access your other resources located in a VPC. The Bamboo Server can be accessed using VPN or an Elastic IP.

Setting up your Version Control System (VCS) for Elastic Bamboo

We recommend that you take the following steps to ensure that your Version Control System is set up securely for Elastic Bamboo:

1. [Make your Version Control System accessible to the public internet](#)
2. [Use VCS authentication and access control](#)
3. [Use encrypted connections to VCS](#)

1. Make your Version Control System accessible to the public internet



You only need to do this if you are not using a VPC for agent connectivity. See [using Bamboo with VPCs](#) for more information.

As SSL tunnelling is not implemented for VCS to EC2 connections, you will need to make your VCS accessible to the public internet to use Elastic Bamboo. If your VCS is behind a firewall this will involve configuring an access point in your firewall. Please consult the documentation for your firewall software for details on how to do this.

2. Use VCS authentication and access control

We highly recommend that you secure access to your VCS by enabling the authentication and access control features on your VCS. Please consult the documentation for your VCS for details.

3. Use encrypted connections to VCS

We also highly recommend that you use encrypted connections for your VCS (e.g. SSL). Please consult the documentation for your VCS for details.

Bamboo cookies

Bamboo uses [Seraph](#), an open source framework, for HTTP cookie authentication.

Authentication cookies

Bamboo uses two cookies:

- The JSESSIONID cookie is created by the application server and used for session tracking purposes.
- The 'remember me' cookie, `seraph.bamboo`, is generated by Bamboo when the user selects the **Remember me** checkbox on the login page.

 You can read about cookies on the [Wikipedia page](#).

On this page:

- [Authentication cookies](#)
- [The 'Remember Me' cookie](#)
 - [Cookie key and value](#)
 - [Use of cookie for authentication](#)
 - [Life of 'Remember Me' cookies](#)
- [Other cookie usage](#)

The 'Remember Me' cookie

The 'remember me' cookie is a long-lived HTTP cookie. This cookie can be used to authenticate an unauthenticated session. Bamboo generates this cookie when the user selects the **Remember me** checkbox on the login page.

Cookie key and value

By default, the cookie key is `seraph.bamboo`. This key is defined in the `BAMBOO-INSTALLATION/webapp/WEB-INF/classes/seraph-config.xml` file, in the `login.cookie.key` parameter.

The cookie contains a unique identifier plus a securely-generated random string.

Use of cookie for authentication

When a user requests a web page, if the request is not already authenticated via session-based authentication or otherwise, Bamboo will

match the 'remember me' cookie (if present) against the token stored for the user in the Bamboo database (if present).

If the random string matches the value stored in the database and the cookie has not expired, the user is authenticated.

Life of 'Remember Me' cookies

You can configure the maximum age of the cookie. To do that you will need to modify the `BAMBOO-INSTALLATION/webapp/WEB-INF/classes/seraph-config.xml` file and insert the following lines below the other `init-param` elements:

```
<init-param>
    <param-name>autologin.cookie.age</param-name>
    <param-value>2592000</param-value><!-- 30 days in seconds -->
</init-param>
```

Other cookie usage

There are several cookies in Bamboo that are used for storing basic presentation states, such as the number of log lines to show, which tab was previously selected etc. They are:

Cookie	Purpose
AJS.conglomerate.cookie	Track which general tabs are open and closed
BAMBOO-AGENT-FILTER	Date range to show the builds for agents
BAMBOO-BUILD-FILTER	Date range to show the builds
BAMBOO-LOG-REFRESH	Log refresh interval in seconds
BAMBOO-MAX-DISPLAY-LINES	Maximum # of lines to show on the live logs page
atlassian.bamboo.dashboard.tab.selected	Which tab is selected on the dashboard
bamboo.author.view	Which tab is selected on the Authors tab
bamboo.build.groupby.type	Which time group-by period is used in the reports
bamboo.dash.display.toggle	The ids of the projects that are expanded on the dashboard

Best practices for Bamboo security

The best way to harden a system is to look at each of the involved systems individually. Contact your company's security officer or department to find out what security policies you should be using. There are many things to consider, such as the configuration of your underlying operating systems, application servers, database servers, network, firewall, routers, etc. It would be impossible to outline all of them here.

This page contains guidelines on good security practices, to the best of our knowledge.

On this page:

- [Configuring the web server](#)
- [Configuring the application server](#)
- [Configuring the application](#)
- [Configuring system admin access](#)
- [Further precautions](#)

Configuring the web server

Please refer to the following guides for system administrators:

- How to configure Apache to lock down the administration interface to those people who really need it. See [Using Apache to limit access to the Confluence administration interface](#) for guidance.
- How to reduce the risk of brute force attacks: [Enabling or Disabling Captcha for Failed Logins](#).

Configuring the application server

See the following system administrator guide for general hints on the application server level:

- [Tomcat security best practices](#)

Configuring the application

The way you set up Bamboo roles, permissions and processes makes a big difference in the security of your Bamboo site.

Below are some more Bamboo-specific items to consider. None of these provides 100% security. They are measures to reduce impact and to slow down an intruder in case your system does become compromised.

- Restrict the number of users with powerful roles or group memberships. If only one department should have access to particularly sensitive data, then do restrict access to the data to those users. Do not let convenience over-rule security. Do not give all staff access to sensitive data when there is no need.
- Put documented procedures in place for the case of employees leaving the company.
- Perform security audits regularly. Know who can help in case a security breach occurs. Perform 'what if' planning exercises. ('What is the worst thing that could happen if a privileged user's password were stolen while he's on vacation? What can we do to minimise damage?').
- Make sure the Bamboo database user (and all datasource database users) only has the amount of database privileges it really needs.
- Monitor your binaries. If an attacker compromises an account on your system, he will usually try to gain access to more accounts. This is sometimes done by adding malicious code, such as by modifying files on the system. Run routine scripts that regularly verify that no malicious change has been made.

Configuring system admin access

Below are some things to consider specifically related to the system admin role:

- Keep the number of Bamboo administrators extremely low. For example, 3 system administrator accounts should be the maximum.
- The administrators should have separate Bamboo accounts for their administrative roles and for their day to day roles. If John Doe is an administrator, he should have a regular user account without administrator access to do his day to day work (such as configuring build plans). This could be a 'john.doe' account. In addition, he should have an entirely separate account (that cannot be guessed by an outsider and that does not even use his proper name) for administrative work. This account could be 'jane smith' – using a username that is so obscure or fake that no outsider could guess it. This way, even if an attacker singles out the actual person John Doe and gets hold of his password, the stolen account would most likely be John's regular user account, and the attacker cannot perform administrative actions with that account.
- Lock down administrative actions as much as you can. If there is no need for your administrators to perform administrative actions from outside the office, then lock down access to those actions to known IP addresses, for example. See [Using Apache to limit access to the Confluence administration interface](#) for guidance.

Further precautions

As another precaution:

- Regularly monitor the above requirements. There are many things that could start out well, but deteriorate over time:
 - A system may start out with just 3 administrators, but over the course of a year this could grow to 30 administrators if no one prevents expansion.
 - Apache administration restrictions may be in place at the start of the year, but when the application server is migrated after a few months, people may forget to apply the rules to the new system.

Again, keep in mind that the above steps may only be a fraction of what could apply to you, depending on your security requirements. Also, keep in mind that none of the above rules can guarantee anything. They just make it harder for an intruder to move quickly.

Securing your remote agents



This page applies to remote agents and not elastic agents. Elastic agents are secured automatically by the Bamboo server and no additional steps are required.

We **strongly recommend** that you do not enable remote agent installation without securing them on any Bamboo instance accessible from a public or untrusted network. Creating remote agents is [disabled](#) by default. If you choose to enable your remote agents without securing them, please read [this Security Advisory](#) to understand the security implications.

You can secure your remote agents by configuring them to use SSL (Secure Sockets Layer). This protocol provides a secure mechanism for communication between your Bamboo server and remote agents. The information below describes how to configure your remote agents to use SSL.

On this page:

- [Step 1. Create keys, stores and certificates](#)
- [Step 2. Tell your Bamboo server and agents where to find the stores](#)
- [Step 3. Configure your Bamboo server to use SSL](#)

Related pages:

- [Security](#)
- [Agent authentication](#)
- [Bamboo remote agent installation guide](#)
- [Disabling and enabling remote agents support](#)
- [Configuring agents](#)

Step 1. Create keys, stores and certificates

The first step in configuring your remote agents to use SSL is to create the required keys, stores and certificates. These artefacts are created using a keytool, as described below:



SSL relies on keys being set up on your server and clients (i.e. agents). To securely store these keys, keystores (databases of keys) need to be created. A certificate is then created by the server (and optionally on the clients, but not for this configuration) to allow publication of the server's key. To establish that the client "trusts" the server, this server certificate is then imported into a truststore (key database file that contains the public keys for a specific server) created on the client.

To create the required keys, stores and certificates for your server and agents:

1. Using a keytool, create a certificate for your server by entering the following command:

```
keytool -genkey -alias server -keyalg RSA -keystore server.ks
```

2. The server's certificate will be created. Export the certificate, so it can be shared with clients, by entering the following command:

```
keytool -export -alias server -keystore server.ks -file server_cert
```

3. Each client should now be able to access the server's certificate. Create a keystore for each client, by entering the following command:

```
keytool -genkey -alias client -keyalg RSA -keystore client.ks
```

4. Create a truststore for each client and import the server's certificate, by entering the command below. This establishes that the client "trusts" the server:

```
keytool -import -alias server -keystore client.ts -file server_cert
```

Step 2. Tell your Bamboo server and agents where to find the stores

The second step in configuring your agents to use SSL is to instruct your Bamboo server and agents to use the keystores and truststores that you have just created.

To tell your server where to find the keystore:

1. Add the system properties 'javax.net.ssl.keyStore=/path/to/server.ks' and 'javax.net.ssl.keyStorePassword=password' to your VM, by carrying out any of the following three steps:

- (Bamboo or Bamboo EAR-WAR) Set the **SSL_OPTS** environment variable to hold the 'javax.net.ssl.keyStore=/path/to/server.ks' and 'javax.net.ssl.keyStorePassword=password' properties. e.g.

```
export SSL_OPTS = -Djavax.net.ssl.keyStore=/path/to/server.ks  
-Djavax.net.ssl.keyStorePassword=password
```

Or,

- (Bamboo only) Add 'javax.net.ssl.keyStore=/path/to/server.ks' and 'javax.net.ssl.keyStorePassword=password' as additional properties to the [wrapper.conf](#) file. e.g.

```
wrapper.java.additional.4=-Djavax.net.ssl.keyStore=/path/to/server.ks  
wrapper.java.additional.5=-Djavax.net.ssl.keyStorePassword=password
```

Or,

- (Bamboo EAR-WAR only) Make the 'javax.net.ssl.keyStore=/path/to/server.ks' and 'javax.net.ssl.keyStorePassword=password' properties visible to the VM, as per the instructions for your webserver.

To tell your agents where to find the keystore and truststore:

For each agent,

1. Tell your agent where to find the keystore and the trust store, by executing the following command to run the agent,

```
java -jar bamboo-agent-2.0-SNAPSHOT.jar <agentserverURL>
```

including the following command line parameters,

```
-Djavax.net.ssl.keyStore=/path/to/client.ks  
-Djavax.net.ssl.keyStorePassword=password  
-Djavax.net.ssl.trustStore=/path/to/client.ts
```

where <agentserverURL> is the URL of the agent's server, e.g.

```
http://192.168.3.235:8085/agentServer/
```

For example,

```
java -Djavax.net.ssl.keyStore=/path/to/client.ks  
-Djavax.net.ssl.keyStorePassword=password  
-Djavax.net.ssl.trustStore=/path/to/client.ts -jar bamboo-agent-2.0.jar  
http://192.168.3.235:8085/agentServer/
```

Step 3. Configure your Bamboo server to use SSL

Once the server and agents know where to find the keystores and truststores, the final step is to instruct your Bamboo server to start using SSL so that agents will be able to authenticate the server.

To configure your Bamboo server to use SSL:

If you are setting up Bamboo for the first time,

1. Launch the [Bamboo Setup Wizard](#) and change the protocol of the '**Broker URL**' to 'SSL'.
i.e. **ssl**://host:port/

Or, if you are configuring an existing installation of Bamboo,

1. Shut down your Bamboo server and agents.
2. Change the protocol of your '**Broker URL**' in the **bamboo.cfg.xml** file to 'SSL'. Note, do not change the address of this URL.
e.g. `<property name="bamboo.jms.broker.uri">ssl://myhost:myport?wireFormat.maxInactivityDuration=0</property>`
3. Start up the Bamboo server.
4. Start up the Bamboo agents. If your agents do not start up, please check that you have set up your certificates correctly.

Bamboo release notes

Latest Release



Bamboo 5.0 has been released. Read the [Bamboo 5.0 Release Notes](#) and [Upgrade Guide](#).
Don't have Bamboo 5.0? Take a look at the features of Bamboo's [latest major version](#) and try it out!

Release Summary

The features of each Bamboo release, up to and including the latest version, can be found in the [Bamboo release summary](#).

For full details on each of the Bamboo releases, please read the relevant release notes listed below:

- [Previous Production Releases](#)

- [Previous Beta Releases](#)

You may also be interested in the [Bamboo upgrade guides](#) for each release.

All Production Releases

-  [Bamboo 4.4.1 Release Notes](#)
-  [Bamboo 5.0 Release Notes](#)
-  [Bamboo 4.4 Release Notes](#)
-  [Bamboo 4.3.3 Release Notes](#)
-  [Bamboo 4.3.2 Release Notes](#)
-  [Bamboo 4.3.1 Release Notes](#)
-  [Bamboo 4.2.1 Release Notes](#)
-  [Bamboo 4.3 Release Notes](#)
-  [Bamboo 4.2 Release Notes](#)
-  [Bamboo 4.1.2 Release Notes](#)
-  [Bamboo 4.1.1 Release Notes](#)
-  [Bamboo 3.3.4 Release Notes](#)
-  [Bamboo 3.4.5 Release Notes](#)
-  [Bamboo 4.1 Release Notes](#)
-  [Bamboo 4.0.1 Release Notes](#)
-  [Bamboo 4.0 Release Notes](#)
-  [Bamboo 3.4.4 Release Notes](#)
-  [Bamboo 3.4.3 Release Notes](#)
-  [Bamboo 3.4.2 Release Notes](#)
-  [Bamboo 3.4.1 Release Notes](#)
-  [Bamboo 3.4 Release Notes](#)
-  [Bamboo 3.3.3 Release Notes](#)
-  [Bamboo 3.3.2 Release Notes](#)
-  [Bamboo 3.3.1 Release Notes](#)
-  [Bamboo 3.3 Release Notes](#)
-  [Bamboo 3.2.2 Release Notes](#)
-  [Bamboo 3.2 Release Notes](#)
-  [Bamboo 3.1.4 Release Notes](#)
-  [Bamboo 3.1.3 Release Notes](#)
-  [Bamboo 3.1.1 Release Notes](#)
-  [Bamboo 3.1 Release Notes](#)
-  [Bamboo 3.0.3 Release Notes](#)
-  [Bamboo 3.0.2 Release Notes](#)
-  [Bamboo 3.0.1 Release Notes](#)
-  [Bamboo 2.7.4 Release Notes](#)
-  [Bamboo 3.0 Release Notes](#)
-  [Bamboo 2.7.3 Release Notes](#)
-  [Bamboo 2.7.1 Release Notes](#)
-  [Bamboo 2.7.2 Release Notes](#)
-  [Bamboo 2.6.3 Release Notes](#)
-  [Bamboo 2.6.2 Release Notes](#)
-  [Bamboo 2.7 Release Notes](#)
-  [Bamboo 2.6.1 Release Notes](#)
-  [Bamboo 2.5.5 Release Notes](#)
-  [Bamboo 2.5.3 Release Notes](#)
-  [Bamboo 2.5.2 Release Notes](#)

-  [Bamboo 2.6 Release Notes](#)
-  [Bamboo 2.5.1 Release Notes](#)
-  [Bamboo 2.4.3 Release Notes](#)
-  [Bamboo 2.5 Release Notes](#)
-  [Bamboo 2.4.2 Release Notes](#)
-  [Bamboo 2.4.1 Release Notes](#)
-  [Bamboo 2.4 Release Notes](#)
-  [Bamboo 2.3.1 Release Notes](#)
-  [Bamboo 2.3 Release Notes](#)
-  [Bamboo 2.2.4 Release Notes](#)
-  [Bamboo 2.2.3 Release Notes](#)
-  [Bamboo 2.2.2 Release Notes](#)
-  [Bamboo 2.2.1 Release Notes](#)
-  [Bamboo 2.1.5 Release Notes](#)
-  [Bamboo 2.1.4 Release Notes](#)
-  [Bamboo 2.2 Release Notes](#)
-  [Bamboo 2.1.3 Release Notes](#)
-  [Bamboo 2.1.2 Release Notes](#)
-  [Bamboo 2.1.1 Release Notes](#)
-  [Bamboo 2.1 Release Notes](#)
-  [Bamboo 2.0.6 Release Notes](#)
-  [Bamboo 2.0.5 Release Notes](#)
-  [Bamboo 2.0.4 Release Notes](#)
-  [Bamboo 2.0.3 Release Notes](#)
-  [Bamboo 2.0.2 Release Notes](#)
-  [Bamboo 2.0.1 Release Notes](#)
-  [Bamboo 1.2.4 Release Notes](#)
-  [Bamboo 1.2.3 Release Notes](#)
-  [Bamboo 2.0 Release Notes](#)
-  [Bamboo 1.2.2 Release Notes](#)
-  [Bamboo 1.2.1 Release Notes](#)
-  [Bamboo 1.2 Release Notes](#)
-  [Bamboo 1.1.2 Release Notes](#)
-  [Bamboo 1.1.1 Release Notes](#)
-  [Bamboo 1.1 Release Notes](#)
-  [Bamboo 1.0.5 Release Notes](#)
-  [Bamboo 1.0.4 Release Notes](#)
-  [Bamboo 1.0.3 Release Notes](#)
-  [Bamboo 1.0.2 Release Notes](#)
-  [Bamboo 1.0.1 Release Notes](#)
-  [Bamboo 1.0 Release Notes](#)

All Beta Releases

-  [Bamboo 2.0 Beta Release Notes](#)
-  [Bamboo 2.0 Beta 9 Release Notes](#)
-  [Bamboo 2.0 Beta 8 Release Notes](#)
-  [Bamboo 2.0 Beta 6 Release Notes](#)
-  [Bamboo 2.0 Beta 5 Release Notes](#)
-  [Bamboo 2.0 Beta 4 Release Notes](#)
-  [Bamboo 2.0 Beta 3 Release Notes](#)

 [Bamboo 2.0 Beta 2 Release Notes](#)

 [Bamboo 2.0 Beta 1 Release Notes](#)

 [Bamboo 1.0-Beta Release Notes](#)

Bamboo release summary

This page shows the highlights of the major Bamboo releases.

Current release

For information about the latest release of Bamboo, please check the main [Bamboo release notes](#) page.

Bamboo 5.0 — 15th July 2013

- Deployment projects
- JIRA versions and release candidate combining
- Release versioning
- Improved communications

Bamboo 4.4 — 29 January 2013

- Jenkins importer
- Enhanced performance
- TestNG reports parser
- Selective task running
- Quick build number copying
- Server pausing

Bamboo 4.3 — 30 October 2012

- Simple deployments to Tomcat
- Deploy to the Cloud with Heroku
- Upload files using SCP
- Build from any revision
- Runtime variables for Manual Stages
- Rebuild with one click
- Multiple build Triggers
- Build dependencies after all Stages
- Automatic dependencies for Maven 3
- Dashboard filtering
- Amazon Virtual Private Cloud for Elastic Bamboo
- Wallboard for Branches

Bamboo 4.2 — 21 August 2012

- Notifications, build strategies and dependencies for plan branches
- Automatic linking of JIRA issues to feature branches
- Bamboo build artifacts are linked from JIRA issues
- Git support in the Bamboo Bitbucket connector
- Subversion 1.7 support
- UX improvements
- Mercurial commit isolation

Bamboo 4.1 — 29 May 2012

- Get builds fixed faster with responsibilities
- Take action with JIRA issue creation
- Get your team communicating with Hipchat notifications
- Welcome to the family, Stash!
- More in [release notes](#)

Bamboo 4.0 — 27 March 2012

- Automatically Build Branches
- Automatically Merge Branches using Gatekeeper and Branch Updater
- Manage Build Failures by Quarantining Intermittent Tests
- Fresh New User Experience
- More in [release notes](#)

Bamboo 3.4 — 14 December 2011

- Git Submodule Support
- Shared Repositories

- Agent Security Improvements
- New Email Templates
- Elastic Bamboo support for Microsoft Windows®
- More in [release notes](#)

Bamboo 3.3 — 11 October 2011

- Multiple Source Repositories
- Reload-able Plugins
- Source Repository User Aliases
- Automatic Agent Upgrades
- Fast, history-friendly tabbed navigation
- Commit Centric View
- More in [release notes](#)

Bamboo 3.2 — 26 July 2011

- Release Management
- Manual Stages
- Rerunning a Failed Stage
- Plan Filters on the Dashboard and Wallboard
- User Management via JIRA
- Improved Application Linking
- More in [release notes](#)

Bamboo 3.1 — 10 May 2011

- Tasks
- Parameterised Builds
- .Net Support
- Bitbucket Support
- GitHub Support
- New Plugin Manager
- Support for Amazon EC2 Spot Instances
- Gravatar Support
- Improved Windows process handling
- More in [release notes](#)

Bamboo 3.0 — 16 February 2011

- Artifact Sharing
- Git Support
- User Interface Overhaul — Redesigned Plan Summary, Job Summary and Build Results. New look and feel.
- Scheduled Repository Polling
- Configuration Changes Captured in Audit Logs
- More in [release notes](#)

Bamboo 2.7 — 9 Nov 2010

- Build Stages — Map Your Build Process, Parallel Builds, Enhanced Plan Structure
- Simplified Plan Creation
- Concurrent Builds
- Mercurial Support
- Improved Wallboards
- New Plan and Job Configuration Summaries
- Recent History on Plan and Job Summaries
- Other User Interface Enhancements — New Breadcrumb Trail, Build Histories, Improved Build Result Summary Tabs
- More in [release notes](#)

Bamboo 2.6 — 1 June 2010

- Support for up to 100 Remote Agents
- Revamped Dashboard Pages and Other Usability Enhancements
- Performance and Security Improvements
- Automatically Managed Elastic Instances
- Grails Integration
- More in [release notes](#)

Bamboo 2.5 — 4 January 2010

- Maven Dependency Management
- Plan Import from a pom.xml
- Additional Bulk Actions
- Streamlined Plan Creation
- Express Setup Wizard

- More in [release notes](#)

Bamboo 2.4 — 6 October 2009

- Bamboo Gadgets in JIRA
- Clover Enhancements
- REST Improvements
- Runtime Log4j Configuration
- More in [release notes](#)

Bamboo 2.3 — 6 August 2009

- Dependency Blocking Strategies
- New Build Notifications and Queue Reordering
- Bulk Actions
- Multiple Elastic Images
- Elastic Instance Scheduling
- PHPUnit Builder
- Bamboo REST APIs
- Plugins Changes
- More in [release notes](#)

Bamboo 2.2 — 9 March 2009

- Elastic Bamboo
- Customisable Email Templates
- Build Comment Notification Event
- Hanging Build Detection Event
- Faster Artifact Transfer
- Dependent Builds
- Agent Improvements
- More in [release notes](#)

Bamboo 2.1 — 5 August 2008

- Link Issues and Builds
- Specify the Issues that are Fixed by a Build
- Track the Builds for your Projects and Versions
- View Issues under Development
- Post Change Detection Plugin Point
- More in [release notes](#)

Bamboo 2.0 — 14 April 2008

- Distributed builds
- Capability matching
- Memory usage improvements
- Parallel VCS updates and checkouts
- Ability to force a 'clean build'
- Quiet Period functionality supported for Subversion & Perforce
- Bamboo Plugin for Confluence
- Support for Oracle and MS SQL Server databases
- Status Summary screens
- More in [release notes](#)

Bamboo 1.2 — 09 July 2007

- Permissions (global and plan-based)
- External database support
- Perforce triggering support
- Scheduled backups
- New Bundled NAnt plugin
- More in [release notes](#)

Bamboo 1.1 — 07 May 2007

- Advanced notifications - rules, triggers, preferences, dynamic recipients
- Build Metadata - pass them to your build, global variables, view them
- File Trigger Inclusions/Exclusions
- More pluggability
- Improved Maven 2 error log parsing
- LDAP and external user management support
- Dashboard loading has been improved
- More in [release notes](#)

- All Plans tab
- More in [release notes](#)

Bamboo 5.0 beta 1 Release Notes

14th May 2013

Our first pre-release build of **Bamboo 5** is now available. The support for deployment orchestration in the Bamboo 5.x series is our most ambitious endeavor yet. And the feedback we get from you in these next few weeks is absolutely critical in making this the most useful and intuitive Bamboo ever.

What We Need From You

1. [Download your favourite distribution](#). Along the way you'll be asked for your email address. This tells us how many people are actually participating vs. how many indicated interest, which helps us plan for future beta programs, and ensures we have an open line of communication with each participant.
2. Learn about the benefits that [Deployment Projects](#), the foundation of Bamboo 5.0's new deployment capabilities, have to offer your team.
3. Install Bamboo 5.0 and start exploring! You can set up a test instance and import a copy of your Bamboo data (If you need a new license, you can get a evaluation license from my.atlassian.com). We're especially interested in the setup process, so if you remember to note how that goes for you, we'd love to hear about it.
4. Look for an initial survey from us about 1-2 days after you download and install. 18 simple questions—should take five minutes or less.
5. As you're using the beta, relay your thoughts about things you like or dislike by clicking the "Feedback for Bamboo 5 Beta" button, found at the top of each page.
6. Look for a second survey after you've been using the beta for about a week. Once again, shouldn't take too long.

What We'll Provide in Return

1. Participants who step through both surveys will get an [awesome Bamboo t-shirt](#).
2. Our product manager and user-experience specialist will follow up personally with as many participants as possible to dig deeper into the aspects of Bamboo 5 that you don't love (yet).
3. The development team will prioritize their work between now and release time based on your feedback.
4. A final release of Bamboo 5.0 tailored for our users, by our users.
5. Our eternal gratitude n' stuff 😊

Download the beta

This list is by no means a complete list of changes that will be available in 5.0 and the scheduling of some items may change without notice.

Updates and fixes in Bamboo 4.4.1

Key	Summary	Status
BAM-10855	Convert ANSI color codes to HTML when displaying logs	 Resolved
BAM-10573	Pass artifacts between plans	 Resolved
BAM-10089	Improvements to EBS mounting	 Resolved
BAM-11104	Ability to skip manual stage	 Resolved
BAM-13065	Disabling Automatic Dependency management does not remove the generated dependencies.	 Resolved
BAM-13062	Please provide Job Name and Job Key as variables	 Resolved
BAM-12059	Working Directory help bubble gives page not found	 Resolved
BAM-11719	Anonymous users can view remote agent details	 Resolved
BAM-11620	JIRA Bamboo release plugin doesn't start build if plan has no repository	 Resolved
BAM-11602	Bamboo sometimes kills processes not belonging to it on Windows	 Resolved

BAM-12043	Unable to link svn repositories to viewVC	 Resolved
BAM-11899	Agent Log message handling improvements	 Resolved
BAM-11995	"Create Issue" button next to "Quarantine"	 Resolved
BAM-12115	Add new instance types to Elastic Bamboo	 Resolved
BAM-12128	Truncate Failure reason for a merge_result before persisting to the database	 Resolved
BAM-11359	Upgrade to Amazon Linux 2012.09.1	 Resolved
BAM-11342	Javascript error when pressing "Run" in Bamboo plan	 Resolved
BAM-12158	Git repository password displayed as plain text	 Resolved
BAM-12169	"Enable Commit Isolation" shows up twice	 Resolved
BAM-12208	"password" variable shows in metadata	 Resolved

 [Authenticate](#) to retrieve your issues

[83 more issues](#)

Bamboo 5.0 Release Notes

15th July 2013

The Atlassian Bamboo team are pleased to announce the release of **Bamboo 5.0**.

The cornerstone of Bamboo 5.0 is the inclusion of deployment projects, which allow you to automate the building, testing and deployment of your software products with the click of a mouse.

Learn more about the advantages of Bamboo 5.0 in the [Bamboo 5.0 release announcement blog](#).

We've also fixed over 87 bugs in this release. Please see the 'Updates and fixes in this release' section below for details.

Bamboo 5.0 is of course free to all customers with [active Bamboo software maintenance](#).

If you are upgrading, please read the [Bamboo 5.0 Upgrade Guide](#).



Improvements and new features

We prioritised the improvements and features voted for by you, including:

- Bamboo 5.0 programatically combines JIRA versions and release candidate builds of your application for better tracking from the planning phase through to development and release
- Bamboo's release versioning provides a unique identifier for the build that's being deployed: know exactly which release version is running in which environment, and see which issues informed that release
- Improved communication helps tame the beasts known as software releases. Commenting on release candidates, broken or approved tagging, and other Bamboo 5 features make improved cross-team communication a reality

Improvements and new features in Bamboo 5.0

Type	Key	Summary
	BAM-9613	Have stage permissions
	BAM-10573	Pass artifacts between plans

	BAM-9515	Bamboo should support all AWS regions
	BAM-9611	Allow to rerun successful stages
	BAM-1850	Jira issues fixed in a range of builds
	BAM-5527	Remove build-number.txt
	BAM-5930	Option to specify ebs volume instead of snapshot - keep volume alive after elastic instance shutdown
	BAM-11104	Ability to skip manual stage
	BAM-10089	Improvements to EBS mounting
	BAM-11995	"Create Issue" button next to "Quarantine"
	BAM-12115	Add new instance types to Elastic Bamboo
	BAM-11899	Agent Log message handling improvements
	BAM-10855	Convert ANSI color codes to HTML when displaying logs
	BAM-11359	Upgrade to Amazon Linux 2012.09.1
	BAM-9261	Make Maven Artifact Sharing non beta
	BAM-12589	Report test status across branches
	BAM-12807	Update Grails to 2.2.0 on Ec2 images
	BAM-12664	Elastic Bamboo doesn't mount EBS snapshot volumes for Windows AMIs
	BAM-12849	Add Node.js to Elastic Bamboo
	BAM-13360	Rename a deployment release

 [Authenticate](#) to retrieve your issues

23 more issues

Updates and fixes in Bamboo 5.0

Type	Key	Summary
	BAM-13428	Please update Upgrade Guide for Bamboo 5.0 on CAC
	BAM-13419	Reorder environments on the deployments and project pages
	BAM-13366	Pattern Match Labelling not working...
	BAM-13360	Rename a deployment release
	BAM-13354	Don't show the plans that a user doesn't have permission to view in the agent activity list
	BAM-13319	After switching to Bamboo 5.0-beta3 exports don't work anymore

	BAM-13314	Artifact Downloader Task broken in Bamboo 5 beta2
	BAM-13306	java.io.UTFDataFormatException: encoded string too long: 85338 bytes
	BAM-13291	5.0-beta2 tries to do AWS setup even if there are no AWS account details
	BAM-13288	Broken link to "instant messaging server" in the Bamboo Administration home page
	BAM-13222	Use an absolute path for Clover history dir for automatic Clover integration
	BAM-13204	Changeset ids are sometimes not shown on result page
	BAM-13199	Correct the wordings of war file location configuration in the tomcat deployer
	BAM-13187	NullPointerException when retrieving files from CommitContext
	BAM-13163	"Change Subversion branches URL" checkbox gets unchecked after editing another tab
	BAM-13161	5 Minor JavaDoc Typos
	BAM-13104	Bamboo is unable to start EC2 instances when custom inbound security rules contain wildcard ports
	BAM-13062	Please provide Job Name and Job Key as variables
	BAM-13022	Show "System Errors" tab in an Agent page
	BAM-13018	False error when re-creating a deleted plan

 [Authenticate](#) to retrieve your issues

77 more issues

Who are the Bamboo team?

Product management

Mark Chaimungkalanont

James Dumay

Agile evangelism

Sarah Goff-Dupont

Development

Brydie McCoy

David Hernandez

Piotr Stefaniak

Krystian Brazulewicz

Marcin Gardias

Marek Went

Przemek Bruski
UI Development

Matthias Schreck

Jason Berry

Documentation

Nathan Pye

Paul Watson

Bamboo 5.0.1 Release Notes

29th July 2013

The Atlassian Bamboo team are proud to announce the release of **Bamboo 5.0.1**.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 5.0.1 is of course free to all customers with [active Bamboo software maintenance](#).

If you are upgrading, please read the [Bamboo 5.0 Upgrade Guide](#).



Notable changes

Test parsing tasks have been changed so that its possible to pick up test results that were created out side of the build by using an advanced option.

Test parsers that support this advanced option are:

- JUnit
- TestNG
- MSTest
- NUnit
- MBUnit
- PHPUnit

10 elastic images have th

Source Code Checkout
Checkout Default Repository

Maven 2.x

JUnit Parser

Final tasks Are always executed even if a previous task fails

Drag tasks here to make them final

Add task

JUnit Parser configuration

Task description

☐ Disable this task

Specify custom results directories

**test-reports/*.xml

Where does the build place generated test results?
this is a comma separated list of test result directories. You can also use Ant style patterns such as **/test-reports/*.xml

Advanced options

☐ Pick up test results that were created outside of this build

Files created before the current build was started will be analyzed as valid tests results

Save Cancel

Key	Summa ry	Type	Created	Update d	Due	Assign ee	Reporte r	Priority	Status	Resolut ion
BAM-12768	Inconsis tent		Feb 05, 2013	Aug 26, 2013		Unassign ed	Armen Khachat		 Res olved	Fixed

	behavior when using Junit parser task						ryan [Atlassian]			
BAM-13437	Status icon does not update on plan completion		Jul 16, 2013	Aug 07, 2013	Unassigned	Matthew Watson [Atlassian]		 Resolved	Fixed	
BAM-13312	Artifact Downloader task in dependent build fails to download artifacts from parent of its parent build when triggered by parent plan (but manual build is OK)		Jun 13, 2013	Aug 23, 2013	Unassigned	Anton Tretyak		 Resolved	Fixed	
BAM-13159	Mask the password "Value" when overriding global variables in a branch		Apr 24, 2013	Jul 28, 2013	Unassigned	Sepideh Setayeshfar [Atlassian]		 Resolved	Fixed	
BAM-9113	Hide password in build variables		Jun 24, 2011	Sep 05, 2013	Marek Went [Atlassian]	Stephan Oudmaijer		 Resolved	Fixed	
BAM-13519	Access denied for plan admin on the permission tab		Jul 25, 2013	Jul 31, 2013	Unassigned	Sepideh Setayeshfar [Atlassian]		 Resolved	Fixed	
BAM-13468	Running deployment shows as "Deployed" on build		Jul 17, 2013	Jul 28, 2013	Unassigned	Brydie McCoy [Atlassian]		 Resolved	Fixed	

	result summary									
BAM-11623	It is possible to edit branch job configuration.		May 27, 2012	Jul 12, 2013	Unassigned	Marcin Gardias [Atlassian]	↓	 Resolved	Fixed	
BAM-13486	Github repository cannot be used in deployment projects		Jul 18, 2013	Jul 28, 2013	Unassigned	James Dumay [Atlassian]	↓	 Resolved	Fixed	
BAM-13494	Bamboo live log is not generated until the build is complete		Jul 19, 2013	Jul 25, 2013	Unassigned	Sepideh Setayeshfar [Atlassian]	↓	 Resolved	Fixed	
BAM-13430	No page refresh inlined comment strips new lines		Jul 15, 2013	Jul 28, 2013	Unassigned	Ian Grunert [Atlassian]	↓	 Resolved	Fixed	
BAM-13384	system.git.executable fails to detect git in PATH or cygwin		Jul 04, 2013	Jul 26, 2013	Unassigned	Sorin Sbârnea	↓	 Resolved	Fixed	
BAM-13476	Can falsely setup Deployment Project for plan branches		Jul 17, 2013	Jul 31, 2013	Unassigned	Jason Monsorino	↑	 Resolved	Fixed	
BAM-13469	Download Artifact task does not seem to substitute variables		Jul 17, 2013	Jul 28, 2013	Unassigned	Nick Pellow [Atlassian]	↓	 Resolved	Fixed	
BAM-13363	Update Bamboo Clover Plugin		Jun 28, 2013	Jul 28, 2013	Marek Parfiano	Marek Parfiano	↓	 Resolved	Fixed	

to
version
3.1.12

wicz
[Atlassi
an]

wicz
[Atlassia
n]



Authenticate to retrieve your issues

Bamboo 5.0 Release Notes

15th July 2013

The Atlassian Bamboo team are pleased to announce the release of **Bamboo 5.0**.

The cornerstone of Bamboo 5.0 is the inclusion of deployment projects, which allow you to automate the building, testing and deployment of your software products with the click of a mouse.

Learn more about the advantages of Bamboo 5.0 in the [Bamboo 5.0 release announcement blog](#).

We've also fixed over 87 bugs in this release. Please see the 'Updates and fixes in this release' section below for details.

Bamboo 5.0 is of course free to all customers with [active Bamboo software maintenance](#).

If you are upgrading, please read the [Bamboo 5.0 Upgrade Guide](#).



Improvements and new features

We prioritised the improvements and features voted for by you, including:

- Bamboo 5.0 programatically combines JIRA versions and release candidate builds of your application for better tracking from the planning phase through to development and release
- Bamboo's release versioning provides a unique identifier for the build that's being deployed: know exactly which release version is running in which environment, and see which issues informed that release
- Improved communication helps tame the beasts known as software releases. Commenting on release candidates, broken or approved tagging, and other Bamboo 5 features make improved cross-team communication a reality

Improvements and new features in Bamboo 5.0

Type	Key	Summary
	BAM-9613	Have stage permissions
	BAM-10573	Pass artifacts between plans
	BAM-9515	Bamboo should support all AWS regions
	BAM-9611	Allow to rerun successful stages
	BAM-1850	Jira issues fixed in a range of builds
	BAM-5527	Remove build-number.txt
	BAM-5930	Option to specify ebs volume instead of snapshot - keep volume alive after elastic instance shutdown
	BAM-11104	Ability to skip manual stage
	BAM-10089	Improvements to EBS mounting
	BAM-11995	"Create Issue" button next to "Quarantine"
	BAM-12115	Add new instance types to Elastic Bamboo

	BAM-11899	Agent Log message handling improvements
	BAM-10855	Convert ANSI color codes to HTML when displaying logs
	BAM-11359	Upgrade to Amazon Linux 2012.09.1
	BAM-9261	Make Maven Artifact Sharing non beta
	BAM-12589	Report test status across branches
	BAM-12807	Update Grails to 2.2.0 on Ec2 images
	BAM-12664	Elastic Bamboo doesn't mount EBS snapshot volumes for Windows AMIs
	BAM-12849	Add Node.js to Elastic Bamboo
	BAM-13360	Rename a deployment release

 Authenticate to retrieve your issues

23 more issues

Updates and fixes in Bamboo 5.0

Type	Key	Summary
	BAM-13428	Please update Upgrade Guide for Bamboo 5.0 on CAC
	BAM-13419	Reorder environments on the deployments and project pages
	BAM-13366	Pattern Match Labelling not working...
	BAM-13360	Rename a deployment release
	BAM-13354	Don't show the plans that a user doesn't have permission to view in the agent activity list
	BAM-13319	After switching to Bamboo 5.0-beta3 exports don't work anymore
	BAM-13314	Artifact Downloader Task broken in Bamboo 5 beta2
	BAM-13306	java.io.UTFDataFormatException: encoded string too long: 85338 bytes
	BAM-13291	5.0-beta2 tries to do AWS setup even if there are no AWS account details
	BAM-13288	Broken link to "instant messaging server" in the Bamboo Administration home page
	BAM-13222	Use an absolute path for Clover history dir for automatic Clover integration
	BAM-13204	Changeset ids are sometimes not shown on result page
	BAM-13199	Correct the wordings of war file location configuration in the tomcat deployer
	BAM-13187	NullPointerException when retrieving files from CommitContext

	BAM-13163	"Change Subversion branches URL" checkbox gets unchecked after editing another tab
	BAM-13161	5 Minor JavaDoc Typos
	BAM-13104	Bamboo is unable to start EC2 instances when custom inbound security rules contain wildcard ports
	BAM-13062	Please provide Job Name and Job Key as variables
	BAM-13022	Show "System Errors" tab in an Agent page
	BAM-13018	False error when re-creating a deleted plan

 Authenticate to retrieve your issues

77 more issues

Who are the Bamboo team?

Product management

Mark Chaimungkalanont

James Dumay

Agile evangelism

Sarah Goff-Dupont

Development

Brydie McCoy

David Hernandez

Piotr Stefaniak

Krystian Brazulewicz

Marcin Gardias

Marek Went

Przemek Bruski

UI Development

Matthias Schreck

Jason Berry

Documentation

Nathan Pye

Paul Watson

Bamboo 5.0 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 5.0 from a previous version of Bamboo. For details on the Bamboo 5.0 release, see the [Bamboo 5.0 Release Notes](#).

Please follow the Bamboo 5.0-specific instructions on this page, in addition to the upgrade instructions in the [Bamboo generic upgrade guide](#).

Please read the [Supported platforms](#) page for the full list of supported platforms for Bamboo.

On this page:

- [Upgrade notes from Bamboo 4.4 to 5.0](#)
- [Upgrading from Bamboo prior to 4.1](#)
- [Developing for Bamboo 5.0](#)
- [Checking for known issues and troubleshooting the Bamboo upgrade](#)

Upgrade notes from Bamboo 4.4 to 5.0

To upgrade to Bamboo 5.0, follow the instructions in the [Bamboo generic upgrade guide](#).

We *strongly recommend* that you back up your Bamboo instance and database before upgrading, as described in the [Bamboo generic upgrade guide](#).

'buildnumber.txt' generation has been removed and replaced by an optional plugin

The 'buildnumber.txt' file is no longer automatically generated in the working directory. If you rely on this functionality, then you must install the [Build Number Stamper plugin](#) from the Atlassian Marketplace.

No agent upgrade required

In Bamboo 3.2, agents were changed so that no upgrade of agents are required. When Bamboo is upgraded, agents will automatically restart and update their executables from the server.

Sonar tasks plugin 1.7.0 may cause some UI problems

Sonar tasks plugin 1.7.0 may cause some parts the UI to display incorrectly but does not cause any functional issues. This problem has [been reported to the plugin vendor](#).

Upgrading from Bamboo prior to 4.1

In addition to the notes below, please read the [Upgrade Guide](#) for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available [here](#).

Notes for upgrading from Bamboo 4.0

- Bamboo's deprecated Remote API has been removed. If you are using this API, migrate to the [Bamboo REST API](#).
- There are no major schema upgrade tasks that may cause the Bamboo upgrade from 3.4 to 4.0 to take an extended amount of time.
- If you are using [Elastic Bamboo](#), we've upgraded JDK6, Grails 1.2, Grails 1.3 and Maven 3 to the latest minor releases on the stock images. Additionally, we've added Grails 2.0 to the image. See [here](#) for a complete list of elastic image contents.

Notes for upgrading from Bamboo 3.2

- If you are using Bamboo with [Crowd](#), follow the instructions in [Upgrading Bamboo with Crowd to Bamboo 3.2](#).
- If you've been using Amazon EC2 images with your custom EBS, see [Updating EBSes created for Fedora to support Amazon Linux](#)
- If you've customised Amazon EC2 images to work with Bamboo, see [Creating a custom elastic image](#).

Notes for upgrading from a version of Bamboo prior to 2.7.4

- You will need to upgrade to Bamboo 2.7.4 before upgrading to Bamboo 4.3. If you are using a version of Bamboo earlier than 2.6.3, we recommend that you upgrade to 2.6.3 before upgrading to Bamboo 2.7.4. Bamboo 2.6.3 can be downloaded from the [Bamboo Archived Downloads page](#). Bamboo 2.7.x introduces a number of significant and irreversible changes, so a phased upgrade is recommended. Please see the [Bamboo 2.7.x Upgrade Guide](#) for more details.
- You will need to set aside time, as described in the [Bamboo 2.7.x Upgrade Guide](#), for Bamboo to migrate existing Plans to the new Plan structure in Bamboo 2.7.4.
- If you are upgrading from Bamboo 2.5 or earlier, you will need to set aside time, as described in the [Bamboo 2.6 Upgrade Guide](#) for Bamboo to migrate its test result data (stored in XML files on the filesystem) into the database.
- If you are upgrading from a version of Bamboo prior to 2.0, you must upgrade to [Bamboo 2.0.6](#) first before upgrading to Bamboo 2.6.3 (and then 2.7.4). Please read the [Bamboo 2.0 Upgrade Guide](#) for important upgrade instructions for upgrading from earlier versions of Bamboo.

Developing for Bamboo 5.0

If you are a Bamboo plugin developer, please refer to our [Bamboo API Changes](#) guide, which outlines changes in Bamboo that may affect Bamboo plugins compiled for earlier versions of Bamboo.

Checking for known issues and troubleshooting the Bamboo upgrade

If something is not working correctly after you have completed the steps above to upgrade your Bamboo installation, please check for known Bamboo issues and try troubleshooting your upgrade as described below:

- **Check for known issues.** Sometimes we find out about a problem with the latest version of Bamboo after we have released the software. In such cases we publish information about the known issues in the Bamboo Knowledge Base. Please check the [Bamboo Known Issues](#) in the Bamboo Knowledge Base and follow the instructions to apply any necessary patches if necessary.
- **Did you encounter a problem during the Bamboo upgrade?** Please refer to the guide to [troubleshooting upgrades](#) in the Bamboo Knowledge Base.
- If you encounter a problem during the upgrade and cannot solve it, please create a [support ticket](#) and one of our support engineers will help you.

Bamboo 5.0 beta 3 Release Notes

12th June 2013

Our third pre-release of **Bamboo 5** is now available. The support for deployment orchestration in the Bamboo 5.x series is our most ambitious endeavor yet. And the feedback we get from you in these next few weeks is absolutely critical in making this the most useful and intuitive Bamboo ever.

Notable changes in beta 3 include:

- A brand new, fresh icon set designed by [Valter Fatia \[Atlassian\]](#) that perfectly complements the [Atlassian Design Guidelines](#).
- Versions can be created without having to deploy.
- Directly linking to the deployment result and logs when the deployment has failed to make troubleshooting tasks quicker.
- New design for the deployment preview.
- Display order for environments can now be changed in the deployment project configuration.
- Both issues and commits for a version can be compared against another version to get changes between any two versions.
- Navigation improvements. It should be no more than two clicks within Deployment projects to get to relevant information or the configuration.

If you notice any more problems or think we have missed anything, please report them via the in app feedback mechanism.

Download beta 3

This list is by no means a complete list of changes that will be available in 5.0 and the scheduling of some items may change without notice.

Updates and fixes

Key	Summary	Status
BAM-10855	Convert ANSI color codes to HTML when displaying logs	 Resolved
BAM-10573	Pass artifacts between plans	 Resolved
BAM-10089	Improvements to EBS mounting	 Resolved
BAM-11104	Ability to skip manual stage	 Resolved
BAM-13065	Disabling Automatic Dependency management does not remove the generated dependencies.	 Resolved
BAM-13062	Please provide Job Name and Job Key as variables	 Resolved
BAM-12059	Working Directory help bubble gives page not found	 Resolved
BAM-11719	Anonymous users can view remote agent details	 Resolved
BAM-11620	JIRA Bamboo release plugin doesn't start build if plan has no repository	 Resolved
BAM-11602	Bamboo sometimes kills processes not belonging to it on Windows	 Resolved
BAM-12043	Unable to link svn repositories to viewVC	 Resolved
BAM-11899	Agent Log message handling improvements	 Resolved

BAM-11995	"Create Issue" button next to "Quarantine"	 Resolved
BAM-12115	Add new instance types to Elastic Bamboo	 Resolved
BAM-12128	Truncate Failure reason for a merge_result before persisting to the database	 Resolved
BAM-11359	Upgrade to Amazon Linux 2012.09.1	 Resolved
BAM-11342	Javascript error when pressing "Run" in Bamboo plan	 Resolved
BAM-12158	Git repository password displayed as plain text	 Resolved
BAM-12169	"Enable Commit Isolation" shows up twice	 Resolved
BAM-12208	"password" variable shows in metadata	 Resolved

 [Authenticate](#) to retrieve your issues

[83 more issues](#)

Bamboo 5.0 beta 2 Release Notes

28th May 2013

Our second pre-release build of **Bamboo 5** is now available. The support for deployment orchestration in the Bamboo 5.x series is our most ambitious endeavor yet. And the feedback we get from you in these next few weeks is absolutely critical in making this the most useful and intuitive Bamboo ever.

Notable changes in beta 2 include:

- Restoration of plugin compatibility for Task plugins – most plugins from 4.4.x will install fine if you manually download them and use the plugin manager to install them. However, they would not be usable within Deployment Projects without a Bamboo 5.0 specific update to the plugin by the 3rdparty plugin developer.
- Fixes for the "Waiting for null" deployment status condition that effected some users.
- Page crashes when viewing the Deployment Details.
- Deployment project now works if the linked Plan is changed.
- Bug fixes to the way that the Download Artifact task downloads artifacts.

If you notice any more problems or think we have missed anything, please report them via the in app feedback mechanism.

Download beta 2

This list is by no means a complete list of changes that will be available in 5.0 and the scheduling of some items may change without notice.

Updates and fixes in Bamboo 4.4.1

Key	Summary	Status
BAM-10855	Convert ANSI color codes to HTML when displaying logs	 Resolved
BAM-10573	Pass artifacts between plans	 Resolved
BAM-10089	Improvements to EBS mounting	 Resolved
BAM-11104	Ability to skip manual stage	 Resolved
BAM-13065	Disabling Automatic Dependency management does not remove the generated dependencies.	 Resolved
BAM-13062	Please provide Job Name and Job Key as variables	 Resolved

BAM-12059	Working Directory help bubble gives page not found	 Resolved
BAM-11719	Anonymous users can view remote agent details	 Resolved
BAM-11620	JIRA Bamboo release plugin doesn't start build if plan has no repository	 Resolved
BAM-11602	Bamboo sometimes kills processes not belonging to it on Windows	 Resolved
BAM-12043	Unable to link svn repositories to viewVC	 Resolved
BAM-11899	Agent Log message handling improvements	 Resolved
BAM-11995	"Create Issue" button next to "Quarantine"	 Resolved
BAM-12115	Add new instance types to Elastic Bamboo	 Resolved
BAM-12128	Truncate Failure reason for a merge_result before persisting to the database	 Resolved
BAM-11359	Upgrade to Amazon Linux 2012.09.1	 Resolved
BAM-11342	Javascript error when pressing "Run" in Bamboo plan	 Resolved
BAM-12158	Git repository password displayed as plain text	 Resolved
BAM-12169	"Enable Commit Isolation" shows up twice	 Resolved
BAM-12208	"password" variable shows in metadata	 Resolved

 [Authenticate](#) to retrieve your issues

83 more issues

Bamboo security advisories

As a distributed application, Bamboo's application-level security is important. This document contains links to version-specific security advisories and related documents for the Bamboo application.

 This document is intended to provide information to system administrators about the security of the Bamboo application. It does not describe [Bamboo's internal security model](#) – user management and permissions – except as it relates to the overall application security.

Finding and reporting a security vulnerability

Atlassian's approach to releasing patches is detailed in [How to Report a Security Issue](#).

Publication of Bamboo security advisories

Atlassian's approach to publishing security advisories is detailed in [Security Advisory Publishing Policy](#).

Severity levels

Atlassian's scale for measuring security issues is detailed in [Severity Levels for Security Issues](#).

Our patch policy

Atlassian's approach to releasing patches is detailed in our [Security Patch Policy](#).

Security advisories

- [Bamboo Security Advisory 2013-07-16](#)
- [Bamboo Security Advisory 2012-08-28](#)
- [Bamboo Security Advisory 2012-05-17](#)
- [Bamboo Security Advisory 2012-01-31](#)
- [Bamboo Security Advisory 2011-11-22](#)
- [Bamboo Security Advisory 2011-03-29](#)
- [Bamboo Security Advisory 2010-05-04](#)
- [Bamboo Security Advisory 2009-03-09](#)
- [Bamboo Security Advisory 2008-02-08 \(Bamboo 2.0 Beta\)](#)

Bamboo Security Advisory 2012-08-28

This advisory announces a security vulnerability that we have found in Bamboo and fixed in a recent version of Bamboo.

- **Customers who have downloaded and installed Bamboo** should upgrade their existing Bamboo installations to fix this vulnerability.
- **Atlassian OnDemand** are not affected by any of the issues described in this advisory.

Atlassian is committed to improving product security. The vulnerability listed in this advisory has been discovered by Atlassian, unless noted otherwise. The reporter may also have requested that we do not credit them.

If you have questions or concerns regarding this advisory, please raise a support request at <http://support.atlassian.com/>.

In this advisory:

- [OGNL Injection Vulnerability](#)

OGNL Injection Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as **Critical**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, moderate or low.

This is an independent assessment and you should evaluate its applicability to your own IT environment.

Description

We have identified and fixed a vulnerability caused by the way WebWorks/Struts and Freemarker templates are used in Bamboo. The vulnerability allows a non-authenticated user to execute arbitrary Java methods in the JVM hosting the Bamboo application. This can be used to execute OS commands as the JVM user.

All versions of Bamboo up to and including 4.0.1 are affected. This issue can be tracked here:


BAM-12066 - OGNL injection vulnerability (
 
Resolved)

This vulnerability has been fixed in Bamboo 4.1. A patch is available for Bamboo 3.0 and above

Bamboo Vulnerability	Affected versions	Fixed Version	Issue Tracking
Elevation of privileges	4.x 3.x	4.1 Patches available	 BAM-12066 - OGNL injection vulnerability ( Resolved)

Risk Mitigation

If you cannot upgrade immediately, you should disable public access to your Bamboo instance to mitigate the risk of this vulnerability.

Fix

Upgrade

The vulnerabilities and fix versions are described in the 'Description' section above.

We recommend that you upgrade to the latest version of Bamboo, if possible. For a full description of the latest version of Bamboo, see the [release notes](#). You can download the latest version of Bamboo from the [download centre](#).

Bamboo Security Advisory 2012-05-17

This advisory discloses a **critical** security vulnerability that exists in all versions of Bamboo up to and including 3.4.4.

- **Customers who have downloaded and installed Bamboo** should upgrade their existing Bamboo installations to fix this

vulnerability.

- **Enterprise Hosted customers** need to request an upgrade by raising a support request at <http://support.atlassian.com> in the "Enterprise Hosting Support" project.
- **JIRA Studio and Atlassian OnDemand customers** are not affected by any of the issues described in this advisory.

Atlassian is committed to improving product security. The vulnerability listed in this advisory has been discovered by Atlassian, unless noted otherwise. The reporter may also have requested that we do not credit them.

If you have questions or concerns regarding this advisory, please raise a support request at <http://support.atlassian.com/>.

In this advisory:

- [Critical XML Parsing Vulnerability](#)
 - [Severity](#)
 - [Description](#)
 - [Risk Mitigation](#)
 - [Fix](#)

Critical XML Parsing Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as **critical**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, medium or low.

This is an independent assessment and you should evaluate its applicability to your own IT environment.

Description

We have identified and fixed a vulnerability in Bamboo that results from the way third-party XML parsers are used in Bamboo.

This vulnerability allows an attacker to:

- execute denial of service attacks against the Bamboo server, and
- read all local files readable to the system user under which Bamboo runs.

The attacker needs to have an account with the affected Bamboo server instance and be able to log in in order to execute the attack.

All versions of Bamboo **up to and including 3.4.4** are affected by this vulnerability. This issue can be tracked here:



Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix this vulnerability.

Alternatively, if you are not in a position to upgrade or apply patches immediately, you should do **all** of the following until you can upgrade or patch. Please note, these measures will only limit the impact of the vulnerability, they will not mitigate it completely.

- Disable public access (such as [anonymous access](#) and [public signup](#)) to your Bamboo instance until you have applied the necessary patch or upgrade.
- Ensure that your Bamboo system user is restricted as described in [best practices for Bamboo security](#).

Fix

Upgrade (recommended)

Upgrade to Bamboo 4.0 or later which fixes this vulnerability. For a full description of this release, see the [Bamboo 4.0 release notes](#). The following releases have also been made available to fix this vulnerability in older Bamboo versions:

- Bamboo 3.3.4 for Bamboo 3.3.x
- Bamboo 3.4.5 for Bamboo 3.4.x

You can download these versions from the [Bamboo download centre](#).

Patches (not recommended)

Patches are only available for Bamboo 3.2.x - 3.4.x. We recommend patching only when you can neither upgrade nor apply external security controls. Patches are usually only provided for vulnerabilities of critical severity (as per our [Security Patch Policy](#)), as an interim solution until you can upgrade. You should not expect that you can continue patching your system instead of upgrading. Our patches are often non-cumulative – we do not recommend that you apply multiple patches from different advisories on top of each other, but strongly recommend upgrading to the most recent version regularly.

If for some reason you cannot upgrade to the latest version of Bamboo, you must do **all** of the following steps to fix the vulnerability described in this security advisory.

1. Download the file <http://www.atlassian.com/software/bamboo/downloads/binary/patch-BAM11316-3.2-atlassian-bundled-plugins.zip>
2. Rename the file to `atlassian-bundled-plugins.zip`
3. Stop Bamboo.

4. Make a backup of the <bamboo_install_dir> directory.
5. Copy [atlassian-bundled-plugins.zip](#) into `webapp/WEB-INF/classes` in the <bamboo_install_dir>, to replace the existing file of the same name.
6. Restart Bamboo.

Bamboo Security Advisory 2012-01-31

This advisory discloses two **CRITICAL** security vulnerabilities that exist in all versions of Bamboo up to and including 3.4.2. You need to upgrade your existing Bamboo installations to fix these vulnerabilities. Enterprise Hosted customers should request an upgrade by raising a support request at <http://support.atlassian.com> in the "Enterprise Hosting Support" project. Neither Bamboo Studio nor Atlassian OnDemand are vulnerable to any of the issues described in this advisory.

Atlassian is committed to improving product security. The vulnerabilities listed in this advisory have been discovered by Atlassian, unless noted otherwise. The reporter may also have requested that we do not credit them.

If you have questions or concerns regarding this advisory, please raise a support request at <http://support.atlassian.com/>.

In this advisory:

- [Code Injection Vulnerability](#)
 - [Severity](#)
 - [Description](#)
 - [Vulnerability](#)
 - [Risk Mitigation](#)
 - [Fix](#)
 - [Patches](#)
 - [Applying the patch](#)
- [Arbitrary File Disclosure Vulnerability](#)
 - [Severity](#)
 - [Description](#)
 - [Vulnerability](#)
 - [Risk Mitigation](#)
 - [Fix](#)
 - [Patches](#)
 - [Applying the patch](#)

Code Injection Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as **CRITICAL**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, medium or low.

Description

We have identified and fixed a code injection vulnerability in Bamboo caused by an underlying vulnerability in the **third-party** Webwork 2 framework. This vulnerability allows an attacker to run arbitrary Java code on a Bamboo server with user privileges of a Bamboo process. This vulnerability is a variant of a recently disclosed [Struts2 vulnerability](#). The vulnerability exists in pages accessible by non-privileged users and can also be exploited by use of social engineering, e.g. having a legitimate click on a specially crafted link.

The maintainer of the original library can be contacted at <http://struts.apache.org/>

Vulnerability

The table below describes the Bamboo version and the specific functionality affected by the Webwork 2 vulnerability.

Bamboo Component	Affected Bamboo Versions	Fixed Versions	Issue Tracking
Webwork 2	All versions up to and including 3.4.2	3.3.4 3.4.3	BAM-10627

Risk Mitigation

We highly recommend that you upgrade your Bamboo installation to fix these vulnerabilities.

Alternatively, if you are not in a position to upgrade immediately and you judge it necessary, you can restrict access to your instance of Bamboo by using a firewall.

Fix

Bamboo 3.4.3 and later versions fix this issue. View the issue linked above for information about fix versions. For a full description of the latest version of Bamboo, see the [release notes](#). You can download the latest version of Bamboo from the [Bamboo download centre](#).

If you cannot upgrade to the latest version of Bamboo, you can patch your existing installation using the patch listed below. We strongly recommend upgrading and not patching.

Patches

A binary patch for the Webwork 2 vulnerability is available for Bamboo versions 3.0 and later. The patch is attached to the [BAM-10627](#) tracking issue.

Vulnerability	Patch	Patch File Name
Code injection vulnerability in third-party Webwork 2 framework used by Bamboo	Attached to BAM-10627 issue	SimpleConversionErrorInterceptor.zip

Applying the patch

If you are using Bamboo 3.0 or later:

1. Download the [SimpleConversionErrorInterceptor.zip](#) file that is attached to the [BAM-10627](#) issue.
2. Stop Bamboo.
3. Make a backup of the <bamboo_install_dir> directory.
4. Create directories com/atlassian/bamboo/ww2/interceptors in the WEB-INF/classes directory, which can be found within your Bamboo installation.
5. Unzip [SimpleConversionErrorInterceptor.zip](#) into com/atlassian/bamboo/ww2/interceptors:

```
mkdir -p com/atlassian/bamboo/ww2/interceptors
cd com/atlassian/bamboo/ww2/interceptors
unzip SimpleConversionErrorInterceptor.zip
```

6. Add a reference to the new SimpleConversionErrorInterceptor in the xwork.xml file in WEB-INF/classes:

```
<xwork>
...
<interceptor name="conversionError"
class="com.atlassian.bamboo.ww2.interceptors.SimpleConversionErrorInterceptor"
/>
...
</xwork>
```

7. Restart Bamboo.

Arbitrary File Disclosure Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as **CRITICAL**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, medium or low.

Description

We have identified and fixed a vulnerability in Bamboo caused by a combination of issues in **third-party** libraries, including FreeMarker template library, used in Bamboo. This vulnerability allows an attacker to access any files on Bamboo server that are readable by the Bamboo server process. The attacker does not need to authenticate in order to exploit the vulnerability. The vulnerability is related to the previously disclosed [FreeMarker issue](#). The vulnerability **does not affect** Bamboo installations using Tomcat as will usually be present only in Bamboo standalone.

Vulnerability

The table below describes the Bamboo versions and the specific functionality affected by the arbitrary file disclosure vulnerability.

Bamboo Component	Affected Bamboo Versions	Fixed Versions	Issue Tracking
FreeMarker	All versions up to and including 3.4.2	3.3.4 3.4.3	BAM-10628

Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix this vulnerability.

Alternatively, if you are not in a position to upgrade immediately and you judge it necessary, you can restrict access to your instance of Bamboo by using a firewall.

Fix

Bamboo 3.4.3 and later versions fix this issue. View the issue linked above for information about fix versions. For a full description of the latest version of Bamboo, see the [release notes](#). You can download the latest version of Bamboo from the [Bamboo download centre](#).

If you cannot upgrade to the latest version of Bamboo, you can patch your existing installation using the patch listed below. We strongly recommend upgrading and not patching.

Patches

A binary patch for the FreeMarker vulnerability is available for Bamboo versions 3.0 and later. The patch is attached to the [BAM-10628](#) tracking issue.

Vulnerability	Patch	Patch File Name
File disclosure vulnerability in third-party FreeMarker template library used by Bamboo	Attached to BAM-10628 issue	freemarker-2.3.16-atlassian-11.jar

Applying the patch

If you are using Bamboo 3.0 or later:

1. Download the [freemarker-2.3.16-atlassian-11.jar](#) file that is attached to the [BAM-10628](#) issue.
2. Stop Bamboo.
3. Make a backup of the <bamboo_install_dir> directory.
4. Copy [freemarker-2.3.16-atlassian-11.jar](#) to WEB-INF/lib.
5. Move the existing freemarker jar to a backed up location.
6. Restart Bamboo.

Bamboo Security Advisory 2011-11-22

This advisory discloses a number of security vulnerabilities that we have found in versions of Bamboo prior to 3.3. You need to upgrade your existing Bamboo installations to fix these vulnerabilities. Enterprise Hosted customers should request an upgrade by raising a support request at <http://support.atlassian.com> in the "Enterprise Hosting Support" project. Neither Bamboo Studio nor OnDemand are vulnerable to any of the issues described in this advisory.

Atlassian is committed to improving product security. The vulnerabilities listed in this advisory have been discovered by Atlassian, unless noted otherwise. The reporter may also have requested that we do not credit them.

If you have questions or concerns regarding this advisory, please raise a support request at <http://support.atlassian.com/>.

In this advisory:

- [XSS Vulnerabilities](#)
 - [Severity](#)
 - [Risk Assessment](#)
 - [Vulnerability](#)
 - [Risk Mitigation](#)
 - [Fix](#)
- [OS Command Injection Vulnerability](#)
 - [Severity](#)
 - [Risk Assessment](#)
 - [Vulnerability](#)
 - [Risk Mitigation](#)
 - [Fix](#)
 - [Patches](#)
 - [Patch Procedure: Install the Patch](#)
- [Information Leakage Vulnerability](#)
 - [Severity](#)
 - [Risk Assessment](#)
 - [Vulnerability](#)
 - [Risk Mitigation](#)
 - [Fix](#)

XSS Vulnerabilities

Severity

Atlassian rates the severity level of all these vulnerabilities as **high**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, medium or low. These vulnerabilities are **not** critical.

Risk Assessment

We have identified and fixed a number of reflected and stored cross-site scripting (XSS) vulnerabilities in Bamboo. XSS vulnerabilities allow an attacker to embed their own JavaScript into a Bamboo page. You can read more about XSS attacks at cgisecurity.com, [The Web](#)

[Application Security Consortium](#) and other places on the web.

Vulnerability

The table below describes the Bamboo versions and the specific functionality affected by the XSS vulnerabilities.

Bamboo Feature	Affected Bamboo Versions	Fixed Version	Issue Tracking
User Picker	all earlier than 2.7.4	2.7.4, 3.0	BAM-10024
Default 'internal server error' page	all earlier than 3.1	3.1	BAM-10026
viewAgent.action	all earlier than 3.1	3.1	BAM-10027
configureAgents resource	all earlier than 3.1	3.1	BAM-10028
chooseBuildsToMove.action	all earlier than 3.1	3.1	BAM-10029

Our thanks to Marian Ventuneac (<http://www.ventuneac.net>) who reported several of the vulnerabilities mentioned above. We fully support the reporting of vulnerabilities and we appreciate it when people work with us to identify and solve the problem.

Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix these vulnerabilities.

Alternatively, if you are not in a position to upgrade immediately and you judge it necessary, you can restrict access to trusted [groups](#).

Fix

Bamboo 3.1 and later versions fix all these issues. View the issue linked above for information on fix versions. For a full description of the latest version of Bamboo, see the [release notes](#). You can download the latest version of Bamboo from the [Bamboo download centre](#).

There are no patches available to fix these vulnerabilities. You must upgrade your Bamboo installation.

OS Command Injection Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as **high**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, medium or low. This vulnerability is **not** critical.

Risk Assessment

We have identified and fixed an OS command injection vulnerability in the **third-party** Perforce library used in Bamboo. This vulnerability allows an attacker to execute arbitrary OS commands on a Bamboo server as Bamboo user. The attacker needs to have plan edit rights. Only the servers that have Perforce integration enabled (i.e. have a Perforce capability defined on the server) can be exploited. You can read more about command injection attacks and consequences at [OWASP](#) and other places on the web.

Note that if your server has local agents enabled, anyone who controls build plans is already capable of causing arbitrary code to run locally as part of the normal build process, and this bug does not lead to any additional access.

The maintainer of the original library can be contacted at <https://github.com/digerata/P4Java/>

Vulnerability

The table below describes the Bamboo versions and the specific functionality affected by the OS command injection vulnerability.

Bamboo Feature	Affected Bamboo Versions	Fixed Version	Issue Tracking
OS command injection vulnerability in Perforce library	2.4 – 3.1	3.1.1, 3.2	BAM-10030

Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix this vulnerability.

Alternatively, if you are not in a position to upgrade immediately and you judge it necessary, you can restrict access to trusted [groups](#).

Fix

Bamboo 3.2 and later versions fix this issue. View the issue linked above for information on fix versions. For a full description of the latest version of Bamboo, see the [release notes](#). You can download the latest version of Bamboo from the [Bamboo download centre](#).

If you cannot upgrade to the latest version of Bamboo, you can patch your existing installation using the patch listed below. We strongly recommend upgrading and not patching.

Patches

If you are running Bamboo 2.4 – 3.1, you can apply the following library patch to fix the [BAM-10030](#) vulnerability. We strongly recommend upgrading and not patching.

Vulnerability	Patch	Patch File Name
OS command injection vulnerability in Perforce library used by Bamboo	Attached to issue BAM-10030	p4java-0.7.5-atlassian-6.jar

Patch Procedure: Install the Patch

A patch is available for Bamboo 2.4 – 3.1.

The patch addresses the following issue:

- OS command injection vulnerability in Perforce library used by Bamboo ([BAM-10030](#)).

Applying the patch

If you are using Bamboo 2.4 – 3.1:

- Download the [p4java-0.7.5-atlassian-6.jar](#) file that is attached to the [BAM-10030](#) issue.
- Stop Bamboo.
- Make a backup of the <bamboo_install_dir> directory.
- Copy the downloaded jar file into <bamboo_install_dir>/Bamboo/webapp/WEB-INF/lib, and delete the existing p4java jar file.
- Restart Bamboo.

Information Leakage Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as **medium**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, medium or low. This vulnerability is **not** critical.

Risk Assessment

We have identified and fixed an information leakage vulnerability in Bamboo. This vulnerability allows an attacker to view all directory listings (but not the content of the files) on the server readable by the Bamboo user.

Vulnerability

The table below describes the Bamboo versions and the specific functionality affected by the information leakage vulnerability.

Bamboo Feature	Affected Bamboo Versions	Fixed Version	Issue Tracking
Information leakage	2.0 – 3.2	3.2.3, 3.3	BAM-10031

Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix this vulnerability.

Alternatively, if you are not in a position to upgrade immediately and you judge it necessary, you can restrict access to trusted [groups](#).

Fix

Bamboo 3.3 and later versions fix this issue. View the issue linked above for information on fix versions. For a full description of the latest version of Bamboo, see the [release notes](#). You can download the latest version of Bamboo from the [Bamboo download centre](#).

There are no patches available to fix this vulnerability. You must upgrade your Bamboo installation.

Bamboo Security Advisory 2011-03-29

This advisory announces a security vulnerability that we have found in all versions of Bamboo prior to 2.7.4 and fixed in 2.7.4 and later. You need to upgrade your existing Bamboo installations to fix this vulnerability. JIRA Studio is not vulnerable to any of the issues described in this advisory.

Atlassian is committed to improving product security. The vulnerabilities listed in this advisory have been discovered by Atlassian, unless noted otherwise. The reporter may also have requested that we do not credit them.

If you have questions or concerns regarding this advisory, please raise a support request at <http://support.atlassian.com/>.

In this advisory:

- [XSS Vulnerability in Bamboo User Management](#)
 - [Severity](#)
 - [Risk Assessment](#)
 - [Vulnerability](#)
 - [Risk Mitigation](#)
 - [Fix](#)

XSS Vulnerability in Bamboo User Management

Severity

Atlassian rates the severity level of these vulnerabilities as **high**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a cross-site scripting (XSS) vulnerability in Bamboo. This XSS vulnerability allows an attacker to embed their own JavaScript into a Bamboo page. You can read more about XSS attacks and consequences at [cgisecurity.com](#), [The Web Application Security Consortium](#) and other places on the web.

Vulnerability

The table below describes the Bamboo versions and the specific functionality affected by the XSS vulnerability.

Bamboo Feature	Affected Bamboo Versions	Issue Tracking
Bamboo User Management	Bamboo 1.0 - 2.7.3	BAM-8260

Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix these vulnerabilities.

Alternatively, if you are not in a position to upgrade immediately and you judge it necessary, you can restrict access to trusted [groups](#).

Fix

Bamboo 2.7.4 and later versions fix this issue. View the issue linked above for information on fix versions. For a full description of this release, see the [Bamboo 2.7.4 Release Notes](#). You can download the latest version of Bamboo from the [Bamboo download centre](#).

There are no patches available to fix these vulnerabilities. You must upgrade your Bamboo installation.

Bamboo Security Advisory 2010-05-04

In this advisory:

- [XSS Vulnerabilities](#)
 - [Severity](#)
 - [Risk Assessment](#)
 - [Vulnerability](#)
 - [Risk Mitigation](#)
 - [Fix](#)
- [General Tightening of the Bamboo Security Model](#)
 - [Severity](#)
 - [Risk Assessment](#)
 - [Vulnerability](#)
 - [Risk Mitigation](#)
 - [Fix](#)
 - [Changed Behaviour in Bamboo](#)

XSS Vulnerabilities

Severity

Atlassian rates these vulnerabilities as **high**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed several cross-site scripting (XSS) vulnerabilities in Bamboo, which may affect Bamboo instances. These vulnerabilities have security implications and are especially important for anyone running publicly accessible instances of Bamboo.

- The attacker might take advantage of the vulnerability to steal other users' session cookies or other credentials, by sending the

- credentials back to the attacker's own web server.
- The attacker's text and script might be displayed to other people viewing a Bamboo page. This is potentially damaging to your company's reputation.

You can read more about XSS attacks at [cgisecurity](#), [CERT](#) and other places on the web.

Vulnerability

All version of Bamboo up to and including Bamboo 2.5.3 are susceptible to these vulnerabilities.

An attacker can inject their own malicious JavaScript code into areas of Bamboo listed in the table below. This code could be executed by simply entering the URL into the browser address bar or when a user performs a specific function in Bamboo, such as clicking a link or a button.

Affected areas in Bamboo
Server Administration User Interface — Including the User and Group Security , System and Communication sections.
Main Bamboo User Interface — Including the Create Plan and Build Configuration areas and Log and various Result views.

Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix these vulnerabilities. Please see the 'fix' section below.

Fix

Bamboo 2.5.5 fixes these vulnerabilities. See the [release notes](#) and [upgrade guide](#) for more information about this release and changes to Bamboo's behaviour. You can download the latest version of Bamboo from the [download centre](#).

There are no patches available to fix these vulnerabilities for previous versions of Bamboo.

General Tightening of the Bamboo Security Model

Severity

Atlassian rates one of these vulnerabilities as **high** and the other as **moderate**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed two potential security vulnerabilities in Bamboo. These vulnerabilities have security implications that are especially important for anyone running publicly accessible instances of Bamboo.

An attacker, who has gained administrator access to a Bamboo instance, could set Bamboo's export, import and scheduled backup paths to a location within the Bamboo web application directory. Once this has been done, the attacker will be able to download any Bamboo data which has been exported or backed up by Bamboo. If you have followed standard guidelines for hardening your application servers, then your Bamboo instance should be less susceptible to this vulnerability. Therefore, we have provided an optional mechanism that prevents directory paths from being changed.

Bamboo does not set a maximum number of repeated login attempts. This makes Bamboo vulnerable to brute force attacks. Therefore, we have prevented brute force attacks by imposing a maximum number of repeated login attempts.

For [Bamboo distributions](#), we have set Bamboo's session ID cookies to use the [HttpOnly](#) flag. This makes it more difficult for malicious (JavaScript) code on a client's browser to gain access to these session ID cookies, thereby minimising the risk of common XSS attacks.

Vulnerability

All version of Bamboo up to and including Bamboo 2.5.3 are susceptible to these vulnerabilities.

Please refer to the following JIRA issues for more information:

- [BAM-5775](#) for restricting the ability to set Bamboo's file paths.
- [BAM-5708](#) for brute force attack prevention in Bamboo.
- [BAM-5668](#) for HttpOnly session ID cookies in the Bamboo distribution (not EAR-WAR).

Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix these vulnerabilities. Please see the 'fix' section below.

If you are running the [Bamboo EAR-WAR distribution](#), then to minimise the risk of common XSS attacks, we strongly recommend that you configure the application server (Tomcat) running Bamboo to transmit session ID cookies using the HttpOnly flag. Please refer to [Configuring Tomcat to Use HttpOnly Session ID Cookies](#) for more information.

Fix

Bamboo 2.5.5 fixes these vulnerabilities. See the [release notes](#) and [upgrade guide](#) for more information about this release and changes to Bamboo's behaviour. You can download the latest version of Bamboo from the [download centre](#).

There are no patches available to fix these vulnerabilities for previous versions of Bamboo.

Changed Behaviour in Bamboo

As a consequence of these security fixes, the following changes to Bamboo's *default* behaviour have occurred.

- When modifying Bamboo's 'File Path' option on the [Export](#) or [Import](#) administration pages or the 'Backup Path' option on the [Scheduled Backup](#) page, you can only change the name of files associated with these options (not the the actual file path component itself). To change these file path components, you must explicitly run Bamboo with the following system property:

```
bamboo.paths.set.allowed=true
```

Please refer to [Configuring system properties](#) for details on how to run Bamboo with system properties.

- If you attempt to log in to Bamboo three times unsuccessfully, Bamboo will then require subsequent login attempts to be accompanied by text from a Captcha image.

For details about changes to Bamboo's behaviour as a result of these fixes to security vulnerabilities, please refer to the [Bamboo 2.5.5 Upgrade Guide](#).

Bamboo Security Advisory 2009-03-09

In this advisory:

- Security vulnerabilities
 - [XSS vulnerabilities on the User Profile page](#)
 - [XSS vulnerabilities when adding Requirements for a Build](#)
 - [XSS vulnerabilities in the user's full name](#)
 - [XSS vulnerabilities in build logs](#)

Security vulnerabilities

XSS vulnerabilities on the User Profile page

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the [Bamboo Security documentation](#). This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect Bamboo instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in Bamboo's '**User Profile**' page. This potentially allows a malicious user (hacker) to hack the URL of controls on the page (e.g. User Profile link) to insert special JavaScript. A hacker could present the hacked URL to users (e.g. disguised in an email). If any users clicked the URL, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.
- The hacker's text and script might be displayed to other people on the User Profile page. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to [Bamboo 2.2](#) to fix the vulnerabilities described below.

You can read more about XSS attacks at [cgisecurity](#), [CERT](#) and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your Bamboo system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict Bamboo access to trusted groups only.

Vulnerability

The User Profile page in Bamboo is affected. The URLs of links on this page are not HTML-escaped.

Fix

The fix is to HTML-encode the URLs of all links on the User Profile page, so that it cannot be used to run special scripts.

This issue has been fixed in [Bamboo 2.2](#) only. There are no patches available for previous versions of Bamboo, for this fix.

XSS vulnerabilities when adding Requirements for a Build

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the [Bamboo Security documentation](#). This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect Bamboo instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability when adding requirements for a build. This potentially allows a malicious user (hacker) to insert special JavaScript in the key of a requirement when adding it to a build. If any users clicked the requirement, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.
- The hacker's text and script might be displayed to other people on the User Profile page. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to [Bamboo 2.2](#) to fix the vulnerabilities described below.

You can read more about XSS attacks at [cgisecurity](#), [CERT](#) and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your Bamboo system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict Bamboo access to trusted groups only.

Vulnerability

The requirements for a build are affected. The key is not HTML-escaped. This affects all versions from 2.0 onwards.

Fix

The fix is to HTML-encode the keys of requirements for builds, so that they cannot be used to run special scripts.

This issue has been fixed in [Bamboo 2.2](#) only. There are no patches available for previous versions of Bamboo, for this fix.

XSS vulnerabilities in the user's full name

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the [Bamboo Security documentation](#). This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect Bamboo instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in the user's full name. This potentially allows a malicious user (hacker) to create a new user and hack the user's full name to insert special JavaScript. The user's full name is presented in a number of places, including author statistics page, build result comments, build changes and commit notifications. If any users clicked the user name, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.
- The hacker's text and script might be displayed to other people on the User Profile page. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to [Bamboo 2.2](#) to fix the vulnerabilities described below.

You can read more about XSS attacks at [cgisecurity](#), [CERT](#) and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your Bamboo system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict Bamboo access to trusted groups only.

Vulnerability

The author statistics page, build result comments, build changes and commit notifications are affected. The user name is not HTML-escaped.

Fix

The fix is to HTML-encode the user's full name on these pages/notifications, so that it cannot be used to run special scripts.

This issue has been fixed in [Bamboo 2.2](#) only. There are no patches available for previous versions of Bamboo, for this fix.

XSS vulnerabilities in build logs

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the [Bamboo Security documentation](#). This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect Bamboo instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in the Bamboo build logs. This potentially allows a malicious user (hacker) to insert special JavaScript into a build log. If a user opened the hacked build log, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.
- The hacker's text and script might be displayed to other people on the User Profile page. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to [Bamboo 2.2](#) to fix the vulnerabilities described below.

You can read more about XSS attacks at [cgisecurity](#), [CERT](#) and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your Bamboo system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict Bamboo access to trusted groups only.

Vulnerability

The Bamboo build logs are affected. The log lines are not HTML-escaped.

Fix

The fix is to HTML-encode the log entries for the build logs, so that they cannot be used to run special scripts.

This issue has been fixed in [Bamboo 2.2](#) only. There are no patches available for previous versions of Bamboo, for this fix.

Please let us know what you think of the format of this security advisory and the information we have provided.

Bamboo Security Advisory 2008-02-08 (Bamboo 2.0 Beta)

In this advisory:

- [Bamboo 2.0 Beta Security Considerations](#)
 - [Risk Assessment](#)
 - [Vulnerability](#)
 - [Fix](#)

Bamboo 2.0 Beta Security Considerations

Risk Assessment

The Bamboo 2.0 Beta does not include the security features that will be present in the final released product. Please note the following security implications when enabling Bamboo's **remote agent** functionality:

- No encryption of data passed between server and agent — this includes data such as:
 - login credentials for version control repositories
 - build logs
 - build artifacts
- No authentication of the agent or server — this could result in unauthorised actions being taken on your system, such as:
 - Unauthorised parties installing new remote agents — version control repository login credentials could be stolen.
 - Unauthorised parties masquerading as a Bamboo server — the unauthorised server could pass malicious code to the agent to run.

We **strongly recommend that you do not enable remote agent installation** on any Bamboo instance accessible from a public or untrusted network. Creating remote agents is [disabled](#) by default.

These are limitations of the beta release only and will be addressed before the final released product.

Vulnerability

An unauthorised party could steal sensitive data passing between the Bamboo server and agents or run malicious code on your agents, as described in the '**Risk Assessment**' section.

Fix

These are limitations of the **beta release only** and will be addressed before the final released product.

Bamboo Security Advisory 2013-07-16

This advisory discloses a security vulnerability that we have found in Bamboo and fixed in a recent version of Bamboo.

- **Customers who have downloaded and installed Bamboo** should upgrade their existing Bamboo installations to fix this vulnerability.
- **Atlassian OnDemand customers** are not affected by any of the issues described in this advisory.

Atlassian is committed to improving product security.

The vulnerability listed in this advisory is a **vulnerability in a third-party framework - Struts 2 / WebWork 2** that is used by Bamboo. The vulnerability has been independently discovered by Atlassian and reported to the Struts maintainers.

More details about the underlying Struts vulnerability CVE-2013-2251 are available at the [CVE database](#) and in the [Struts advisory](#).

If you have questions or concerns regarding this advisory, please raise a support request at <http://support.atlassian.com/>.

OGNL injection in WebWork 2

Severity

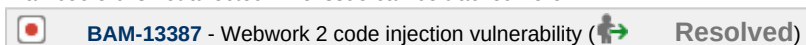
Atlassian rates the severity level of this vulnerability as **critical**, according to the scale published in [Severity Levels for Security Issues](#). The scale allows us to rank the severity as critical, high, moderate or low.

This is an independent assessment and you should evaluate its applicability to your own IT environment.

Description

We have fixed a vulnerability in the third-party web framework WebWork 2, which is a part of the Struts web framework. In specific circumstances, attackers can use this vulnerability to execute Java code of their choice on systems that use these frameworks. In the case of Bamboo, the attacker needs to be able to access the Bamboo web interface. A valid user account is not required to exploit this vulnerability.

Customers should be advised that this affects all versions of Bamboo, except Bamboo OnDemand, Bamboo 4.3.4 and Bamboo 4.4.8 or later. Bamboo 5.0 is not affected. The issue can be tracked here:



Risk Mitigation

If you are unable to upgrade or patch your Bamboo server: as a **temporary workaround**, you can do the following:

- Block access to all URLs on a Web Application Firewall or a reverse proxy that contain any of the following strings: "redirect:", "action:" or "redirect-action:" strings. A partial example for an nginx server is below. Note that the example only covers the "redirect:" prefix and does not account for any URL encoding that may be present.

```
location ~* ^/<path to your Bamboo>/ {
    if ($args ~* "redirect:") {
        return 403;
    }
    proxy_pass http://$host.internal$request_uri;
}
```

or

- Block access to your Bamboo server from untrusted networks, such as the Internet.

Fix

This vulnerability can be fixed by upgrading Bamboo to either version 4.3.4, 4.4.8 or later. There are no patches available for this vulnerability — for any questions, please raise a support request at <http://support.atlassian.com/>.

The [Security Patch Policy](#) describes when and how we release security patches and security upgrades for our products.

Upgrading Bamboo

The fix versions for this vulnerability are described in the 'Description' section above.

We recommend that you upgrade to the latest version of Bamboo. For a full description of the latest version of Bamboo, see the [release notes](#). You can download the latest version of Bamboo from the [download centre](#).

Bamboo 5.0-rc1 Release Notes

18th June 2013

The first release candidate **Bamboo 5** is now available. The support for deployment orchestration in the Bamboo 5.x series is our most ambitious endeavor yet. And the feedback we get from you in these next few weeks is absolutely critical in making this the most useful and intuitive Bamboo ever.

Release candidate 1 is a near final build and is composed of bug fixes and smaller improvements related to deployment projects.

If you notice any more problems or think we have missed anything, please report them via the in app feedback mechanism.

Download release candidate 1

This list is by no means a complete list of changes that will be available in 5.0 and the scheduling of some items may change without notice.

Updates and fixes

Key	Summary	Status
BAM-10855	Convert ANSI color codes to HTML when displaying logs	 Resolved
BAM-10573	Pass artifacts between plans	 Resolved
BAM-10089	Improvements to EBS mounting	 Resolved
BAM-11104	Ability to skip manual stage	 Resolved
BAM-13065	Disabling Automatic Dependency management does not remove the generated dependencies.	 Resolved
BAM-13062	Please provide Job Name and Job Key as variables	 Resolved
BAM-12059	Working Directory help bubble gives page not found	 Resolved
BAM-11719	Anonymous users can view remote agent details	 Resolved
BAM-11620	JIRA Bamboo release plugin doesn't start build if plan has no repository	 Resolved
BAM-11602	Bamboo sometimes kills processes not belonging to it on Windows	 Resolved
BAM-12043	Unable to link svn repositories to viewVC	 Resolved
BAM-11899	Agent Log message handling improvements	 Resolved
BAM-11995	"Create Issue" button next to "Quarantine"	 Resolved
BAM-12115	Add new instance types to Elastic Bamboo	 Resolved
BAM-12128	Truncate Failure reason for a merge_result before persisting to the database	 Resolved
BAM-11359	Upgrade to Amazon Linux 2012.09.1	 Resolved
BAM-11342	Javascript error when pressing "Run" in	 Resolved

Bamboo plan

BAM-12158	Git repository password displayed as plain text	 Resolved
BAM-12169	"Enable Commit Isolation" shows up twice	 Resolved
BAM-12208	"password" variable shows in metadata	 Resolved



Authenticate to retrieve your issues

83 more issues

Installing and upgrading Bamboo

Release notes

Bamboo release notes

Bamboo release summary



Bamboo 4.4.1
Release Notes



Bamboo 5.0
Release Notes



Bamboo 4.4
Release Notes



Bamboo 4.3.3
Release Notes



Bamboo 4.3.2
Release Notes

Showing first 5
of 87 results

Security advisories

Bamboo security advisories



Bamboo
Security
Advisory
2011-11-22



Bamboo
Security
Advisory
2011-03-29



Bamboo
Security
Advisory
2010-05-04



Bamboo
Security
Advisory
2009-03-09



Bamboo
Security
Advisory
2008-02-08
(Bamboo 2.0
Beta)

Installing

Bamboo installation guide

Bamboo EAR-WAR installation guide

Bamboo installation guide for Linux

Bamboo installation guide for Mac

Bamboo installation guide for Windows

Connecting Bamboo to an external database

Bamboo remote agent installation guide

Supported platforms

Upgrading

Bamboo upgrade guides

Bamboo generic upgrade guide



Bamboo 5.0 Upgrade Guide



Bamboo 4.4 Upgrade Guide



Bamboo 4.3 Upgrade Guide



Bamboo 4.2 Upgrade Guide



Bamboo 4.1 Upgrade Guide

Showing first 5 of 67 results

Supported platforms

This page describes the supported platforms for **Bamboo 5.0.x**

Bamboo 4.4.x

.

Key: = Supported; = Not Supported

Java		
Oracle JDK	1.7 1.6 1.5¹	For the server, it is not enough to have just the JRE. Please ensure that you have the full JDK. You can download the Java SE Development Kit (JDK) from the Oracle website. Once the JDK is installed, you will need to set the JAVA_HOME environment variable, pointing to the root directory of the JDK. Some JDK installers set this automatically (check by typing 'echo %JAVA_HOME%' in

OpenJDK	1.7	a DOS prompt, or 'echo \$JAVA_HOME' in a shell). You need to do this before installing Bamboo, as Bamboo will automatically configure JDK capabilities based on the system environment variables on your machine. ¹Note that your agents can build software with any JDK version. You only need to run the agent and server using a supported JDK.
Operating Systems		
Microsoft Windows		Bamboo is a pure Java application and should run on any platform, provided all the JDK requirements are satisfied. If you are using Linux/UNIX: A dedicated user should be created to run Bamboo, as Bamboo runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run Bamboo in Linux: <pre>\$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/bamboo --shell /bin/bash bamboo</pre>
Linux / Solaris		
Apple Mac OS X		
Application Servers		
Apache Tomcat	7.0.x 6.0.x 5.5.x	Deploying multiple Atlassian applications in a single Tomcat container is not supported. We do not test this configuration and upgrading any of the applications (even for point releases) is likely to break it. There are also a number of known issues with this configuration (see this FAQ for more information). We also do not support deploying multiple Atlassian applications to a single Tomcat container for a number of practical reasons. Firstly, you must shut down Tomcat to upgrade any application and secondly, if one application crashes, the other applications running in that Tomcat container will be inaccessible. Finally, we recommend not deploying <i>any other applications</i> to the same Tomcat container that runs Bamboo, especially if these other applications have large memory requirements or require additional libraries in Tomcat's lib subdirectory.
Databases		
MySQL	5.x with JDBC Connector/J 5.1 5.0.x	
PostgreSQL	8.2+ with PostgreSQL Driver 8.4.x	
Microsoft SQL Server	2008 with JTDS 1.2.2 2005 with JTDS 1.2.2	
Oracle	11G with Oracle 11.2.x 10G	

HSQldb	 (for evaluation only)	Bamboo ships with a built-in HSQL database, which is fine for evaluation purposes but is somewhat susceptible to data loss during system crashes. For production environments we recommend that you configure Bamboo to use an external database .
Web Browsers		
Microsoft Internet Explorer	 9.0  8.0  7.0	
Mozilla Firefox	 Latest stable version supported	
Safari	 Latest stable version supported	
Chrome	 Latest stable version supported	
Source Repositories		
Mercurial	 client: 1.6-2.1.x  client: 2.1.0	NOTE: Mercurial 2.1 has a bug that makes it incompatible with Bamboo. Please use Mercurial 2.1.1 or later.
Subversion	 with server 1.5-1.7	- Bamboo 4.2, and later versions, support Subversion 1.7, but use the Subversion 1.6 Workspace Format by default to keep backwards compatibility with older Subversion working copies. You can set the <code>bamboo.svn.workspace.format</code> system property if your Bamboo plans need to use Subversion 1.7 commands as part of your build scripts. See Setting Bamboo to Support Subversion 1.7 Workspace Format for details. - Bamboo works with Subversion 1.8, but does not yet support the 1.8 Workspace Format.
Git	 with client 1.7+	
Perforce		
CVS		

Notes

Please note, Atlassian does not support custom elastic images. Consider customising the elastic agents started from your existing image instead. See [Creating a custom elastic image](#).

End of Support Announcements for Bamboo

This page contains announcements of the end of support for various platforms and browsers when used with Bamboo. This is summarised in the table below. Please see the sections following for the full announcements.

End of Support Matrix for Bamboo

The table below summarises information regarding the end of support announcements for **upcoming** Bamboo releases. If a platform (version) has already reached its end of support date, it is **not** listed in the table.

Platform	Bamboo End of Support
MySQL 5.0	December 2011
Oracle 10g	December 2011



Why is Atlassian ending support for these platforms?

Atlassian is committed to delivering improvements and bug fixes as fast as possible. We are also committed to providing world class support for all the platforms our customers run our software on. However, as the complexity of our applications grows, the cost of supporting multiple platforms increases exponentially. Each new feature has to be tested on several combinations of application servers, databases, web browsers, etc, with setup and ongoing maintenance of automated tests. Moving forward, we want to reduce the time spent there to increase Bamboo development speed significantly.

On this page (most recent announcements first):

- [Deprecated Databases for Bamboo \(4 October 2011\)](#)
- [Deprecated Java Platforms for Bamboo \(16 February 2011\)](#)
- [Deprecated Web Browsers for Bamboo \(16 February 2011\)](#)

Deprecated Databases for Bamboo (4 October 2011)

This section announces the end of Atlassian support for certain database versions for Bamboo. End of support means that Atlassian will not fix bugs related to certain database versions past the support end date.

We will **stop supporting the following database versions** in Bamboo 3.4, from December 2011:

- MySQL 5.0
- Oracle 10g

The details are below. Please refer to the list of [supported platforms](#) for details of platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eol-announcement@atlassian.com.

End of Life Announcement for Database Support

Database	Support End Date
MySQL 5.0	When Bamboo 3.4 releases, after December 2011
Oracle 10g	When Bamboo 3.4 releases, after December 2011

- **Notes for MySQL 5.0 and Oracle 10g:**
 - Atlassian intends to end support for MySQL 5.0 and Oracle 10g in Bamboo 3.4. Bamboo 3.3 is the last version that will support MySQL 5.0 and Oracle 10g.
 - 'Support End Date' means that Bamboo 3.3 and previously released versions will continue to work with MySQL 5.0 and Oracle 10g. However, Atlassian will not fix bugs affecting MySQL 5.0 and Oracle 10g past the support end date.
 - Bamboo 3.4 will not be tested with MySQL 5.0 and Oracle 10g.

Deprecated Java Platforms for Bamboo (16 February 2011)

This section announces the end of Atlassian support for certain Java Platforms for Bamboo.

We will **stop supporting the following Java Platforms:**

- From Bamboo 3.1, due in the first half of 2011, support for Java Platform 5 (JDK/JRE 1.5) will end.

We are ending support for Java Platform 5, in line with [Sun's Java SE Support Road Map](#) (i.e. "End of Service Life" for Java Platform 5 dated October 30, 2009). We are committed to helping our customers understand this decision and assist them in updating to Java Platform 6, our supported Java Platform.

The details are below. Please refer to the [Supported platforms](#) for more details regarding platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eol-announcement@atlassian.com.

End of Life Announcement for Java Platform Support

Java Platform	Support End Date
Java Platform 5 (JDK/JRE 1.5)	When Bamboo 3.1 releases, due in the first half of 2011

- **Java Platform 5 End of Support Notes:**
 - 'Support End Date' means that Bamboo 3.0.x and previous released versions will continue to work with Java Platform 5 (JDK/JRE 1.5), however we will not fix bugs related to Java Platform 5 past the support end date.
 - Bamboo 3.1 will only be tested with and support Java Platform 6 (JDK/JRE 1.6).
 - If you have concerns with this end of support announcement, please email eol-announcement@atlassian.com.

Deprecated Web Browsers for Bamboo (16 February 2011)

This section announces the end of Atlassian support for certain web browser versions for Bamboo. End of support means that Atlassian will not fix bugs related to certain web browser versions past the support end date.

We will **stop supporting the following web browser versions** from Bamboo 3.0, due February 2011:

- Microsoft Internet Explorer 7 (IE7)

The details are below. Please refer to the list of [supported platforms](#) for details of platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eo1-announcement@atlassian.com.

End of Life Announcement for Web Browser Support

Web Browser	Support End Date
Microsoft Internet Explorer (version 7 only)	When Bamboo 3.0 releases, due February 2011

- **Internet Explorer Notes:**
 - Atlassian intends to end support for IE7 in Bamboo 3.0. Bamboo 2.7 is the last version that will support IE7.
 - IE8 will still be supported.
 - 'Support End Date' means that Bamboo 2.7 and previously released versions will continue to work with IE7. However, we will not fix bugs affecting IE7 past the support end date.
 - Bamboo 3.0 will not be tested with IE7.

Bamboo system requirements

The system requirements for Bamboo are described in the following sections:

- [Hardware requirements & considerations](#)
- [Client/server software requirements](#)
- [Servlet requirements](#)

Hardware requirements & considerations



Note that Atlassian currently only supports Bamboo on x86 and 64 bit x86 derived hardware platforms.

Hardware considerations

For Bamboo, the minimum hardware requirements depend on the size and complexity of your plans. You should consider:

1. Will your builds have functional tests as part of the plans?
2. Are your plans executed simultaneously? If so, how many plans will be running at any given time?
3. What are the requirements for your running builds, for example do they need large amounts of memory/disk/swap space?
4. How many users will be using Bamboo at any given time? Like any web application, the system resource needed is proportional to the load experienced by the server.
5. How many local agents do you plan on running?

Storage considerations

The Bamboo installation size is approximately 140MB, however when running, Bamboo's storage requirements depend upon its usage pattern. The usage pattern further depends on factors such as:

- How many plans you will run.
- How many tests each plan will be executing.
- How many artifacts you are going to have and how large they are.

We recommend you allocate about 20GB on top of the Bamboo installation size, and evaluate your usage patterns. Where usage is likely to grow, consider adding additional storage.

Database connection pool size

The number of database connections available to Bamboo is the lower of two values: your DBMS connection limit and the configured Bamboo connection pool size. From Bamboo 4.2, the Bamboo connection pool size has a default value of 100.

For a small to medium instances (~5 concurrent users, ~5 busy/building local agents, 20 remote agents, 50 plans), the default values are sufficient.

You should increase the connection limit if you notice UI freezes or general sluggish UI performance. Do not decrease the number of available connections below 25.

Note: having too many connections available to Bamboo carries no performance penalty as long as your DBMS can handle the load.

Bamboo's connection limit can be modified by altering the following value in your `bamboo.cfg.xml` file:

```
<property name="hibernate.c3p0.max_size">100</property>
```

Local agents considerations

If you run more than 5 concurrently building local agents, note that each busy local agent requires a live database connection, so you'll probably need to adapt the connection limit.

Also, note that large amounts of busy (building) local agents can negatively influence the performance of a Bamboo server (and other services running on that host).

Remote (or elastic) agents considerations

Remote agents do not require special database connection settings.

Estimating the number of db connections

The following formula gives a rough estimate of the number of database connections that will be required:

$(\text{Concurrent users})/5 + (\text{Busy remote agents})/5 + (\text{Local agents}) * 1.1 + (\text{Amount of concurrent change detections})$

For example, an instance with:

- 5 concurrent users
- 30 busy remote (or elastic) agents
- 30 busy local agents
- 60 plans with repository polling set to 60 second intervals (assume 3 seconds per change detection)

would require $1 + 6 + 33 + 3 = 43$ connections.

Client/server software requirements

Bamboo is a pure Java application and should run on any platform, provided all the JDK requirements are satisfied. The [Supported Platforms](#) page lists the required server and client software, and their versions, supported by **Bamboo 5.0.x**

Bamboo 4.4.x

, however a brief summary can be seen in the table below:

	Browsers	Java Platform	Operating Systems	Application Servers	Databases	Source Repositories
Server side		• Oracle JDK • OpenJDK	• MS Windows • MacOS X • Linux	• Apache Tomcat	• MySQL • PostgreSQL • MS SQL Server • Oracle • HSQLDB	• Mercurial • Subversion • Git • Perforce • CVS
Client side	• Internet Explorer • Firefox • Safari • Chrome					

Browser

If you have disabled JavaScript in your browser or are using a script blocking tool like NoScript, you must enable your browser to execute JavaScript to access Bamboo's full functionality.

Java

Bamboo requires a full **Java Developers Kit (JDK)** platform to be installed on your server's operating system.

Application Server

Bamboo is a web application that requires an application server. Currently Apache Tomcat is supported. Tomcat is a stable, lightweight and fast performing application server, however, please note the following:

1. Deploying multiple Atlassian applications in a single Tomcat container is **not supported**. We do not test this configuration and upgrading any of the applications (even for point releases) is likely to break it. There are also a number of known issues with this configuration (see [this FAQ](#) for more information).
2. We also do not support deploying multiple Atlassian applications to a single Tomcat container for a number of practical reasons. Firstly, you must shut down Tomcat to upgrade any application and secondly, if one application crashes, the other applications running in that Tomcat container will be inaccessible.
3. Finally, we recommend not deploying *any other applications* to the same Tomcat container that runs Bamboo, especially if these other applications have large memory requirements or require additional libraries in Tomcat's `lib` subdirectory.

Database

Bamboo requires a relational database to store its data. Bamboo supports most popular relational database servers, so we suggest using the

one that you are most comfortable with administering. Bamboo ships pre-configured with an integrated HSQL database for evaluation purposes only. Since HSQLDB is prone to database corruption, we recommend configuring an external database for production environments.

Hence, if you intend to use Bamboo in a production environment, we **strongly recommend** that you connect Bamboo to an [enterprise database \(supported by Atlassian\)](#).

Servlet requirements

The Bamboo EAR-WAR distribution requires a servlet container that supports the Servlet 2.4 specification.

Hardware sizing considerations



Also see the [Bamboo system requirements](#).

On this page:

- [General information](#)
- [Database connection pool size](#)
- [Local agents considerations](#)
- [Remote \(or elastic\) agents considerations](#)
- [Estimating the number of db connections](#)

General information

For Bamboo, the minimum hardware requirements depend on the size and complexity of your plans. Considerations include:

1. Will your builds have functional tests as part of the plans?
2. Are your plans executed simultaneously? If so, how many plans will be running at any given time?
3. What are the requirements for your running builds, e.g. do they need large amounts of memory/disk/swap space?
4. How many users will be using Bamboo at any given time? Like any web application, the system resource needed is proportional to the load experienced by the server.
5. How many local agents do you plan on running?

Database connection pool size

The number of database connections available to Bamboo is the lower of two values: your DBMS connection limit and the configured Bamboo connection pool size. From Bamboo 4.2, the Bamboo connection pool size has a default value of 100. For a small to medium instances (~5 concurrent users, ~5 busy/building local agents, 20 remote agents, 50 plans), the default values are sufficient.

You should increase the connection limit if you notice UI freezes or general sluggish UI performance. Do not decrease the amount of available connection below 25.

Note: having too many connections available to Bamboo carries no performance penalty as long as your DBMS can handle the load.

Bamboo's connection limit can be modified by altering the following value in your bamboo.cfg.xml file:

```
<property name="hibernate.c3p0.max_size">100</property>
```

Local agents considerations

If you run more than 5 concurrently building local agents, note that each busy local agent requires a live database connection, so you'll probably need to adapt the connection limit.

Also, note that large amounts of busy (building) local agents can negatively influence the performance of a Bamboo server (and other services running on that host).

Remote (or elastic) agents considerations

Remote agents do not require special database connection settings.

Estimating the number of db connections

The following formula gives a rough estimate of the number of database connections that will be required:

$(\text{Concurrent users})/5 + (\text{Busy remote agents})/5 + (\text{Local agents}) * 1.1 + (\text{Amount of concurrent change detections})$

For example, an instance with:

- 5 concurrent users
- 30 busy remote (or elastic) agents
- 30 busy local agents
- 60 plans with repository polling set to 60 second intervals (assume 3 seconds per change detection)

would require $1 + 6 + 33 + 3 = 43$ connections.

Bamboo installation guide

1. Check the system requirements

Supported platforms

Please read the [Supported platforms](#) page before you install Bamboo. The Supported Platforms page lists the applications servers, databases, operating systems, web browsers and JDKs that we have tested Bamboo with and recommend.

Note that Bamboo ships with a built-in HSQL database, which is fine for evaluation purposes but is somewhat susceptible to data loss during system crashes. For production environments we recommend that you configure Bamboo to use an [external database](#).

Hardware requirements

While some of our customers run Bamboo on SPARC-based hardware, Atlassian only officially supports Bamboo running on x86 hardware and 64-bit derivatives of x86 hardware.

Servlet container requirements

If you are using the Bamboo EAR-WAR distribution, you will need a servlet container that supports the Servlet 2.4 specification. Most modern containers should comply with this.

On this page:

1. [Check the system requirements](#)
 2. [Choose your Bamboo distribution](#)
 3. [Installation and setup](#)
- [Checking for known issues and troubleshooting the Bamboo installation](#)

Related pages:

- [Running the Setup Wizard](#)
- [Upgrade guide](#)
- [Bamboo remote agent installation guide](#)
- [Release notes](#)

2. Choose your Bamboo distribution

Bamboo is available in two 'distributions' — Bamboo or Bamboo EAR-WAR. The Bamboo distribution is recommended (even for organisations with an existing application server environment).

Bamboo distribution

- Pre-packaged with the Jetty application server
- Requires virtually no setup
- Recommended for all users

Bamboo EAR-WAR distribution

- Deploys into an existing application server
- Requires manual configuration
- Suitable only for system administrators

3. Installation and setup



Notes for when the JDK is 64-bit:

If you're installing Bamboo 3.0 or later and choose to download the ZIP Archive distribution (Windows) or TAR.GZ Archive distribution (Linux) or TGZ Archive distribution (Mac OS X), the 32-bit wrapper will be included, and Bamboo may not start as expected. So after extracting Bamboo from the compressed file, please follow these steps:

1. Download the 64-bit wrapper:
 - a. [For Windows](#).
 - b. [For Linux](#).
 - c. For Mac(Not yet available).
2. Extract it from the compressed file.

3. Delete the existing <Bamboo installation directory>/wrapper directory.
4. Copy the extracted wrapper directory to <Bamboo installation directory>.

Installation and setup for Bamboo distribution

- [Installation Guide — Linux](#)
- [Installation Guide — Mac](#)
- [Installation Guide — Windows](#)

Installation and setup for Bamboo EAR-WAR distribution

- [EAR-WAR Installation Guide](#)

Checking for known issues and troubleshooting the Bamboo installation

If something is not working correctly after you have completed the steps above to install Bamboo, please check for known Bamboo issues and try troubleshooting your upgrade as described below:

- **Check for known issues.** Sometimes we find out about a problem with the latest version of Bamboo after we have released the software. In such cases we publish information about the known issues in the Bamboo Knowledge Base. Please check the [known issues](#) in the Bamboo Knowledge Base and follow the instructions to apply any necessary patches if necessary.
- **Did you encounter a problem during the Bamboo installation?** Please refer to the guide to [troubleshooting upgrades](#) in the Bamboo Knowledge Base.
- If you encounter a problem during the upgrade and cannot solve it, please create a [support ticket](#) and one of our support engineers will help you.

Bamboo installation guide for Linux

This page contains instructions to help you install Bamboo on Linux. If you want to use your application server, rather than the bundled Jetty server, see the [Bamboo EAR-WAR installation guide](#) instead.

Note that Bamboo ships with a built-in HSQL database, which is fine for evaluation purposes but is somewhat susceptible to data loss during system crashes. For production environments we recommend that you configure Bamboo to use an [external database](#).



Before you begin

Please ensure that you have read the [Requirements](#) section of the Bamboo Installation Guide.

On this page:

1. [Download and install Bamboo](#)
2. [Launch Bamboo on Linux](#)
3. [Configure Bamboo](#)

Related pages:

- [Bamboo EAR-WAR installation guide](#)
- [Bamboo installation guide for Mac](#)
- [Bamboo installation guide for Windows](#)

1. Download and install Bamboo

1. Download Bamboo for Linux. Bamboo for Linux is available for download from the [Bamboo Download Center](#) (click the '**Linux**' tab).
2. Extract the files from the Linux archive to a **Bamboo installation directory** of your choice. By default, the root directory of the tar file is "Bamboo".
3. Set up your **Bamboo home directory** — this is the directory where Bamboo will store its configuration data. To do this, open the file named `bamboo-init.properties` in the <Bamboo installation directory>/webapp/WEB-INF/classes directory. In this file, insert the property "bamboo.home", with an absolute path to your Bamboo home directory. Your file should look something like this:

```
bamboo.home=/test/bamboo-home
```



You must use forward-slashes in your directory path. Backslashes are not recognised by Bamboo. Please ensure that the **Bamboo home directory** is not located inside the **Bamboo installation directory**

Alternatively, you can specify an environment variable 'BAMBOO_HOME' which specifies the absolute path to your {BAMBOO_HOME} directory. Bamboo will check if an environment variable is defined.

2. Launch Bamboo on Linux

There are two ways you can launch Bamboo on Linux — using either a startup script or a Java Service Wrapper:

Launch via `bamboo.sh` startup script

You can start Bamboo with the default `bamboo.sh` file in your installation root directory. The `bamboo.sh` command accepts the following options (e.g. `./bamboo.sh start`):

- `start` — this starts Bamboo.
- `stop` — this stops Bamboo.
- `restart` — this restarts Bamboo
- `status` — this provides the current status of Bamboo.

Launch via Java Service Wrapper



The wrapper is platform-specific and doesn't work on SunOS.

You can also start Bamboo using a Java Service Wrapper, which provides services such as automatic restarting. To do this, you will need to use the `start-bamboo` command available in the `/wrapper` folder of the Bamboo installation. You need to run the command with one of the following options (e.g. `./start-bamboo start`):

- `console` — this starts Bamboo in a console. The logs will scroll to standard out.
- `start` — this starts Bamboo.
- `stop` — this stops Bamboo.
- `restart` — this restarts Bamboo
- `status` — this provides the current status of Bamboo.
- `dump` — stops Bamboo abruptly by killing the process



If you have installed Bamboo on a machine with multiple interfaces, and need to bind Bamboo to a single IP address, please see [Binding Bamboo to one IP address](#).

3. Configure Bamboo

1. Access your running Bamboo instance by going to your web browser and entering the address: <http://localhost:8085/>.
2. Configure Bamboo using the Setup Wizard that is displayed. Read [Running the Setup Wizard](#) for further instructions.

Bamboo installation guide for Mac

This page contains instructions to help you install Bamboo on Mac OS X. If you want to use your application server, rather than the bundled Jetty server, see [Bamboo EAR-WAR installation guide](#) instead.

Note that Bamboo ships with a built-in HSQL database, which is fine for evaluation purposes but is somewhat susceptible to data loss during system crashes. For production environments we recommend that you configure Bamboo to use an [external database](#).



Before you begin

Please ensure that you have read the [Requirements section](#) of the Bamboo Installation Guide.

On this page:

1. [Download and Install Bamboo](#)
2. [Launch Bamboo on Mac OS X](#)
3. [Configure Bamboo](#)

Related pages:

- [Bamboo EAR-WAR installation guide](#)
- [Bamboo installation guide for Linux](#)
- [Bamboo installation guide for Windows](#)

1. Download and Install Bamboo

You can choose to install Bamboo via a **Mac OS X Installer (.dmg)** or a **TGZ Archive (.tgz)**:

Download and install Bamboo for Mac OS X (Mac OS X Installer)

1. Bamboo for Mac OS X is available for download from the [Bamboo Download Center](#) (click the **Mac OS X** tab if necessary). Choose the **Mac OS X Installer (.dmg)** download.
2. Launch the Bamboo Mac OS X installer. This will mount the Atlassian Bamboo installation volume.
3. Launch the Bamboo Continuous Integration Server Installer.app to begin the installation wizard. The installer requires you to specify two directories:
 - **Bamboo installation directory**— This is the directory where Bamboo's application files will be installed. The default is:


```
/Applications/Bamboo
```

- **Bamboo home directory**— This is the directory where Bamboo will store its configuration data. If the directory you specify doesn't exist, Bamboo will create the directory when it launches. The default is:

```
/Users/<current-user>/bamboo-home
```



You must use forward-slashes in your directory path. Backslashes are not recognised by Bamboo. Please ensure that the **Bamboo home directory** is not located inside the **Bamboo installation directory**.

Download and install Bamboo for Mac OS X (TGZ Archive)

1. Bamboo for Mac OS X is available for download from the [Bamboo Download Center](#) (click the **Mac OS X** tab). Choose the **TGZ Archive (.tgz)** download (click **Show all** to show the **TGZ Archive** download link).
2. Extract the files from the Mac OS X archive version to a Bamboo installation directory of your choice. By default, the root directory of the tgz file is "Bamboo".
3. Set up your **Bamboo home directory** — this is the directory where Bamboo will store its root configuration data. To do this, open the file named `bamboo-init.properties` in the `<Bamboo installation directory>/webapp/WEB-INF/classes` directory. In this file, insert the property "bamboo.home", with an absolute path to your Bamboo home directory. Your file should look something like this:

```
bamboo.home=/test/bamboo-home
bamboo.jms.broker.uri=tcp://localhost:54663
```

Alternatively, you can specify an environment variable 'BAMBOO_HOME' which specifies the absolute path to your {BAMBOO_HOME} directory. Bamboo will check if an environment variable is defined.

4. If you are going to use Bamboo remote agents, set the following in the `bamboo-init.properties` file in the `<Bamboo installation directory>/webapp/WEB-INF/classes` directory:
 - Replace 'localhost' with the real host name or IP address of your Bamboo server.
 - If port number 54663 is already in use, specify a different port number.

2. Launch Bamboo on Mac OS X

There are two ways you can launch Bamboo on Mac OS X:

Launch using the `bamboo.sh` startup script

You can start Bamboo with the default `bamboo.sh` file in your installation root directory. In a terminal window, type: `/Applications/Bamboo/bamboo.sh start`

The `bamboo.sh` command accepts the following options:

- `console` — this starts Bamboo in a console. The logs will scroll to standard out.
- `start` — this starts Bamboo.
- `stop` — this stops Bamboo.
- `status` — this provides the current status of Bamboo.

Launch using the Java Service Wrapper

Alternatively, you can start Bamboo using a Java Service Wrapper, which provides services such as automatic restarting. To do this, use the `run-bamboo` command available in the `/wrapper` folder of the Bamboo installation, by typing the following in a terminal window: `/Applications/Bamboo/wrapper/run-bamboo start`

The `run-bamboo` command accepts the following options:

- `console` — this starts Bamboo in a console. The logs will scroll to standard out.
- `start` — this starts Bamboo.
- `stop` — this stops Bamboo.
- `status` — this provides the current status of Bamboo.

3. Configure Bamboo

1. Access your running Bamboo instance by going to your web browser and entering the address: <http://localhost:8085/>.
2. Enter a license code.

3. Configure Bamboo using the Setup Wizard. Read [Running the Setup Wizard](#) for further instructions.

Bamboo installation guide for Windows

This page contains instructions to help you install Bamboo on Windows. If you want to use your application server, rather than the bundled Jetty server, see [EAR-WAR Installation Guide](#) instead.

Note that Bamboo ships with a built-in HSQL database, which is fine for evaluation purposes but is somewhat susceptible to data loss during system crashes. For production environments we recommend that you configure Bamboo to use an [external database](#).

1. Download and install Bamboo

Before you begin:

- Please ensure that you have read the [Requirements section](#) of the Bamboo Installation Guide.
- Note, you can choose to install Bamboo via a **Windows Installer(.exe)** or a **ZIP Archive (.zip)**.

Download and install Bamboo for Windows (Windows Installer)

 *Note, if you wish to run Bamboo on a **Windows x64 platform**, make sure that you download the 64-bit version of Bamboo distribution and have a 64-bit JDK installed*

1. Download Bamboo for Windows. Bamboo for Windows is available for download from the [Bamboo Download Center](#). Choose the **Windows Installer (.exe)** download.
2. Launch the Bamboo Windows installer to begin the installation wizard.
3. The installer requires you to specify two directories:
 - **Destination directory**— This is the directory where Bamboo's application files will be installed. The default is:

```
C:/Program Files/Bamboo
```

On this page:

1. Download and install Bamboo
2. Launch Bamboo
3. Configure Bamboo

Related pages:

- [Bamboo EAR-WAR installation guide](#)
- [Bamboo installation guide for Linux](#)
- [Bamboo installation guide for Mac](#)

- **Bamboo home directory**— This is the directory where Bamboo will store its configuration data. If the directory you specify doesn't exist, Bamboo will create the directory when it launches. The default is:

```
C:/Documents and Settings/<current-user>/Bamboo-home
```



You must use forward-slashes in your directory path. Backslashes are not recognised by Bamboo. Please ensure that the **Bamboo home directory** is not located inside the **Bamboo installation directory**

Download and install Bamboo for Windows (ZIP Archive)

1. Download Bamboo for Windows. Bamboo for Windows is available for download from the [Bamboo Download Center](#). Choose the **ZIP Archive (.zip)** download (click the **'Show all'** link to show the **'ZIP Archive'** download link).
2. Extract the files from the ZIP Archive to a **Bamboo installation directory** of your choice. By default, the root directory in your zip file is named "Bamboo".



Warning: Some unzip programs cause errors

Some archive-extract programs cause errors when unzipping the Bamboo archive file. We highly recommend that you use the free [7Zip](#) archive-extract program (if in doubt, download the '32-bit .exe' version).

3. Set up your **Bamboo home directory** — this is the directory where Bamboo will store its root configuration data. To do this, edit the file named `bamboo-init.properties` in the `Bamboo/webapp/WEB-INF/classes` directory. In this file, insert the property "bamboo.home", with an absolute path to your Bamboo home directory. Your file should look something like this:

```
bamboo.home=C:/test/bamboo-home
```

Alternatively, you can specify an environment variable 'BAMBOO_HOME' which specifies the absolute path to your {BAMBOO_HOME} directory. Bamboo will check if an environment variable is defined.

4. If you are going to use Bamboo remote agents, set the following in the `bamboo-init.properties` file in the <Bamboo installation directory>/webapp/WEB-INF/classes directory:

```
bamboo.jms.broker.uri=tcp://localhost:54663
```

- Replace 'localhost' with the real host name or IP address of your Bamboo server.
- If port number 54663 is already in use, specify a different port number.

2. Launch Bamboo

There are two ways you can launch Bamboo on Windows:

Launch via the Start Menu

If you have used the 'Windows Installer' to install Bamboo, you can start Bamboo via the Start Menu in Windows (generally under the 'Bamboo' folder by default). The following options will be available in your Start Menu:

- **Bamboo Continuous Integration Server Uninstaller** — uninstalls Bamboo from your computer
- **Install Service** — installs Bamboo as a Windows service (note, this will not start Bamboo)
- **Remove Service** — removes the Bamboo Windows service, if you have previously installed it (note, Bamboo will not be uninstalled from your computer)
- **Start in Console** — starts Bamboo in a Windows console
- **Start Service** — starts your installed Bamboo Windows service
- **Stop Service** — stops your installed Bamboo Windows service

You can run Bamboo in two modes, either in a Windows console or as a Windows service:

- **To run Bamboo in a Windows console**, click the **Start in Console** option.
- **To run Bamboo as a Windows service**, click the **Install service** option. After the service is installed, click **Start Service**. Once you have installed Bamboo as a service, Bamboo will start up automatically every time Windows restarts.



Running Bamboo as a service

- **Changing the user running the Bamboo service** — The default behaviour in Windows is to start the service under the **SYSTEM** user when Bamboo starts up. You can change this behaviour from the **Services** console available via the **Control Panel**. Please note, you may need to uninstall the service and re-install it again, if you have just upgraded or re-installed Bamboo.
- **Running Bamboo as a service in Windows Vista** — Bamboo ships with a service wrapper in Windows and by default, the wrapper installs itself as the NT SYSTEM user. However, in Vista the temporary directory System Variable is not available to untrusted apps. In order to run Bamboo as service in Vista, you must run Bamboo as a non-system user — see [Running Bamboo service on Windows as the local user](#) for further instructions.

Launch via batch file

You can start Bamboo via the batch files that are shipped with Bamboo. If you have installed Bamboo via the ZIP Archive, you will need to use the batch files to start Bamboo. You can find the following batch files in your installation directory:

- **BambooConsole.bat** — this starts Bamboo in a Windows console.
- **InstallAsService.bat** — this installs Bamboo as a Windows service. Note that this will not start Bamboo.
- **StartBamboo.bat** — this starts your installed Bamboo Windows service.
- **StopBamboo.bat** — this stops your installed Bamboo Windows service
- **UninstallService.bat** — this un-installs the Bamboo Windows service from your machine. Note that your Bamboo installation still remains.

You can run Bamboo in two modes, either in a Windows console or as a Windows service:

- **To run Bamboo in a Windows console**, run **BambooConsole.bat**
- **To run Bamboo as a Windows service**, run **InstallAsService.bat**. After the service is installed, run **StartBamboo.bat**. Once you have installed Bamboo as a service, Bamboo will start up automatically every time Windows restarts.



Running Bamboo as a service

- **Changing the user running the Bamboo service** — The default behaviour in Windows is to start the service under the **SYSTEM** user when Bamboo starts up. You can change this behaviour from the **Services** console available via the **Control Panel**. Please note, you may need to uninstall the service and re-install it again, if you have just upgraded or re-installed Bamboo.

- **Running Bamboo as a service in Windows Vista** — Bamboo ships with a service wrapper in Windows and by default, the wrapper installs itself as the NT SYSTEM user. However, in Vista the temporary directory System Variable is not available to untrusted apps. In order to run Bamboo as service in Vista, you must run Bamboo as a non-system user — see [Running Bamboo service on Windows as the local user](#) for further instructions.

3. Configure Bamboo

Access your running Bamboo instance by going to your web browser and entering the address <http://localhost:8085/>. Configure Bamboo using the Setup Wizard that is displayed. Read [Running the Setup Wizard](#) for further instructions.

Running the Setup Wizard

When you launch Bamboo for the first time, the Bamboo setup wizard will display. The wizard will lead you through the Bamboo settings that you need to configure before you can start using it.



Before you begin

If you are currently using [Atlassian's Crowd](#) with Bamboo and wish to import existing data into Bamboo (see [Step 5. Starting Data](#) below), you will need to *disable Crowd* before starting the Setup Wizard. To do this, go to **Administration > User Repositories** (under 'Security') and choose **Local users and groups**.

You can then re-enable Crowd and restart Bamboo at the completion of the Setup Wizard.

Step 1. License Details and Setup Method

You must have a valid Bamboo license (evaluation or commercial) to use Bamboo. You can generate your own Bamboo evaluation license from your MyAtlassian self-service account [here](#). If you have any problems with this, please email [sales](#).

Once you have entered a valid license key, you can choose which setup method you prefer for your Bamboo installation:

Express Installation — we recommend that you choose this method if you are evaluating or demonstrating Bamboo.

- The 'Express Installation' method requires only a minimum of configuration information. It sets up Bamboo with default settings and an embedded database (HSQL).
- If you choose the 'Express Installation' method you can skip to [Step 6. Set Up Administrator User](#) below.

On this page:

[Step 1. License Details and Setup Method](#)
[Step 2. General Configuration](#)
[Step 3. Choose a Database Configuration](#)
[Step 4. Database Configuration](#)
[Step 5. Starting Data](#)
[Step 6. Set Up Administrator User](#)

Related pages:

- [Bamboo EAR-WAR installation guide](#)
- [Bamboo installation guide for Linux](#)
- [Bamboo installation guide for Mac](#)
- [Bamboo installation guide for Windows](#)

Custom Installation — we recommend that you choose this method if you are setting up a production instance of Bamboo.

- The 'Custom Installation' method takes longer, but allows you to configure Bamboo with an external database, customise the default settings, and/or initialise the server with your own data.
- If you choose, the 'Custom Installation' method, proceed to [Step 2. General Configuration](#) below.

Screenshot: License Details and Setup Method

Welcome to Atlassian Bamboo!

Welcome to Bamboo Continuous Integration Server. Please enter your license information and choose a setup method below to complete the installation of Bamboo.

Enter Your License

Server ID: BPBD-R68G-1V01-C2PP

License Key:

* sdaadwdKHDDlWakjowedkhjwadKSWkhjsKhkdsakjddsakh
asdlOSDASlFOSAosdoiaiodASDONDAOSndASDOSDOAasd
oidwowdadwAOAWoidwaOIDWQJAOISTEUNVjrioueiosa
askjhasdkhadskhjjasdhkaieQOIQOIoldazsIPlofLKPqwiaoI

Please enter your Bamboo license key above - either commercial or evaluation.
Please contact [Atlassian](#) if you require a license key.

Select Setup Method

Express Installation

Installs Bamboo with default settings and an embedded database. Recommended if you are evaluating or demonstrating Bamboo, as it will get you up and running as quickly as possible.

Express Installation >>

Custom Installation

Installs Bamboo but allows you to configure Bamboo with an external database, customise the default settings, and/or initialise the server with your own data. Recommended if you are setting up a production instance.

Custom Installation >>

Step 2. General Configuration

i This step applies to the 'Custom Installation' method only.

On this page you specify a number of Bamboo server settings, such as the address of the server, where data is stored and the message broker used to communicate with remote agents.

! You may find it simplest to keep the default settings for the three directory settings, in the table. For more information please see [Locating important directories and files](#).

Setting	Details
Name	See Specifying Bamboo's Title
Base URL	See Specifying Bamboo's URL
Configuration Directory	The location for Bamboo configuration files.
Build Data Directory	The location for Bamboo project data files.
Build Working Directory	The location of project files checked out from source control.
Broker URL	Only visible if you are permitted remote agents under your Bamboo license. The URL of the embedded messaging broker that Bamboo sets up to communicate with its remote build agents. This URL will be written to <code>bamboo.cfg.xml</code> as a property. You can update this file if you want to change your Broker URL. - Replace <code>localhost</code> with the real host name or IP address of your Bamboo server. You should not use <code>localhost</code> as the host name in the Broker URL, as remote agents are provided with the Broker URL on startup and use it to communicate to the server. - If port number 54663 is already in use, specify a different port number.

Screenshot: General Configuration

General Configuration

Please specify the following settings for the Bamboo Server.

What is the title of this Bamboo instance?	
Name:	Atlassian Bamboo
The name of this Bamboo instance.	
What is the server's address?	
Base URL:	http://testuser:8086/bamboo
This is the base URL of this installation of Bamboo. All links created (for emails etc) will be prefixed by this URL. For example "http://brydie:8086/bamboo"	
System Paths and Directories	
Configuration Directory:	/Users/testuser/dev/src/bamboo-trunk-test-mc/bamboo-homes/8086/xml-data/configuration
The configuration directory is where your settings for Bamboo will be stored. For example "C:/bamboo-home/xml-data/configuration".	
Build Data Directory:	/Users/testuser/dev/src/bamboo-trunk-test-mc/bamboo-homes/8086/xml-data/builds
The build directory is where Bamboo saves the build data such as results and artifacts. For example "C:/bamboo-home/xml-data/builds".	
Build Working Directory:	/Users/testuser/dev/src/bamboo-trunk-test-mc/bamboo-homes/8086/xml-data/build-dir
Specify the absolute path on the server. For example "C:/bamboo-home/xml-data/build-dir".	
Remote Agent Communication	
Broker URL:	tcp://testuser.sydney.atlassian.com:54663?wireFormat.maxInactivityDuration=300000
The url on which your messaging broker will be set up. The messaging broker is used for communication with Bamboo remote agents. Bamboo remote agents will also use this path to communicate back to this server.	
Continue	

Step 3. Choose a Database Configuration

i This step applies to the 'Custom Installation' method only.

Picking a database configuration is an important choice. If you pick the 'Embedded Database' configuration, you do not have to set up a database. However, the embedded HSQL database is **only suitable for evaluation purposes**. You will need to move to an external database, if you decide to deploy Bamboo in production at a later stage (as described in [Moving your Bamboo data to a different database](#)).

Choose one of the following:

- **Embedded Database** — Choose this for quick and easy first-time installation of Bamboo. This option is suitable for evaluation purposes only. Skip to [Step 5. Starting Data](#).
- **External Database** — Choose this if you wish to use an external database. Proceed to [Step 4. Database Configuration](#) below.

Screenshot: Choose a Database Configuration

Choose a Database Configuration	
Choose how Bamboo will connect to your database.	
Select Database	
☒	Embedded Database The embedded database will allow Bamboo to operate without an external database
☐	External Database If you wish to store your Bamboo data in an external database, choose it from the list of supported databases. This is recommended for production systems. If your database is not listed in the menu, you may configure an 'Unsupported Database', but be aware that Bamboo may not be fully tested
	PostgreSQL 8.2 and above
Continue	

Step 4. Database Configuration

i This step applies to the 'Custom Installation' method only.

If you selected 'External Database' in Step 3, you will need to provide the configuration details for your database. Please see [Connecting Bamboo to an external database](#) for further instructions.

Screenshot: Database Configuration

Database Configuration

Choose how you wish Bamboo to connect to your database

Select Database Connection

- ☒ Direct JDBC connection
☐ Connect via a datasource (configured in the application server)

Driver Class Name:
The class name of the database driver. Ensure that this class is in your class path.

Database URL:
The url to access the database.

User Name:
The username to access the database.

Password:
(optional) Enter the password if the database configuration requires it.

☐ Overwrite existing data
If you wish Bamboo to overwrite any existing tables that may exist in the database.

Continue

Step 5. Starting Data

i This step applies to the 'Custom Installation' method only.

Screenshot: Starting Data

Starting Data

Select Starting Data for Bamboo

Would you like to:

☒ Create a new Bamboo home
☐ Import existing data

Continue

On this page you specify how Bamboo will populate the 'home directory' that you set up when you [installed Bamboo](#).

Choose one of the following:

- **Create new Bamboo home** — choose this if you are performing a normal installation or upgrade.
- **Import existing data** — only choose this under exceptional circumstances, e.g. if you are [connecting Bamboo to a different database](#), or moving your pre-existing Bamboo installation to a different server. Avoid importing backups from different versions of Bamboo.

Step 6. Set Up Administrator User

The final step of the setup wizard is to enter the details of the first registered user for the Bamboo system. This user will have [global administrative privileges](#) over the entire installation of Bamboo and should not be removed.

Once you have entered the details for your administrator user, click **Finish**. The [Bamboo dashboard](#) will be displayed.

Congratulations, you have successfully set up Bamboo!

Screenshot: Set Up Administrator User

Set Up Administrator User

Please enter the details of the administrator user for this installation of Bamboo.

Administrator User Details

Username: *
The name to use to log in to Bamboo.

Password: *

Confirm Password: *

Full Name: *

Email: *

Finish

Bamboo EAR-WAR installation guide

The Bamboo EAR-WAR distribution is intended for deployment into an existing J2EE application server. It is assumed that you already know how to deploy a web application on the application server of choice. If not, we recommend that you install the [Bamboo distribution](#).

The following instructions are only indicative of the process and examples are based on installing the Bamboo WAR file on the [Apache Tomcat](#) application server. Deployment and configuration will differ from the procedure below if you choose to deploy the Bamboo EAR-WAR distribution on another (unsupported) application server.

Before you begin:

- Please ensure that you have read the [Requirements](#) section of the Bamboo Installation Guide.
- Deploying multiple Atlassian applications in a single Tomcat container is **not supported**. We do not test this configuration and upgrading any of the applications (even for point releases) is likely to break it. There are also a number of known issues with this configuration (see [this FAQ](#) for more information).

We also do not support deploying multiple Atlassian applications to a single Tomcat container for a number of practical reasons. Firstly, you must shut down Tomcat to upgrade any application and secondly, if one application crashes, the other applications running in that Tomcat container will be inaccessible.

Finally, we recommend not deploying *any other applications* to the same Tomcat container that runs Bamboo, especially if these other applications have large memory requirements or require additional libraries in Tomcat's `lib` subdirectory.

On this page:

1. Download and install Bamboo EAR-WAR
2. Set the Bamboo Home
3. Set `jms.broker.uri`
4. Add these other parameters into `JAVA_OPTS`
5. (Optional) Edit the Bamboo Context Descriptor (Tomcat)
6. Configure Bamboo

Related pages:

- [Bamboo installation guide for Linux](#)
- [Bamboo installation guide for Mac](#)
- [Bamboo installation guide for Windows](#)

1. Download and install Bamboo EAR-WAR

1. Download Bamboo WAR for your operating system. Bamboo WAR files for Linux, Mac OS X and Windows are available for download from the [Bamboo Download Center](#).
2. Create a new directory for the Bamboo instance inside the `<Tomcat Installation Directory>\webapps` folder and extract the WAR file into this new directory. This directory is referred to as the [installation directory](#) (i.e. `<Bamboo-Install>`)



Extracting the Bamboo WAR

- **Windows** users must avoid Win XP's built-in unzip as it doesn't extract all the files. Use a third-party zip extractor like [WinZip](#).
- **Linux** and **Mac** users can use the command below:

```
jar -xvf atlassian-bamboo-X.Y.Z.war
```

Where X.Y.Z is the Bamboo distribution version number.



The name of the directory in the `webapps` folder will form the URL required to access Bamboo (e.g. `<Tomcat Installation Directory>\webapps\bamboo-4.4.5/` will become `http://host:port/bamboo-4.4.5/`). You may wish to change the directory name for a more concise access URL.

2. Set the Bamboo Home

Please choose **only one** of the three methods below in order to set your Bamboo Home directory. You may also want to check if Bamboo Home is not already defined in one of the methods below, to avoid having Bamboo Home defined in multiple places.

Method 1: Set the `bamboo.home` property in the file `<Bamboo-Install>/WEB-INF/classes/bamboo-init.properties` to your chosen Bamboo home directory.

Method 2: Pass the Bamboo home directory to the application server:

Doing this on **Windows**:

- a) Create or edit the `<Tomcat Installation Directory>\bin\setenv.bat` file.
- b) Create or edit the `JAVA_OPTS` variable inside `setenv.bat` and add the `-Dbamboo.home` parameter. It should look like this:

```
set JAVA_OPTS=-Dbamboo.home=C:/path/to/bamboo_home %JAVA_OPTS%
```

Doing this on **Linux** or **Mac**:

- a) Create or edit the <Tomcat Installation Directory>\bin\setenv.sh file.
- b) Create or edit the JAVA_OPTS variable inside setenv.sh and add the -Dbamboo.home parameter. It should look like this:

```
#!/bin/sh
#
# ENVARS for Tomcat and TDS environment
#
JAVA_OPTS="-Dbamboo.home=/opt/bamboo/bamboohome $JAVA_OPTS"
export JAVA_OPTS
```

- c) If you've created the <Tomcat Installation Directory>\bin\setenv.sh file, you need to grant execute permissions:

```
chmod +x setenv.sh
```

Method 3: Specify an environment variable BAMBOO_HOME which specifies the absolute path to your BAMBOO_HOME directory.

3. Set jms.broker.uri

If you are going to use Bamboo Remote Agents, set the following in the <Bamboo_Install>/WEB-INF/classes/bamboo-init.properties file:

```
bamboo.jms.broker.uri=tcp://localhost:54663
```

- Replace 'localhost' with the real host name or IP address of your Bamboo server.
- If port number 54663 is already in use, specify a different port number.

4. Add these other parameters into JAVA_OPTS

You have to set the following Java OPTs on your application server:

- -server — Ensures that the jvm starts up in server mode. This will perform various optimisation tasks which are beneficial for long-running applications.
- -Xmx512m — Sets the maximum memory recommended for Bamboo.
- -XX:MaxPermSize=256m — Sets the maximum permgen memory recommended for Bamboo.
- -Djava.awt.headless=true — *(Unix systems only)* This allows AWT to run in headless mode and is required if running Bamboo in non-graphical environments. For more details visit the [Sun Developer Network](#).

On Windows:

1. Open the <Tomcat Installation Directory>\bin\setenv.bat file.
2. Assign the desired properties to the JAVA_OPTS variable:

```
set JAVA_OPTS=-server -Xmx512m -XX:MaxPermSize=256m
-Dbamboo.home=C:/path/to/bamboo_home %JAVA_OPTS%
```

 Delete the Bamboo Home definition from the JAVA_OPTS above if you've defined it using a different method (as explained in Step 2).

On Linux and Mac:

1. Open the <Tomcat Installation Directory>\bin\setenv.sh file.
2. Assign the desired properties to the JAVA_OPTS variable:

```
JAVA_OPTS="-server -Xmx512m -XX:MaxPermSize=256m -Djava.awt.headless=true
-Dbamboo.home=/opt/bamboo/bamboohome $JAVA_OPTS"
export JAVA_OPTS
```

 Delete the Bamboo Home definition from the JAVA_OPTS above if you've defined it using a different method (as explained in Step 2).

5. (Optional) Edit the Bamboo Context Descriptor (Tomcat)

If you have extracted the WAR file to a directory other than the default directory (e.g. for a Tomcat application server), you need to deploy <bamboo-install> by following the steps below:

1. Create a file called `bamboo.xml` in your Tomcat installation's `conf/Catalina/localhost` directory. If you have set up a different hostname for your Tomcat instance, please use your specified hostname instead of `localhost`.

 Please note, if you are using Tomcat 6 you must create the `Catalina` and `localhost` directories.

2. Open your new `bamboo.xml` and add the following:

```
<Context path="/bamboo" docBase="c:/applications/bamboo-war/" debug="0"
reloadable="true">
</Context>
```

 If you wish to change the context path to a different name, change both the context path and the name of the xml file.

3. For `docBase`, specify the <bamboo-install> absolute path that you noted down earlier.



If you are installing Bamboo 3.4 or newer versions

If installing Bamboo 3.4, or newer versions, please make sure that you apply the Tomcat configuration suggested in [this KB Article](#).

6. Configure Bamboo

1. Start Tomcat.
2. Access your running Bamboo instance on <http://host:port/bamboo> (please note that the URL may vary as explained in Step 1)
 The default port used by Tomcat is 8080. If you have other applications running on this same port, please edit the default port that Tomcat should use, by editing the port parameter for the Connector element in <Tomcat Installation Directory>\conf\server.xml:

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="30000"
redirectPort="8443" />
```

3. Configure Bamboo via the Setup Wizard which will display. Read [Running the Setup Wizard](#) for further instructions.

Configuring Tomcat to Use HttpOnly Session ID Cookies

[Bamboo distributions](#) from version 2.5.5 now enforce the [HttpOnly](#) flag on session ID cookies by default, as a means to minimise the risk of common XSS attacks. For more information about this feature, please refer to the [Bamboo Security Advisory 2010-05-04](#).

If you are running the Bamboo EAR-WAR distribution on Tomcat (or another application server that is unsupported), it is likely that Bamboo's session ID cookies will not be transmitted with the `HttpOnly` flag. To reduce the risk of common XSS attacks, we recommend that you configure your application server to transmit `HttpOnly` session ID cookies.

To configure Bamboo EAR-WAR distribution running on Tomcat to use HttpOnly Session ID Cookies:

1. Shutdown the Bamboo service running on Tomcat and the Tomcat application server.
2. Open the `context.xml` file of the Tomcat installation running Bamboo in a text editor.
 This file is typically located in the `conf` subdirectory of the main Tomcat installation directory.
3. Add the following `Manager` element within the `Context` element of this file:

```
...
<Context>
  ...
  <Manager useHttpOnly="true"/>
  ...
</Context>
...
```

 To disable `HttpOnly` Session ID cookies, either remove this `Manager` element or change the value of its `useHttpOnly` parameter to `false`.

4. Save your changes to the `context.xml` file.
5. Restart Bamboo.

Bamboo remote agent installation guide

This page describes how to install the Bamboo Remote Agent manually.

 Bamboo remote agents are available only for [Downloaded Bamboo](#) - NOT for OnDemand Bamboo because of the [OnDemand restrictions](#).

Before you begin:

- **Not sure whether to install a remote agent?** See [Agents and capabilities](#) to understand how remote agents interact with your Bamboo server.
- **Do you have sufficient agent licenses?** See [Bamboo licensing](#) for details.
- **Ensure that your system meets the system requirements.** See [Supported platforms](#).
- **Ensure that you have a supported version of Java** installed on the agent machine. See [Supported platforms](#).
- **Have you enabled the creation of remote agents**, as described in [Disabling and enabling remote agents support](#).
- **Ensure that you have specified the Broker URL**, as described in the [Bamboo Setup Wizard](#) and the [Bamboo 2.0 Upgrade Guide](#).
- **Have you implemented your own remote agent service wrapper?** You may not want to use the [remote agent supervisor](#) that is bundled with the remote agent. You can choose to install the legacy remote agent (pre-Bamboo 2.2) instead, which does not have a service wrapper.

On this page:

- Step 1. Download and install the remote agent
- Step 2. Launch the remote agent
- Step 3. Configure the remote agent's capabilities
- Step 4. (Optional) Rename the remote agent

Related pages:

- [Configuring remote agent capabilities using bamboo-capabilities.properties](#)
- [Legacy remote agent installation guide](#)

Step 1. Download and install the remote agent

1. Create a directory on the agent machine (e.g. `bamboo-agent-home`) to serve as the [Bamboo agent home](#) for the remote agent.
2. On your Bamboo server, click **Administration** in the menu bar.
3. Click **Agents** in the left panel. This will display the 'Agents' screen, showing lists of all local agents and all remote agents that currently exist on your Bamboo system.
4. Click **Enable Remote Agent Support**, if necessary, and then **Install Remote Agent**.
5. Click **DOWNLOAD Remote Agent JAR** and save the JAR file to the directory on the agent machine that you created in step 1.1.
6. Take note of the command under the heading 'Running a Remote Agent' for use in step 2 below.

Step 2. Launch the remote agent

Once installed, you can run the remote agent by executing the command line obtained in the previous step. This command will look something like the following:

```
java -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar  
http://bamboo-host-server:8085/agentServer/
```

 The name of the jar file (e.g. `atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar`) will vary depending on the version of Bamboo you are running.

You can also choose to run the remote agent with different command line parameters, to change where the remote agent stores its data or to suppress the self-signed certificate of the server.



If you are having issues launching the agent, then take a look at our [troubleshooting guide](#).

Changing where the remote agent stores its data

By default, the remote agent will store its data in a `USER_HOME/bamboo-agent-home`. If you wish to specify a different directory, add the following command line parameter before the JAR file name:

```
-Dbamboo.home=RemoteAgentHome
```

where `RemoteAgentHome` is the path to the Bamboo agent home directory you created in step 1.1.
Your command line will look something like this:

```
java -Dbamboo.home=RemoteAgentHome -jar  
atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar  
http://bamboo-host-server:8085/agentServer/
```

 The name of the jar file (e.g. `atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar`) will vary depending on the version of Bamboo you are running.

Disabling auto-capability detection for the remote agent

There may be situations where you want to prevent Bamboo from automatically detecting and adding capabilities (such as JDKs) to the remote agent, or where you don't want to run the remote agent with default capabilities.

To disable auto-capability detection for the remote agent, restart the agent with the following command line parameter before the JAR file name:

```
java -Ddisable_agent_auto_capability_detection=true -jar  
atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar  
http://bamboo-host-server:8085/agentServer/
```

Changing the logging on the remote agent

By default, the remote agent will use the same logging level as the Bamboo server. However, you can control the level of logging of your remote agent independently of your Bamboo server by setting up a separate logging configuration file.

Please see [Logging in Bamboo](#) for further details.

Suppressing the self-signed certificate of the server

If your Bamboo server uses SSL (https) with a self-signed certificate, you will need to carry out one of the following two options:

- Add the following parameter `"-Dbamboo.agent.ignoreServerCertName=true"` to the remote agent's command line, for example:

```
java -Dbamboo.agent.ignoreServerCertName=true -jar  
atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar  
http://bamboo-host-server:8085/agentServer/
```

Please be aware that this **reduces the security of your configuration**, as the identity of your Bamboo server will not be authenticated by the remote agent.

- Use the **keytool utility to add the self-signed certificate to the trusted certificates in your keystore**. This is a more secure option, but is complex to set up. For detailed instructions of how to do this, please refer to the relevant [Sun documentation](#).

Running Bamboo without the Remote Agent Supervisor

The [remote agent supervisor](#) is included in the remote agent JAR bundled with Bamboo. The appropriate remote agent supervisor for the operating system of your remote machine, will be automatically installed when you run the default remote agent start-up command line.

 The remote agent supervisor cannot be installed on a small number of operating systems (i.e. the remote agent will start without the remote agent supervisor). If the remote agent supervisor fails to install, please check the operated systems list on the [remote agent supervisor](#) page. If your operating system is on the list and the remote agent supervisor still fails to install, please raise a [support request](#) in the Bamboo project.

If you need to run the remote agent without running the remote agent supervisor, you can execute the 'classic' version of the remote agent JAR.

The 'classic' agent jar is available from Bamboo's agent installation page for download. Follow the steps below to run the 'classic' version of the remote agent:

1. Browse to:

```
http://<host>:8085/admin/agent/addRemoteAgent.action
```

- Click the **direct agent JAR is available at bamboo-agent-2.2.2.jar** link and save 'classic' agent jar.
- Start the agent with:

```
java -jar bamboo-agent-2.2.2.jar http://<host>:8085/agentServer/
```

 The name of the jar file (e.g. *bamboo-agent-2.2.2.jar*) will vary depending on the version of Bamboo you are running.

Running the remote agent with different start-up commands

The [remote agent supervisor](#) is executed by default when you run the default remote agent start-up command line. The [remote agent supervisor](#) is implemented via a Java Service wrapper. The wrapper allows you to execute a number of general start-up commands when the remote agent is run. These commands are appended to the end of the default remote agent start-up command line:

```
java -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar  
http://bamboo-host-server:8085/agentServer <wrapper_command>
```

where <wrapper_command> is one of the keywords described below:

- console** — runs the remote agent in the foreground, i.e. display all of the commands on the screen. The [agent home directory](#) will be populated **only** if it is empty. This parameter is used by default.
- start** — runs the remote agent in the background, i.e. no commands are displayed on screen. If you have installed the remote agent as a [Windows service](#), this command will work with the service.
- stop** — stops a remote agent that is running. If you have installed the remote agent as a [Windows service](#), this command will work with the service.
- status** — (*non-Windows OS only*) returns the status of the remote agent, e.g. "Remote agent is not running."
- install** — installs the files for the remote agent, but does not start it. This will overwrite any changes that have been made to the wrapper .conf file. The [agent home directory](#) will be populated, **regardless of whether it is empty or not**, i.e. existing files will be overwritten. You may wish to use this option, if you want to customise the remote agent files before starting it.

 The name of the jar file (e.g. *atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar*) will vary depending on the version of Bamboo you are running.

(Windows only) Installing the remote agent as a Windows service

The [remote agent supervisor](#) is executed by default when you run the default remote agent start-up command line. The [remote agent supervisor](#) is implemented via a Java Service wrapper. The wrapper allows you to install or uninstall the remote agent as a service in Windows (i.e. start the Bamboo remote agent automatically when the machine boots). This is done by appending the appropriate wrapper commands to the end of the default remote agent start-up command line:

```
java -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar  
http://bamboo-host-server:8085/agentServer <wrapper_command>
```

where <wrapper_command> is one of the keywords described below:

- installntservice** — (*Windows only*) installs the remote agent as a Windows service.
- uninstallntservice** — (*Windows only*) uninstalls the remote agent as a Windows service.

 The name of the jar file (e.g. *atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar*) will vary depending on the version of Bamboo you are running.

If you have installed the NT service, you will be able to use the start and stop [start-up console commands](#) with the service.



The remote agents connect to the Bamboo server on the normal http/https port and 54663. You need to ensure that the network firewall isn't blocking these ports.

If you're having issues connecting the remote agent with the Bamboo server, please [this Troubleshooting Guide](#)

Step 3. Configure the remote agent's capabilities

Please see [Configuring capabilities](#).

Step 4. (Optional) Rename the remote agent

Your new remote agent has been automatically given a default name (e.g. 'Remote Agent on mymachine'). If you wish to rename your new remote agent, please see [Editing an agent's details](#).

Configuring remote agent capabilities using bamboo-capabilities.properties

You can define the [capabilities](#) for a specific remote agent by configuring a `bamboo-capabilities.properties` file on the agent machine. When the bamboo agent starts up, it will look in the current runtime directory (i.e. `<bamboo-agent-home>/bin`) for a file named `bamboo-capabilities.properties`. The capabilities defined in that file will then be published for the bamboo agent after registering.



We are aware of an issue that prevents a remote agent capability from being updated once it has been added using the `bamboo-capabilities.properties` file. If you choose to add capabilities with the `bamboo-capabilities.properties` file, you will only be able to update them by deleting the capability in Bamboo and restarting the remote agent. Please see [BAM-4213](#) for further details.

To configure remote agent capabilities:

1. Shut down the remote agent, if it is running.
2. Create a file named `bamboo-capabilities.properties` on the agent machine.
3. Edit the `bamboo-capabilities.properties` file and add the desired capabilities to the agent as desired. You need to follow the capability formats below in the file:

Notes:



Use `'` to escape spaces, periods and backslashes (`'`).



All capabilities, other than custom capabilities, should start with `'system'`.

- **JDK capabilities** — `system.jdk.JDK\<jdk number>=<jdk location>`

Examples:

```
system.jdk.JDK\
1.6=/System/Library/Frameworks/JavaVM.framework/Versions/1.6
system.jdk.JDK\ 1.6=C:\\Program Files\\Java\\jdk6.0.17
```

Note the double backslashes in the path for the example above.

- **Builder capabilities** — `system.builder.<builder type>.<builder label>=<builder path>`

Examples:

```
system.builder.ant.Ant=/opt/apache-ant-1.7.1
system.builder.maven.Maven\ 1=/opt/maven-1.0.2
system.builder.mvn2.Maven\ 2=/opt/maven-2.0
```

- **Perforce capabilities** — `system.p4Executable=<perforce executable location>`

Example:

```
system.p4Executable=/usr/bin/p4
```

- **Custom capabilities** — `<custom capability name>=<custom capability value>`

Example:

```
system.os=osx
```

4. Save your changes to the `bamboo-capabilities.properties` file.
5. Start up your remote agent. The capabilities defined in the `bamboo-capabilities.properties` file will be configured for your agent.

Legacy remote agent installation guide

If you have implemented your own remote agent service wrapper or have problems with the service wrapper used by the [remote agent supervisor](#) in Bamboo, you can install the legacy remote agent (pre-Bamboo 2.2) which does not have a service wrapper.



Before you begin:

- **Not sure whether to install a Remote Agent?** See [About Agents](#) to understand how Remote Agents interact with your Bamboo server.
- **Ensure that you have specified the Broker URL**, as described in the [Bamboo Setup Wizard](#) and the [Bamboo 2.0 Upgrade Guide](#).
- **Do you have sufficient Agent licenses?** See [Bamboo licensing](#) for details.
- **Have you enabled the creation of Remote Agents**, as described in [Disabling and enabling remote agents support](#).
- **Ensure that you have Java Runtime Environment 5.0 or later** installed on the agent machine.

Step 1. Download and install the Legacy Remote Agent

1. Create a directory on the agent machine (e.g. `bamboo-agent-home`), to serve as the "Bamboo agent home" for the remote agent.
2. On your Bamboo server, click **Administration** in the top menu.
3. Click **Agents** in the left navigation panel. This will display the 'Agents' screen, showing lists of all Local Agents and all Remote Agents that currently exist in your Bamboo system.
4. Click **Install Remote Agent**. The 'Install Remote Agent' screen will be displayed.
5. Click **bamboo-agent-jar** under the 'Running the agents without the service wrapper' section and save the JAR file to the directory you created in step 1.1.

 Note that if you [configure the capabilities of the remote agent](#) using a `bamboo-capabilities.properties` file, that file should be located in the same directory as the JAR file (that is, `bamboo-agent-home` in the above instructions).

Step 2. Launch the Remote Agent

Once installed, you can run the remote agent by executing the command line obtained in the previous step. This command will look something like the following:

```
java -jar bamboo-agent-2.0-SNAPSHOT.jar http://bamboo-host-server:8085/agentServer/
```

 You may wish to configure the remote agent machine to start the Bamboo remote agent automatically when the machine boots. Please consult your operating system documentation for instructions on how to do this.

You can also choose to run the remote agent with different command line parameters, to change where the remote agent stores its data or suppress the self-signed certificate of the server.

Changing where the remote agent stores its data

By default, the remote agent will store its data in a directory called `bamboo-agent-home`. If you wish to specify a different directory, add the following command line parameter:

```
-Dbamboo.home=RemoteAgentHome
```

where *RemoteAgentHome* is the path to the Bamboo agent home directory you created in step 1.1.
Your command line will look something like this:

```
java -Dbamboo.home=RemoteAgentHome -jar bamboo-agent-2.0-SNAPSHOT.jar  
http://bamboo-host-server:8085/agentServer/
```

Suppressing the self-signed certificate of the server

If your Bamboo server uses SSL (https) with a self-signed certificate, you will need to carry out one of the following two options:

- **Add the parameter `-Dbamboo.agent.ignoreServerCertName=true` to the remote agent's command line**, for example:
`java -Dbamboo.agent.ignoreServerCertName=true -jar bamboo-agent-2.0-SNAPSHOT.jar http://bamboo-host-server:8085/agentServer/`
Please be aware that this **reduces the security of your configuration**, as the identity of your Bamboo server will not be authenticated by the remote agent.
- **Use the keytool utility to add the self-signed certificate to the trusted certificates in your keystore**. This is a more secure option, but is complex to set up. For detailed instructions of how to do this, please refer to the relevant [Oracle documentation](#).

Step 3. Configure the Remote Agent's Capabilities

Please see [Configuring capabilities](#).

Step 4. (Optional) Rename the Remote Agent

Your new remote agent has been automatically given a default name (e.g. 'Remote Agent on mymachine'). If you wish to rename your new remote agent, please see [Editing an agent's details](#).

Bamboo upgrade guides

You should read the general [Bamboo generic upgrade guide](#) as well as the upgrade guide for the version of Bamboo you are upgrading to. The upgrade guides for each of the Bamboo releases, can be found below.

Please also read the relevant [release notes](#) for the version you are upgrading to.

Previous Releases

-  [Bamboo 5.0 Upgrade Guide](#)
-  [Bamboo 4.4 Upgrade Guide](#)
-  [Bamboo 4.3 Upgrade Guide](#)
-  [Bamboo 4.2 Upgrade Guide](#)
-  [Bamboo 4.1 Upgrade Guide](#)
-  [Bamboo 4.0 Upgrade Guide](#)
-  [Bamboo 3.4 Upgrade Guide](#)
-  [Bamboo 3.3 Upgrade Guide](#)
-  [Bamboo 3.2.2 Upgrade Guide](#)
-  [Bamboo 3.2 Upgrade Guide](#)
-  [Bamboo 3.1.4 Upgrade Guide](#)
-  [Bamboo 3.1.3 Upgrade Guide](#)
-  [Bamboo 3.1.1 Upgrade Guide](#)
-  [Bamboo 3.1 Upgrade Guide](#)
-  [Bamboo 3.0.3 Upgrade Guide](#)
-  [Bamboo 3.0.2 Upgrade Guide](#)
-  [Bamboo 3.0.1 Upgrade Guide](#)
-  [Bamboo 2.7.4 Upgrade Guide](#)
-  [Bamboo 3.0 Upgrade Guide](#)
-  [Bamboo 2.7.3 Upgrade Guide](#)
-  [Bamboo 2.7.1 Upgrade Guide](#)
-  [Bamboo 2.7.2 Upgrade Guide](#)
-  [Bamboo 2.6.3 Upgrade Guide](#)
-  [Bamboo 2.6.2 Upgrade Guide](#)
-  [Bamboo 2.7 Upgrade Guide](#)
-  [Bamboo 2.6.1 Upgrade Guide](#)
-  [Bamboo 2.6 Upgrade Guide](#)
-  [Bamboo 2.5.1 Upgrade Guide](#)
-  [Bamboo 2.4.3 Upgrade Guide](#)
-  [Bamboo 2.5 Upgrade Guide](#)
-  [Bamboo 2.4.2 Upgrade Guide](#)
-  [Bamboo 2.4.1 Upgrade Guide](#)
-  [Bamboo 2.4 Upgrade Guide](#)
-  [Bamboo 2.3.1 Upgrade Guide](#)
-  [Bamboo 2.3 Upgrade Guide](#)
-  [Bamboo 2.2.4 Upgrade Guide](#)
-  [Bamboo 2.2.3 Upgrade Guide](#)
-  [Bamboo 2.2.2 Upgrade Guide](#)
-  [Bamboo 2.2.1 Upgrade Guide](#)
-  [Bamboo 2.1.5 Upgrade Guide](#)
-  [Bamboo 2.1.4 Upgrade Guide](#)
-  [Bamboo 2.2 Upgrade Guide](#)

-  [Bamboo 2.1.3 Upgrade Guide](#)
-  [Bamboo 2.1.2 Upgrade Guide](#)
-  [Bamboo 2.1.1 Upgrade Guide](#)
-  [Bamboo 2.1 Upgrade Guide](#)
-  [Bamboo 2.0.6 Upgrade Guide](#)
-  [Bamboo 2.0.5 Upgrade Guide](#)
-  [Bamboo 2.0.4 Upgrade Guide](#)
-  [Bamboo 2.0.3 Upgrade Guide](#)
-  [Bamboo 2.0.2 Upgrade Guide](#)
-  [Bamboo 2.0.1 Upgrade Guide](#)
-  [Bamboo 2.0 Upgrade Guide](#)
-  [Bamboo 1.2.4 Upgrade Guide](#)
-  [Bamboo 1.2.3 Upgrade Guide](#)
-  [Bamboo 1.2.2 Upgrade Guide](#)
-  [Bamboo 1.2.1 Upgrade Guide](#)
-  [Bamboo 1.2 Upgrade Guide](#)
-  [Bamboo 1.1.2 Upgrade Guide](#)
-  [Bamboo 1.1.1 Upgrade Guide](#)
-  [Bamboo 1.1 Upgrade Guide](#)
-  [Bamboo 1.0.5 Upgrade Guide](#)
-  [Bamboo 1.0.4 Upgrade Guide](#)
-  [Bamboo 1.0.3 Upgrade Guide](#)
-  [Bamboo 1.0.2 Upgrade Guide](#)
-  [Bamboo 1.0.1 Upgrade Guide](#)
-  [Bamboo 1.0 Upgrade Guide](#)

Bamboo generic upgrade guide

This guide describes upgrade tasks that are common to upgrades for all Bamboo versions. Please ensure that you also read the [upgrade guide](#) for the version you are upgrading to.



Bamboo 5.0 has been released. Read the [Bamboo 5.0 Release Notes](#) and [Upgrade Guide](#).
Don't have Bamboo 5.0? Take a look at the features of Bamboo's latest major version and try it out!

Step 1. Export and back up Bamboo data

Export the Bamboo database

In Bamboo, export your Bamboo database, for backup purposes. See [Exporting data for backup](#) for instructions. Note that this may take a long time to complete depending on the number of builds and tests in your system.

If you are using an [external database](#), then use the native database backup tools to acquire a database dump.

On this page:

- [Step 1. Export and back up Bamboo data](#)
- [Step 2. Install the new Bamboo version](#)
- [Step 3. Configure the new Bamboo](#)
- [Step 4. Launch Bamboo](#)
- [Troubleshooting](#)

Related pages:

- [Bamboo installation guide](#)
- [Bamboo release notes](#)
- [Bamboo security advisories](#)

Back up the Bamboo configuration

Shut down Bamboo.

Back up your 'Bamboo Home' directory, which includes the builds and configuration directories.

▼ [Expand for info on how to find these directories...](#)

Click **Administration**, and then **System** Information (under 'System') in your Bamboo instance, and note the location of the 'Bamboo Home', 'Build Path' and 'Configuration Path' directories (under 'Bamboo Paths'):

Bamboo Paths	
Current running directory:	/opt/j2ee/domains/bamboo.atlassian.com/jira/apache-tomcat-5.5.20
Configuration Path:	/home/j2ee/bamboo/xml-data/configuration
Build Path:	/home/j2ee/bamboo/xml-data/builds
Build Working Directory:	/home/j2ee/bamboo/xml-data/build-dir
Bamboo Home:	/home/j2ee/bamboo

For more information about these directories see [Important Directories and Files](#).

Step 2. Install the new Bamboo version



- Make sure that your {BAMBOO_INSTALL} directory is either a new directory, or else delete your old {BAMBOO_INSTALL} directory before you begin, as legacy files may cause problems.
- The {BAMBOO_HOME} directory must be different from the {BAMBOO_INSTALL} directory. This will ensure that your data is not lost when upgrading or re-installing Bamboo.

If you are installing using the .zip or .tgz packages:

- Specify the {BAMBOO_HOME} directory in <Bamboo-install>/webapp/WEB-INF/classes/bamboo-init.properties, use the same 'Bamboo Home' directory as in your old installation. That is, specify the same directory and path as the 'Bamboo Home' directory shown in Step 1 (above).

If you are installing using the Windows installer or the Mac installer:

- Ensure that you then point the new installation to the old 'Bamboo Home', changing the path at file <Bamboo-install>/webapp/WEB-INF/classes/bamboo-init.properties. Note that in a EAR-WAR installation, the webapp directory may have a different name.

Follow Steps 1 and 2 of the installation instructions for your operating system:

- [Bamboo Installation Guide — Windows](#)
- [Bamboo Installation Guide — Linux](#)
- [Bamboo Installation Guide — Mac](#)
- [Bamboo EAR-WAR installation guide](#)

Step 3. Configure the new Bamboo

Reconnect external user directories

You only need to perform this step if either of the following apply:

- **LDAP integration** — If you had previously integrated Bamboo with LDAP/AD, copy your old ../<Bamboo-install>/webapp/WEB-INF/classes/atlassian-user.xml to its *new location*. Starting with version 3.2, the atlassian-user.xml file will be stored at {BAMBOO-HOME}/xml-data/configuration/. Please replace the existing file using your old atlassian-user.xml. Note that in a EAR-WAR installation, the webapp directory may have a different name.
- **Crowd integration** — If you had previously integrated Bamboo with Crowd, you will need to re-enable Crowd integration. For details please see [Integrating Crowd with Bamboo](#).

Update any installed plugins

If you are using any plugins other than the ones that ship with Bamboo, check that each one is compatible with the new version of Bamboo. Upgrade any plugins that are out-of-date, and disable any plugins that are incompatible with your new version of Bamboo.

Automatic upgrade of remote agents

For Bamboo 3.2 and later, remote agents are upgraded automatically. The remote agent can automatically detect when a new version is available, and has a special classloader that downloads the new classes from the server.

See also the [Bamboo remote agent installation guide](#).

Configure Context path

If you had a context path configured for your Bamboo instance ([http://hostname:\[port\]/context_path](http://hostname:[port]/context_path)), please follow the steps from here: [Changing the Root Context Path](#) and make sure to update the {BAMBOO-INSTALL}/conf/wrapper.conf file and Bamboo Base URL (under

Administration -> General Configuration page).

Step 4. Launch Bamboo

Start Bamboo

Before starting Bamboo, ensure that Bamboo has *write access* to your database. This is required to complete the upgrade tasks that will run when you start up Bamboo. Please consult your database documentation to ensure that you have configured your database appropriately.

Once you have installed Bamboo and set the `bamboo.home` property (as described in the [installation guides](#)), start Bamboo. The upgrade process will be performed when Bamboo starts up. You will not see the Setup Wizard.

Monitor the `atlassian-bamboo.log` to ensure that the upgrade process completes successfully.

Reindex Bamboo (if indicated in the release notes)

Bamboo maintains an index of its build results. This allows Bamboo to display aggregate build results information across builds. You may need to perform a re-index of Bamboo if the upgrade process requires it. This step may or may not be required (depending on the upgrade versions). Also note that you only need to do this if you have existing data in Bamboo.

To re-index, go to **Administration**, then **Indexing** (under 'System').



Depending on the number of builds and tests you may have, the indexing process may take a significant amount of time. During this period, Bamboo will not be available. Also, it is advisable to ensure that all you have disabled all build queues (or all agents, if you are upgrading from Bamboo 2.0 or later), and that no builds are in progress when you start the re-indexing process. If you have a large instance, it is recommended that you reindex overnight.

Troubleshooting

If you have any problems during upgrade, please raise a support request at <https://support.atlassian.com/> and attach your `atlassian-bamboo.log` so we can help you find out what's gone wrong.

Bamboo FAQ

Bamboo FAQ

Answers to commonly raised questions about configuring and using Bamboo:

- What Is Continuous Integration?
- Installation FAQ
 - Changing the Root Context Path
 - Configuring Bamboo on start-up
 - Getting Bamboo to use the jetty.xml file
 - Installation notes for Bamboo on JBoss 4.x
 - Running Bamboo as a Service on Windows
 - Running Bamboo over HTTPS
 - Running Bamboo service on Windows as the local user
 - Setting up JNDI mail on JBoss 4.2.2
 - Setting up JNDI on Jetty
- Support Policies
 - Bamboo Support Policy
 - Bug Fixing Policy
 - Deploying Multiple Atlassian Applications in a Single Tomcat Container
 - How to Report a Security Issue
 - New Features Policy
 - Patch Policy
 - Security Advisory Publishing Policy
 - Security Patch Policy
 - Severity Levels for Security Issues
- Usage FAQ
 - Backing up Bamboo instances over 4GB
 - Bamboo Database Schema
 - Binding Bamboo to one IP address
 - Can Bamboo build and test non-Java projects
 - Can multiple plans share a common 3rd-party directory
 - Changing Bamboo database settings
 - Finding the Support Entitlement Number (SEN)
 - How do I manually set the version of new Subversion workspaces
 - Securing your repository connection
 - Changing the remote agent heartbeat interval
 - Cloning a Bamboo instance
 - CVS Error logging in Bamboo
 - Do I have to upgrade all remote agents for Bamboo Release 2.1.2
 - Enable User Management debug logging in Bamboo
 - Hibernate errors in logs after upgrading to Bamboo 2.0
 - How do I construct a cron expression in Bamboo
 - How do I disable SSH access to my elastic instances
 - How do I shut down my elastic instances if I have restarted my Bamboo server
 - How do I stop Bamboo from shutting itself down and restarting
 - How do I stop the Bamboo server from automatically configuring my remote agent's capabilities
 - JUnit parsing in Bamboo
 - Known issues with CVS in Bamboo
 - Monitoring and Profiling Bamboo
 - Monitor Memory usage and Garbage Collection in Bamboo
 - Moving Bamboo-Home of an agent
 - Performing a thread dump.
 - Removing Coverage plug-in data from the Bamboo database
 - Restoring passwords to recover admin users
 - Send Errors to stderr - Script Builder in Visual Studio WinXP to build Solutions Files
 - Using Bamboo with Clover
 - Getting gcov results in Clover coverage summary
 - Working with Sun JAVA libraries
 - Bamboo indicates that my Ant or Maven builds failed, even though they were successful
 - How can I pass bamboo variables to my build script
- Contributing to the Bamboo documentation
 - Tips of the Trade
 - Bamboo Documentation in Other Languages

Bamboo Evaluator's FAQ

If you are evaluating Bamboo, you may also wish to consult the [Bamboo Evaluator's FAQ](#):

- [Can Bamboo be Extended or Integrated with Other Tools?](#)
- [Can Bamboo be Used for Release Management?](#)
- [Can I use Clover Code Coverage with Bamboo?](#)
- [How is Bamboo Licensed?](#)
- [What are Remote and Elastic Agents?](#)
- [What are the Hardware Requirements for Bamboo?](#)
- [What Build Tools can Bamboo Work With?](#)
- [What Environments are Supported?](#)
- [What is Continuous Integration?](#)
- [Who Broke The Build?](#)

Need more help?

Do you have a question, or need help with Bamboo? Please create a [support request](#).

Browse our [Bamboo Developer FAQ](#).

You may also like to check out the forums:

- [Bamboo General Forum](#)
- [Bamboo Developers Forum](#)

On this page:

- [Bamboo FAQ](#)
- [Bamboo Evaluator's FAQ](#)
- [Need more help?](#)

Installation FAQ

- [Changing the Root Context Path](#)
- [Configuring Bamboo on start-up](#)
- [Getting Bamboo to use the jetty.xml file](#)
- [Installation notes for Bamboo on JBoss 4.x](#)
- [Running Bamboo as a Service on Windows](#)
- [Running Bamboo over HTTPS](#)
- [Running Bamboo service on Windows as the local user](#)
- [Setting up JNDI mail on JBoss 4.2.2](#)
- [Setting up JNDI on Jetty](#)

Changing the Root Context Path

When running Bamboo behind a proxy, you might need to change the Root Context Path i.e. the host URL referenced while accessing Bamboo (e.g. <http://localhost:8085/bamboo>). You might also need to change the Base URL to the Apache proxy URL, you can change this value from "General Configuration" page under Bamboo Administration.

To change the context path from '/' to '/Your_Context_Path':

For Bamboo 3.2 or below

- **If you are using the `bamboo.sh` script to start Bamboo:**

Change the following line in your `bamboo.sh` script:

```
RUN_CMD="java -server -Xms256m -Xmx512m -XX:MaxPermSize=256m
-Djava.awt.headless=true -classpath $CLASSPATH
-Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085
com.atlassian.bamboo.server.Server 8085 ./webapp /"
```

to:

```
RUN_CMD="java -server -Xms256m -Xmx512m -XX:MaxPermSize=256m
-Djava.awt.headless=true -classpath $CLASSPATH
-Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085
com.atlassian.bamboo.server.Server 8085 ./webapp /Your_Context_Path"
```

- **Or, if you are using the wrapper to start Bamboo:**

The wrapper reads the configuration information from the `wrapper.conf` file in the `../<Bamboo-Install>/conf/` folder. Find the following line:

```
wrapper.app.parameter.4=
```

and replace it with the following line:

```
wrapper.app.parameter.4=/Your_Context_Path
```

- **Or, if you are using the `jetty.xml` file to configure Bamboo**

See our documentation on [configuring jetty.xml file](#)

For Bamboo 3.3 or above

- **If you are using the `bamboo.sh` script or wrapper to start Bamboo:**

The wrapper reads the configuration information from the `wrapper.conf` file in the `../<Bamboo-Install>/conf/` folder. Find the following line:

```
wrapper.app.parameter.4=
```

and replace it with the following line:

```
wrapper.app.parameter.4=/Your_Context_Path
```

- **Or, if you are using the `jetty.xml` file to configure Bamboo**

See our documentation on [configuring jetty.xml file](#)

Configuring Bamboo on start-up



Restarting Bamboo

After editing the options below, Bamboo needs to be shut down and restarted for the changes to take effect. If you have any [elastic agents](#) running, ensure that they are [shut down](#) before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become [orphaned from your Bamboo server](#).

- [Configuring Bamboo Server on start-up](#)
 - [Configuring Bamboo's start-up parameters under Linux](#)
 - [Bamboo 3.1 and above: Modifying the wrapper.](#)
 - [Editing wrapper.conf file to increase memory:](#)
 - [Bamboo 3.0 and below: Modifying the `bamboo.sh` script.](#)
 - [Configuring Bamboo's start-up parameters under Windows](#)
 - [Configuring Bamboo runtime parameters for `bamboo.war`](#)
- [Configuring Bamboo agent on start-up](#)
 - [Configuring Bamboo agent service](#)
 - [Configuring classic Bamboo agent](#)
 -  In order to manually configure the `/tmp` directory (`/tmp` directory) for Bamboo server (or remote agents), run the Bamboo server (or remote agents) with the following parameter:

Configuring Bamboo Server on start-up

Configuring Bamboo's start-up parameters under Linux

Bamboo on Linux/Unix can be started by either executing the `bamboo.sh` script or using the wrapper. Either way, the Bamboo server can be customised at start-up.

Bamboo 3.1 and above: Modifying the wrapper.

The wrapper reads the configuration from wrapper.conf found in ../<BAMBOO_INSTALL>/conf. (The properties are documented inside the file.)

Editing wrapper.conf file to increase memory:

On the Server edit the following lines to set the init & max memory:

```
# Initial Java Heap Size (in MB)
wrapper.java.initmemory=256
# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=512
```

On the Agent edit the following line to set the Xmx & Xms settings:

```
wrapper.java.additional.3=-Xms256m
wrapper.java.additional.4=-Xmx512m
```

Bamboo 3.0 and below: Modifying the bamboo.sh script.

The bamboo.sh script takes four parameters: start|stop|restart|status.

To customise these parameters at startup , edit the \$RUN_CMD variable:

```
#
#This is how the Bamboo server will be started
#
RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH
-Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085
com.atlassian.bamboo.server.Server 8085 ./webapp /"
```

- **java -Xms256m -Xmx512m** specifies the minimum and maximum Java Heap size.
- **-classpath \$CLASSPATH** sets the class path at startup.
- **-Djetty.port=8085** specifies the port number for the Jetty server.
- **com.atlassian.bamboo.server.Server 8085 ./webapp /** is the main class that will be executed followed by the context path.

In some cases it might be useful to increase the PermGen space. To do this, add the following parameter to the RUN_CMD variable: "XX:MaxPermSize=512m". This will set the PermGen space to 512mb next time Bamboo is run. For example, this could be a valid RUN_CMD configuration: RUN_CMD="java -Xms256m -Xmx512m -XX:MaxPermSize=512m -Djava.awt.headless=true -classpath \$CLASSPATH -Dorg.eclipse.jetty.xml.XmlParser.Validating=false com.atlassian.bamboo.server.Server webapp/WEB-INF/classes/jetty.xml"

Configuring Bamboo's start-up parameters under Windows

Bamboo can be started in Windows with the startup.bat file (from the command line) or as a Windows Service. Both use the wrapper to start Bamboo. As in Linux (see above), the wrapper reads the configuration from wrapper.conf. Please edit the ../conf/wrapper.conf file (situated in the root of your Bamboo_Installation directory) as required.

For example to add more java parameters to the bamboo process extend the parameter list:

```
wrapper.java.additional.1=-Dorg.mortbay.xml.XmlParser.NotValidating=true
wrapper.java.additional.2=-XX:MaxPermSize=256m
wrapper.java.additional.3=-Djava.awt.headless=true
wrapper.java.additional.4=-D<your-parameter>
```

Configuring Bamboo runtime parameters for bamboo.war

The application container that deploys bamboo has to be configured with the additional java parameter.

Example Tomcat:

`./bin.setenv.sh`

```
...
JAVA_OPTS="-server -XX:MaxPermSize=256m -Dbamboo.home=/path/to/bamboo-tomcat-home
-Xmx512m -Djava.awt.headless=true -D<your-parameter>=<value> $JAVA_OPTS"
export JAVA_OPTS
...
```

Configuring Bamboo agent on start-up

Configuring Bamboo agent service

Bamboo agent uses the same wrapper as the server. If you are running Bamboo as a service, then edit the `conf/wrapper.conf` file in `<Bamboo-Agent-Home>`

For example, to add more java parameters to the Bamboo process, extend the parameter list:

```
wrapper.java.additional.1=-Dorg.mortbay.xml.XmlParser.NotValidating=true
wrapper.java.additional.2=-XX:MaxPermSize=256m
wrapper.java.additional.3=-Djava.awt.headless=true
wrapper.java.additional.4=-D<your-parameter>
```

Configuring classic Bamboo agent

To apply additional properties to the classic Bamboo agent, append the system to the start-up command.

In the example below, we are specifying a Bamboo Home for the agent by adding `-Dbamboo.home` system property during startup:

```
// Without a Bamboo Home specified
java -jar bamboo-agent-2.6.2.jar http://192.168.35.128:8085/bamboo/agentServer/
// With a Bamboo Home specified
java -jar -Dbamboo.home=/bamboo/home bamboo-agent-2.6.2.jar
http://192.168.35.128:8085/bamboo/agentServer/
```

i *In order to manually configure the `/tmp` directory (`/tmp` directory) for Bamboo server (or remote agents), run the Bamboo server (or remote agents) with the following parameter:*

```
-Djava.io.tmpdir=_path_to_temp_directory_
```

Getting Bamboo to use the `jetty.xml` file

Overview

Bamboo can be configured to use advanced functionality such as JNDI resources and HTTPS. This is achieved through use of the `jetty.xml` file, however Bamboo doesn't use this for configuration by default. To use `jetty.xml` for advanced configuration, you will need to tell Bamboo to use it.

The method for doing this uses the Java Service Wrapper. Older versions of Bamboo used the `bamboo.sh` file.

The instructions on this page apply to the Bamboo distribution, not the Bamboo EAR-WAR distribution.

Step 1 - Instruct Bamboo to use `jetty.xml`

To use the Java Service Wrapper to start Bamboo:

When starting up Bamboo with the Java Service Wrapper, modify the `wrapper.conf` file in the `conf` directory:

1. Replace the argument which specifies your port number "`wrapper.app.parameter.2=8085`" with "`wrapper.app.parameter.2=../webapp/WEB-INF/classes/jetty.xml`".

2. Comment out the other arguments: "wrapper.app.parameter.3=../webapp" and "wrapper.app.parameter.4=/"

This will make Bamboo start up using your `jetty.xml` configuration file instead of the default three arguments (port, web app directory, context path).

If you are using the bamboo.sh script to start Bamboo:

The standard Bamboo startup script can be customised to use the `jetty.xml` file by modifying the following section in your `bamboo.sh` script (this section specifies how the Bamboo server will start):

Bamboo 3.1 and newer

Configuration is performed using the Java Service Wrapper as described above.

Bamboo 3.0

```
RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH
-Dorg.eclipse.jetty.xml.XmlParser.Validating=false -Djetty.port=8085
com.atlassian.bamboo.server.Server 8085 ../webapp /"
```

Now, modify this startup script to read the `jetty.xml` file from `webapp/WEB-INF/classes/jetty.xml` by changing the `RUN_CMD` argument as follows:

```
RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH
-Dorg.eclipse.jetty.xml.XmlParser.Validating=false
com.atlassian.bamboo.server.Server webapp/WEB-INF/classes/jetty.xml"
```

Bamboo earlier than 3.0:

```
RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH
-Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085
com.atlassian.bamboo.server.Server 8085 ../webapp /"
```

Now, modify this startup script to read the `jetty.xml` file from `webapp/WEB-INF/classes/jetty.xml` by changing the `RUN_CMD` argument as follows:

```
RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH
-Dorg.mortbay.xml.XmlParser.NotValidating=true com.atlassian.bamboo.server.Server
webapp/WEB-INF/classes/jetty.xml"
```

Step 2 - Set root context web application in jetty.xml

Linux Platforms

Replace your existing `<Bamboo-install>/webapp/WEB-INF/classes/jetty.xml` file, with [this jetty.xml](#) file.

Windows Platforms

Replace your existing `<Bamboo-install>\webapp\WEB-INF\classes\jetty.xml` file, with [this jetty.xml](#) file.

Installation notes for Bamboo on JBoss 4.x

This page is for people who are deploying the Bamboo EAR/WAR edition on the JBoss 4.x application server. For full installation instructions please see the [Bamboo EAR-WAR installation guide](#).



Bamboo 2.4.x does not run on JBoss 4.2.3 or later

We are aware of a JBoss issue that currently prevents Bamboo 2.4.x from running on JBoss 4.2.3 or later. If you are using JBoss 4.2.3 or later, we recommend that you **do not upgrade** your Bamboo installation until a fix has been implemented. Please see [BA M-4705](#) for more information.

File extraction notes

To deploy Bamboo EAR-WAR onto your JBoss application server, copy the Bamboo WAR file to `../<JBoss-install>/server/default/deploy/atlassian-bamboo-1.1.2.war`.

By default the WAR file will extract to `atlassian-bamboo-<version>`. The name of the directory in the webapps folder will form the URL required to access Bamboo, e.g. `<JBoss-install>/server/default/deploy/atlassian-bamboo-1.1.2.war` will become <http://host:port/atlassian-bamboo-1.1.2/>

How to set Java OPTs on JBoss 4.x

- **Windows:**

1. Find the run.bat file.
2. Edit JAVA_OPTS to set the desired properties variable:
FROM

```
if exist "%JBOSS_HOME%\bin\native" set JAVA_OPTS=%JAVA_OPTS%  
-Djava.library.path=%JBOSS_HOME%\bin\native
```

TO

```
if exist "%JBOSS_HOME%\bin\native" set JAVA_OPTS=%JAVA_OPTS%  
-Djava.library.path=%JBOSS_HOME%\bin\native -server -XX:MaxPermSize=256m  
-Dbamboo.home=/opt/bamboo/bamboohome -Xmx512m -Djava.awt.headless=true  
-Dbamboo.home=Your_Path_To_Bamboo
```

- **Linux-based systems:**

1. Find the run.sh file
2. Edit JAVA_OPTS to set the desired properties variable:
FROM

```
# Setup JBoss sepecific properties  
JAVA_OPTS="-Dprogram.name=$PROGNAME $JAVA_OPTS"
```

TO

```
# Setup JBoss sepecific properties  
JAVA_OPTS="-Dprogram.name=$PROGNAME -server -XX:MaxPermSize=256m  
-Dbamboo.home=/opt/bamboo/bamboohome -Xmx512m -Djava.awt.headless=true  
-Dbamboo.home=Your_Path_To_Bamboo $JAVA_OPTS"
```



For further reference

Please visit the [JBoss Wiki](#) page on setting JavaOpts

Running Bamboo as a Service on Windows

Once you have installed Bamboo, you can choose to run Bamboo as service so that it starts up every time windows restarts.

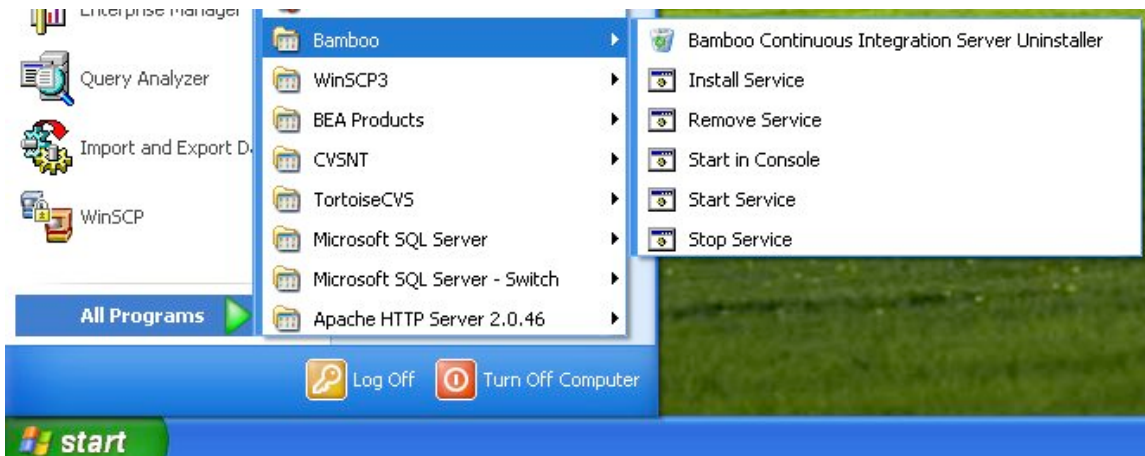
To do this,

1. Click on the **Start menu** in Windows,
2. Select **Bamboo** from the programs list,
3. Click on **Install Service** option to install Bamboo as a service in Windows.
4. Click **Start Service** to start the service.



Running Bamboo as local user

Bamboo service installs itself as the windows NT SYSTEM User, to run Bamboo as a local user see [Running Bamboo service on](#)



Upgrading Bamboo server

If you have just upgraded your Bamboo server, **you must** re-install the Bamboo service. You can do this by removing the service and installing it again.

Running Bamboo over HTTPS

This document is a guide to configuring the Bamboo Standalone distribution (not EAR-WAR) with basic HTTPS authentication. For further reference please visit the Jetty page on [configuring SSL with Jetty](#).

1. Adding Certificate to your Keystore

Option 1. Using a self-signed Certificate

The simplest way to generate keys and certificates is to use the keytool application that comes with the JDK, as it generates keys and certificates directly into the keystore.

The following command will generate a key pair and certificate directly into a keystore:

```
keytool -keystore keystore -alias jetty -genkey -keyalg RSA
```

This command will prompt for information about the certificate and for passwords to protect both the keystore and the keys within it. The only mandatory response is to provide the fully qualified host name of the server at the "first and last name" prompt.

Option 2. Using Certificate issued by an Certificate Authority

Certificate Option 2 – Use a Certificate Issued by a Certificate Authority

When running Bamboo in a production environment, you will need a certificate issued by a certificate authority (CA, sometimes also called a 'certification authority') such as [VeriSign](#), [Thawte](#) or [TrustCenter](#). The instructions below are adapted from the [Tomcat documentation](#).

First you will generate a local certificate and create a 'certificate signing request' (CSR) based on that certificate. You will submit the CSR to your chosen certificate authority. The CA will use that CSR to generate a certificate for you.

1. Use Java's keytool utility to generate a local certificate, as described in the previous section.
2. Use the keytool utility to generate a CSR, replacing the text <MY_KEYSTORE_FILENAME> with the path to and file name of the .keystorefile generated for your local certificate:

```
keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <MY_KEYSTORE_FILENAME>
```

3. Submit the generated file called certreq.csr to your chosen certificate authority. Refer to the documentation on the CA's website to find out how to do this.
4. The CA will send you a certificate.
5. Import the new certificate into your local keystore:

```
keytool -importcert -alias tomcat -keystore <MY_KEYSTORE_FILENAME> -file <MY_CERTIFICATE_FILENAME>
```

Now, we need to configure an SSL listener.

NOTE: If you're running a **Bamboo EAR-WAR** distribution deployed in Tomcat, please follow [these instructions](#). Otherwise, go to Step 2 below.

2. Configuring Jetty

If you're running Bamboo on Windows, your jetty.xml should look like this:

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN"
"http://www.eclipse.org/jetty/configure.dtd">

<!-- ===== -->
<!-- Configure the Jetty Server -->
<!-- -->
<!-- Documentation of this file format can be found at: -->
<!-- http://wiki.eclipse.org/Jetty/Reference/jetty.xml_syntax -->

<Configure id="Server" class="org.eclipse.jetty.server.Server">

    <!-- ===== -->
    <!-- Server Thread Pool -->
    <!-- ===== -->
    <!-- Set name="ThreadPool" -->
    <!-- Default queued blocking threadpool -->
    <!--New class="org.eclipse.jetty.util.thread.QueuedThreadPool">
        <Set name="minThreads">10</Set>
        <Set name="maxThreads">200</Set>
    </New>
    </Set-->

    <!-- ===== -->
    <!-- Set connectors -->
    <!-- ===== -->
    <!-- Add and configure a Connector to port 8085 -->
    <!-- The default port can be changed using: java -Djetty.port=8085 -->
    <!-- - - - - - -->

    <Call name="addConnector">
        <Arg>
            <New class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">
                <Set name="Port">8443</Set>
                <Set name="Keystore"><SystemProperty name="jetty.home"
default=""/>..\keystore</Set>

                <Set name="Password">password</Set>
                <Set name="KeyPassword">password</Set>
            </New>
        </Arg>
    </Call>

    <!-- - - - - - -->
    <!-- For SSL Connexctions use: -->
    <!-- - - - - - -->

    <!--<Call name="addConnector">-->
    <!--<Arg>-->
    <!--<New class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">-->

    <!-- - - - - - -->
    <!-- Add web applications Context. -->
    <!-- The default location can be changed using: java -Dbamboo.webapp= -->
    <!-- - - - - - -->

    <Call name="setHandler">
```

```

    <Arg>
      <New class="org.eclipse.jetty.webapp.WebAppContext">
        <Arg name="webApp">
          <!--SystemProperty name="bamboo.webapp"
default="/opt/dev/src/atlassian/bamboo-trunk/components/bamboo-web-app/src/main/web
app"/-->
          <SystemProperty name="bamboo.webapp" default="../webapp"/>
        </Arg>
        <Arg name="contextPath"></Arg>
        <!--<Set name="parentLoaderPriority">true</Set>-->
        <Set name="defaultsDescriptor">webdefault.xml</Set>
        <Get name="sessionHandler">
          <Set name="sessionManager">
            <New
class="org.eclipse.jetty.server.session.HashSessionManager">
              <Set name="httpOnly">true</Set>              <!-- use M$ http
only cookies? -->
            </New>
          </Set>
        </Get>
      </New>
    </Arg>
  </Call>
  <!-- - - - - - -->
  <!-- Add Any JNDI Resources -->
  <!-- The default location can be changed using: java -Dbamboo.webapp= -->
  <!-- - - - - - -->

  <!--New id="resourceID" class="org.eclipse.jetty.plus.jndi.Resource">
  <Arg>resourceName</Arg>
  <Arg>
    <New class="org.eclipse.jetty.jndi.factories.MailSessionReference">
      <Set name="user">name</Set>
      <Set name="password">password</Set>
      <Set name="properties">
        <New class="java.util.Properties">
          <Put name="mail.smtp.host">host</Put>
          <Put name="mail.from">fromaddress@example.com</Put>
        </New>
      </Set>
    </New>
  </Arg>
</New-->

  <!-- - - - - - -->
  <!-- Uncomment to enable AJP Support -->

  <!--
  <Call name="addConnector">
    <Arg>
      <New class="org.eclipse.jetty.ajp.Ajp13SocketConnector">
        <Set name="port">8009</Set>
      </New>
    </Arg>
  </Call>
  -->

```

```
</Configure>
```

1. Please note that Password and KeyPassword indicate the passwords you entered when you imported the certificate to the keystore.
2. The keystore file in this example is given relative to the Bamboo Application Directory, so copy your keystore file to BAMBOO_INSTALL directory.

If you're running Bamboo on Linux or Mac, your jetty.xml should look like this:

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN"
"http://www.eclipse.org/jetty/configure.dtd">

<!-- ===== -->
<!-- Configure the Jetty Server -->
<!--
<!-- Documentation of this file format can be found at:
<!-- http://wiki.eclipse.org/Jetty/Reference/jetty.xml_syntax -->

<Configure id="Server" class="org.eclipse.jetty.server.Server">

    <!-- ===== -->
    <!-- Server Thread Pool -->
    <!-- ===== -->
    <!-- Set name="ThreadPool" -->
    <!-- Default queued blocking threadpool -->
    <!-- New class="org.eclipse.jetty.util.thread.QueuedThreadPool">
        <Set name="minThreads">10</Set>
        <Set name="maxThreads">200</Set>
    </New>
</Set-->

    <!-- ===== -->
    <!-- Set connectors -->
    <!-- ===== -->
    <!-- Add and configure a Connector to port 8085 -->
    <!-- The default port can be changed using: java -Djetty.port=8085 -->
    <!-- - - - - - -->

    <Call name="addConnector">
        <Arg>
            <New class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">
                <Set name="Port">8443</Set>
                <Set name="Keystore"><SystemProperty name="jetty.home"
default=""/>keystore</Set>

                <Set name="Password">password</Set>
                <Set name="KeyPassword">password</Set>
            </New>
        </Arg>
    </Call>

    <!-- - - - - - -->
    <!-- For SSL Connexctions use: -->
    <!-- - - - - - -->

    <!--<Call name="addConnector">-->
    <!--<Arg>-->
    <!--<New class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">-->
```

```

<!-- - - - - - -->
<!-- Add web applications Context. -->
<!-- The default location can be changed using: java -Dbamboo.webapp= -->
<!-- - - - - - -->

<Call name="setHandler">
  <Arg>
    <New class="org.eclipse.jetty.webapp.WebAppContext">
      <Arg name="webApp">
        <!--SystemProperty name="bamboo.webapp"
default="/opt/dev/src/atlassian/bamboo-trunk/components/bamboo-web-app/src/main/web
app"/-->
        <SystemProperty name="bamboo.webapp" default="./webapp"/>
      </Arg>
      <Arg name="contextPath"/></Arg>
      <!--<Set name="parentLoaderPriority">true</Set>-->
      <Set name="defaultsDescriptor">webdefault.xml</Set>
      <Get name="sessionHandler">
        <Set name="sessionManager">
          <New
class="org.eclipse.jetty.server.session.HashSessionManager">
            <Set name="httpOnly">true</Set>          <!-- use M$ http
only cookies? -->
          </New>
        </Set>
      </Get>
    </New>
  </Arg>
</Call>
<!-- - - - - - -->
<!-- Add Any JNDI Resources -->
<!-- The default location can be changed using: java -Dbamboo.webapp= -->
<!-- - - - - - -->

<!--New id="resourceID" class="org.eclipse.jetty.plus.jndi.Resource">
<Arg>resourceName</Arg>
<Arg>
  <New class="org.eclipse.jetty.jndi.factories.MailSessionReference">
    <Set name="user">name</Set>
    <Set name="password">password</Set>
    <Set name="properties">
      <New class="java.util.Properties">
        <Put name="mail.smtp.host">host</Put>
        <Put name="mail.from">fromaddress@example.com</Put>
      </New>
    </Set>
  </New>
</Arg>
</New-->

<!-- - - - - - -->
<!-- Uncomment to enable AJP Support -->

<!--
<Call name="addConnector">
  <Arg>
    <New class="org.eclipse.jetty.ajp.Ajp13SocketConnector">
      <Set name="port">8009</Set>
    </New>
  </Arg>
</Call>
-->

```

```
</Configure>
```

1. Please note that Password and KeyPassword indicate the passwords you entered when you imported the certificate to the keystore.
2. The keystore file in this example is given relative to the Bamboo Application Directory, so copy your keystore file to BAMBOO_INSTALL directory.

3. Modify wrapper.conf

Please modify <Bamboo Install Directory>\conf\wrapper.conf by:

- Replacing the argument which specifies the port number "wrapper.app.parameter.2=8085" with "wrapper.app.parameter.2=../webapp/WEB-INF/classes/jetty.xml".
- Commenting out the other arguments: "wrapper.app.parameter.3=../webapp" and "wrapper.app.parameter.4=/".

After following the steps mentioned above, you should be able to start your Bamboo instance by reaching it at <https://127.0.0.1:8443>

Running Bamboo EAR-WAR over HTTPS

After adding a certificate to your Keystore (Step 1), please follow these steps:

1. Open Tomcat's server.xml, located at <Tomcat_Installation_Directory>/conf
2. Uncomment this section:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS" />
```

3. Edit it so it looks like this:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystorePass="<keystore_password_defined>"
    keystoreFile="<keystore_file_location>" />
```

4. Start Tomcat
5. Access Bamboo at https://host_name:8443/bamboo_context

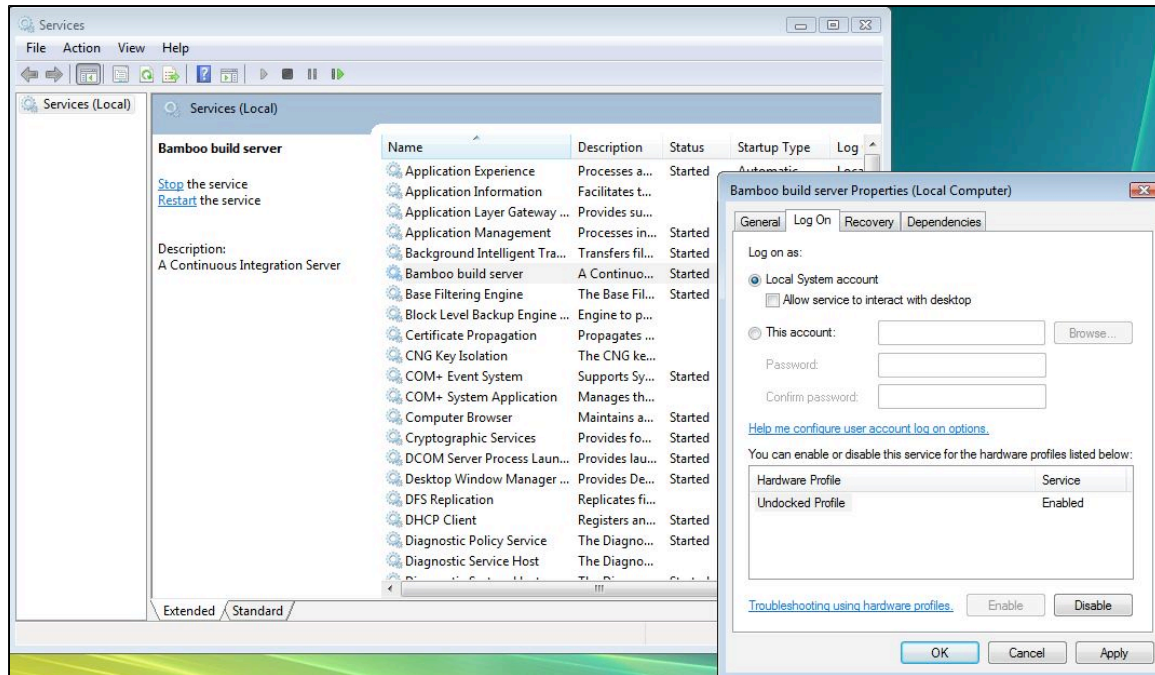
Running Bamboo service on Windows as the local user

1. Install Bamboo Application Server

1. [Download Bamboo](#) and run the [Setup Wizard](#).
2. Install Bamboo as Windows service, as described in the [Bamboo Installation Guide \(Windows\)](#).

2. Edit the Bamboo service to run as the "local user"

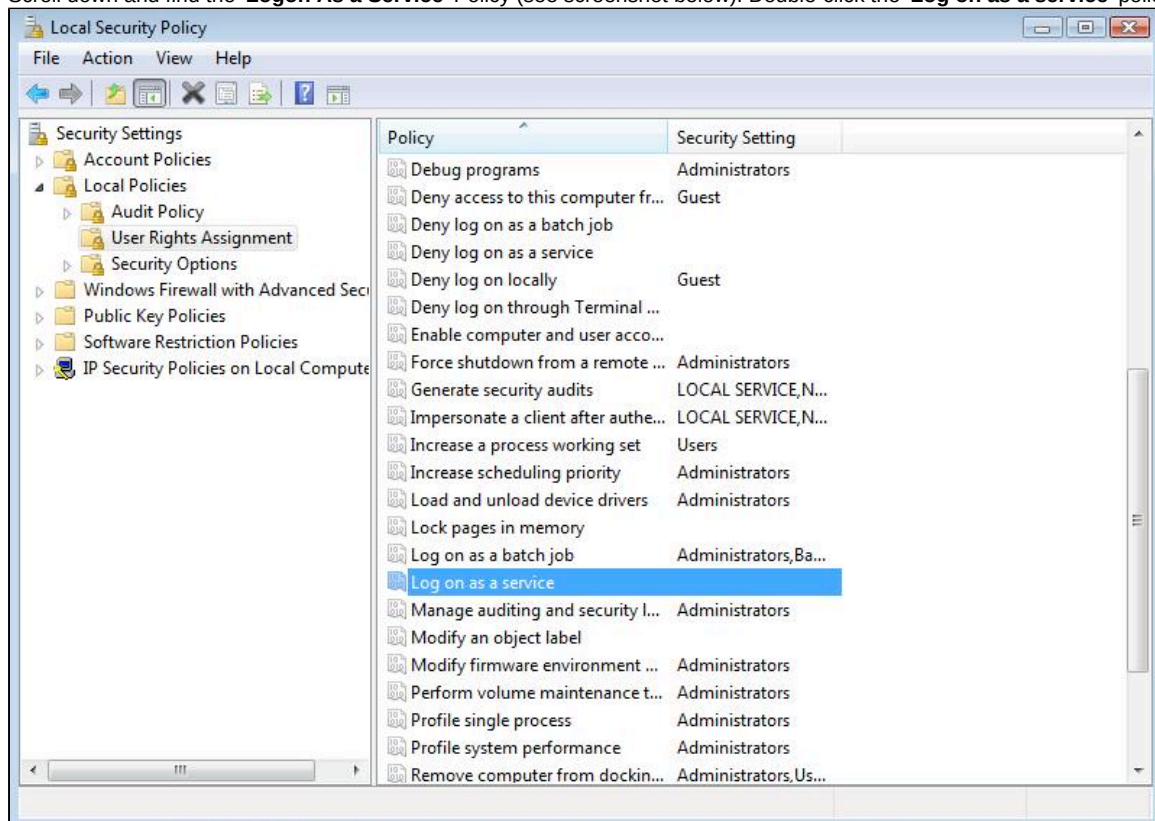
1. Go to **Start -> Run** and enter 'services.msc'.
2. The 'Services' window will display (see screenshot below). Double-click the 'Bamboo build server' row.



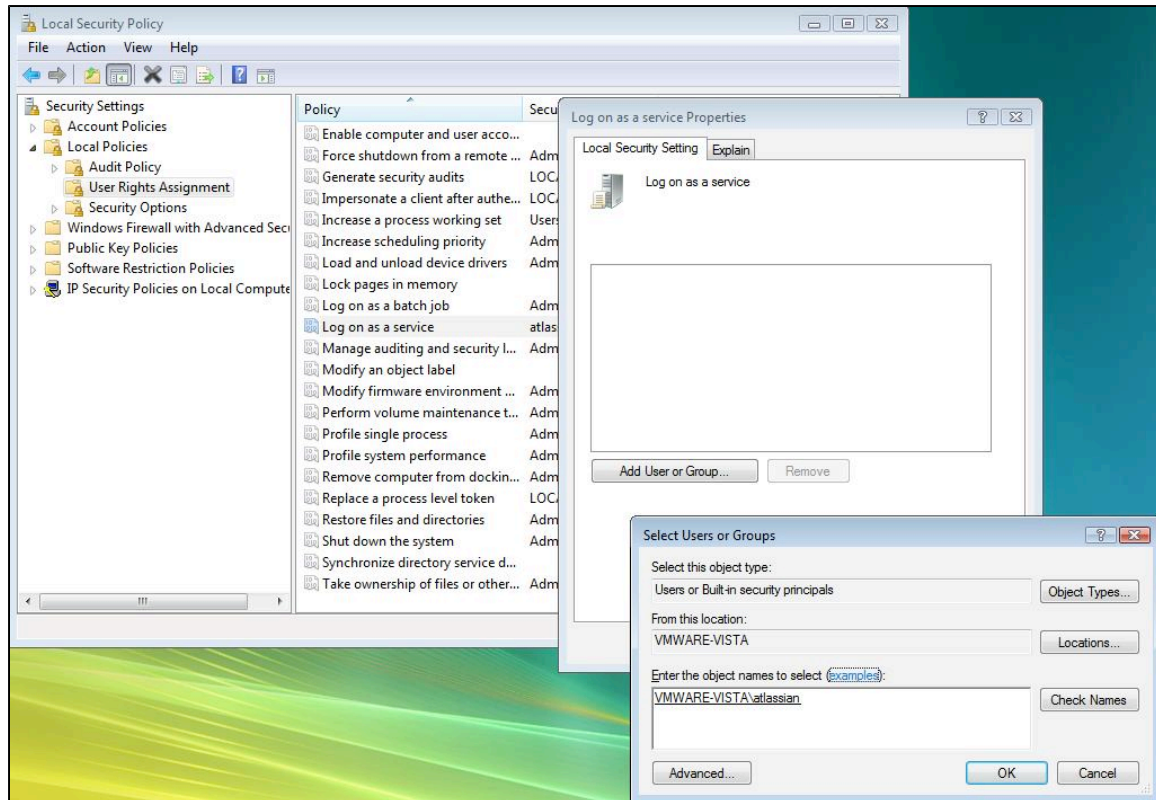
3. The 'Bamboo build server Properties' window will display (see screenshot above). Select the '**This account**' option, provide a local admin account credentials, and click the '**OK**' button to apply your changes.

3. Give the local user access to "logon as a service"

1. Go to Start -> Run and enter '**secpol.msc**'
2. The 'Local Security Settings' window will display. Expand the '**Local Policies**' tree and click '**User Rights Assignment**'.
3. Scroll down and find the '**Logon As a Service**' Policy (see screenshot below). Double-click the '**Log on as a service**' policy.



4. The properties window for the 'Log on as a service' policy will display (see screenshot below). Click the '**Add User or Group**' button.



5. The 'Select Users or Groups' window will display (see screenshot above). Enter your local user and click 'OK' to allow your user to "logon as a service".
6. Click 'OK' and close all open windows.

Bamboo will now start as service, under the local user.

Setting up JNDI mail on JBoss 4.2.2

This page explains how to set up the (gmail) mail service for JBoss 4.2.2. with the following features

- smtp over SSL
- TLS encryption

After installing bamboo.war on jboss modify <jboss-install>/server/default/deploy/mail-service.xml to be

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: mail-service.xml 62349 2007-04-15 16:48:15Z dimitris@jboss.org $ -->
<server>

    <!-- ===== -->
    <!-- Mail Connection Factory -->
    <!-- ===== -->

    <mbean code="org.jboss.mail.MailService"
          name="jboss:service=Mail">
        <attribute name="JNDIName">java:/Mail</attribute>
        <attribute name="User">${account}@gmail.com</attribute>
        <attribute name="Password">${pw}</attribute>
        <attribute name="Configuration">
            <!-- A test configuration -->
            <configuration>
                <!-- Change to your mail server protocol -->
                <property name="mail.transport.protocol" value="smtp"/>

                <!-- Change to the user who will receive mail -->
                <property name="mail.user" value="${account}@gmail.com"/>

                <!-- Change to the SMTP gateway server -->
                <property name="mail.smtp.host" value="smtp.gmail.com"/>

                <!-- The mail server port -->
                <property name="mail.smtp.port" value="465"/>

                <!-- Change to the address mail will be from -->
                <property name="mail.from" value="${account}@whatever.com"/>

                <property name="mail.smtp.auth" value="true"/>
                <property name="mail.smtp.user" value="${account}@gmail.com"/>
                <property name="mail.smtp.password" value="${pw}"/>
                <property name="mail.smtp.ssl.enable" value="true"></property>
                <property name="mail.smtp.socketFactory.class"
value="javax.net.ssl.SSLSocketFactory"></property>
                <property name="mail.smtp.starttls.enable" value="true"/>

                <!-- Enable debugging output from the javamail classes -->
                <property name="mail.debug" value="false"/>
            </configuration>
        </attribute>
        <depends>jboss:service=Naming</depends>
    </mbean>

</server>

```

Problems

If you encounter a class loading problem you will need to remove

- <bamboo-war>/WEB-INF/lib/activation-x.x.x.jar
- <bamboo-war>/WEB-INF/lib/mail-x.x.x.jar

from bamboo.war to avoid the clash with jboss' native libraries.

Setting up JNDI on Jetty

The Bamboo start up script can be customised to setup JNDI resources

Follow [this guide](#) to setup Bamboo to use the jetty.xml file

You will also need to change the jetty.xml file under webapp/WEB-INF/classes by change the context path from /bamboo to /. Example of this

is below:

If you are using Bamboo 1.2.4 (or earlier):

```
<!-- Add root context web applications. -->
<Call name="addWebApplication">
  <Arg>/</Arg>
  <Arg>
    <SystemProperty name="bamboo.webapp" default="webapp"/>
  </Arg>
</Call>
```

If you are using Bamboo 2.0:

```
<Call name="addHandler">
  <Arg>
    <New class="org.mortbay.jetty.webapp.WebAppContext">
      <Arg name="webApp">
        <SystemProperty name="bamboo.webapp" default="webapp"/>
      </Arg>
      <Arg name="contextPath">
        /
      </Arg>
    </New>
  </Arg>
</Call>
```

To set up the JNDI mail session, you will also need to uncomment and modify the section of this `jetty.xml` shown below. You will need to replace the values inside the `<Arg>` tags with appropriate values (username, password, host, from address).

In Bamboo 1.2.4 and earlier:

```
<!--
    <Call name="addService">
      <Arg>
        ...
      </Arg>
    </Call>
-->
```

In Bamboo 2.0:

```
<!--
<New id="resourceID" class="org.mortbay.jetty.plus.naming.Resource">
  <Arg>resourceName</Arg>
  <Arg>
    <New class="org.mortbay.naming.factories.MailSessionReference">
      ...
    </New>
  </Arg>
</New>
-->
```

If you are experiencing class loading problems with your mail session. Try uncommenting the following line in the web applications context

(2.0 only):

```
<!--<Set name="parentLoaderPriority">true</Set>-->
```

Support Policies

Welcome to the support policies index page. Here, you'll find information about how Atlassian Support can help you and how to get in touch with our helpful support engineers. Please choose the relevant page below to find out more.

- [Bamboo Support Policy](#)
- [Bug Fixing Policy](#)
- [Deploying Multiple Atlassian Applications in a Single Tomcat Container](#)
- [How to Report a Security Issue](#)
- [New Features Policy](#)
- [Patch Policy](#)
- [Security Advisory Publishing Policy](#)
- [Security Patch Policy](#)
- [Severity Levels for Security Issues](#)

To request support from Atlassian, please raise a support issue in our online support system. To do this, visit support.atlassian.com, log in (creating an account if need be) and create an issue under Bamboo. Our friendly support engineers will get right back to you with an answer.

Bamboo Support Policy

This page contains details about the scope of Bamboo Support.

On this page:

- [Build Failures](#)
- [Distributed Builds](#)
- [EC2](#)
- [Plugins](#)
- [Related Information](#)

Build Failures

Atlassian will provide [Troubleshooting Guide\(s\)](#) and documentation to help customers resolve Bamboo-related issues.

Ultimately, users are responsible for the administration and maintenance of their build systems and infrastructure.

However, if the root cause of the problem is partially or wholly related to Bamboo, we will create a Bug Report or Feature request to address the issue.



Any bug or feature request reported during the course of investigation is subject to Atlassian's [Bug Fixing](#) and [New Features Policies](#), as outlined in the [Atlassian Support Offerings](#) document.

Distributed Builds

The pre-requisites outlined in the Technical Overview section of [Troubleshooting Guide](#) must be met for server/agent communication to work.

If Atlassian determines that a customer's agent connectivity or communication problem results from a network or environmental factor, it is the customer's responsibility to address this problem and keep their network maintained.

EC2

Atlassian does not support [custom elastic images](#) (custom AMIs) and recommends using an [EBS volume](#) to customise your image if desired. While we are happy to assist with issues related to the elastic agent, we can not help troubleshoot modifications to the [stock images](#) which are not directly related to Bamboo functionality.

Plugins

Atlassian offers support for certain third party plugins as listed in our [supported plugins](#) list. For unsupported plugins, issues should be raised with the provider of the plugin.

The following can be classified as being third-party plugins:

- Integration with repositories other than Subversion, CVS and Perforce.
- Third party builders, test and code coverage tools other than what is shipped with Bamboo.

Each plugin's supported status is listed on its page in the Plugin Exchange.

Related Information

[Atlassian Bug Fixing Policy](#)

[New Features Policy](#)

[Patch Policy](#)

[Atlassian Support Offerings](#)

Bug Fixing Policy

Summary

- Atlassian Support will help with workarounds and bug reporting.
- Critical bugs will generally be fixed in the next maintenance release.
- Non critical bugs will be scheduled according to a variety of considerations.



Raising a Bug Report

Atlassian Support is eager and happy to help verify bugs — we take pride in it! Please open a support request in our [support system](#) providing as much information as possible about how to replicate the problem you are experiencing. We will replicate the bug to verify, then lodge the report for you. We'll also try to construct workarounds if they're possible.

Customers and plugin developers are also welcome to open bug reports on our issue tracking systems directly. Use <http://jira.atlassian.com> for the stand-alone products and <http://studio.atlassian.com> for JIRA Studio and Atlassian OnDemand.

When raising a new bug, you should rate the priority of a bug according to our [JIRA usage guidelines](#). Customers [should watch](#) a filed bug in order to receive e-mail notification when a "Fix Version" is scheduled for release.

How Atlassian Approaches Bug Fixing

Maintenance (bug fix) releases come out more frequently than major releases and attempt to target the most critical bugs affecting our customers. The notation for a maintenance release is the final number in the version (ie the 1 in 3.0.1).

If a bug is critical (production application down or major malfunction causing business revenue loss or high numbers of staff unable to perform their normal functions) then it will be fixed in the next maintenance release provided that:

- The fix is technically feasible (i.e. it doesn't require a major architectural change).
- It does not impact the quality or integrity of a product.

For non-critical bugs, the developer assigned to fixing bugs prioritises the non-critical bug according to these factors:

- How many of our supported configurations are affected by the problem.
- Whether there is an effective workaround or patch.
- How difficult the issue is to fix.
- Whether many bugs in one area can be fixed at one time.

The developers responsible for bug fixing also monitor comments on existing bugs and new bugs submitted in JIRA, so you can provide feedback in this way. We give high priority consideration to [security issues](#).

When considering the priority of a non-critical bug we try to determine a 'value' score for a bug which takes into account the severity of the bug from the customer's perspective, how prevalent the bug is and whether roadmap features may render the bug obsolete. We combine this with a complexity score (i.e. how difficult the bug is). These two dimensions are used when developers self serve from the bug pile.

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Deploying Multiple Atlassian Applications in a Single Tomcat Container

Deploying multiple Atlassian applications in a single Tomcat container is **not supported**. We do not test this configuration and upgrading any of the applications (even for point releases) is likely to break it. There are also a number of known issues with this configuration:

- You may not be able to start up all of the applications in the container, due to class conflicts (in 3rd party libraries bundled with our application) that result from the Atlassian applications sharing a single JVM in the Tomcat container.
- You will not be able to determine the startup order of the applications. Hence, you may experience problems such as JIRA starting before Crowd, rather than vice versa.
- Memory problems are also common as one application may allocate all of the memory in the Tomcat JVM to itself, starving the other applications.

We also do not support deploying multiple Atlassian applications to a single Tomcat container for a number of practical reasons. Firstly, you must shut down Tomcat to upgrade any application and secondly, if one application crashes, the other applications running in that Tomcat container will be inaccessible.

Finally, we recommend not deploying *any other applications* to the same Tomcat container that runs the Atlassian application, especially if these other applications have large memory requirements or require additional libraries in Tomcat's `lib` subdirectory.

How to Report a Security Issue

Finding and Reporting a Security Issue

If you find a security issue in the product, open an issue on <http://jira.atlassian.com> in the relevant project.

- Set the **security level** of the bug to '**Reporters and Developers**'.
- Set the priority of the bug to 'Blocker'.
- Provide as much information on reproducing the bug as possible.

All communication about the security issue should be performed through JIRA, so that Atlassian can keep track of the issue and get a patch out as soon as possible.

If you cannot find the right project to file your issue in, email the details to security@atlassian.com.



When reporting a security vulnerability, please keep in mind the following:

We need a technical description that allows us to assess exploitability and impact of the issue.

- Provide steps to reproduce the issue, including any URLs or code involved.
- If you are reporting an XSS, your exploit should at least pop up an alert in the browser. It is much better if the XSS exploit shows user's authentication cookie.
- If you are reporting an SQL injection, we want to see the exploit extracting database data, not just producing an error message.
- HTTP request / response captures or simply packet captures are also very useful to us.

Please refrain from sending us links to non-Atlassian web sites, or reports in PDF / DOC / EXE files. Image files are ok. Make sure the bug is exploitable by someone other than the user himself (e.g. "self-XSS").

Without this information it is not possible to assess your report and it is unlikely to be addressed.

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

New Features Policy

Summary

- We encourage and display customer comments and votes openly in our issue tracking system, <http://jira.atlassian.com>.
- We do not publish roadmaps.
- Product Managers review our most popular voted issues on a regular basis.
- We schedule features based on a variety of factors.
- Our [Atlassian Bug Fixing Policy](#) is distinct from our Feature Request process.
- Atlassian provides consistent updates on the top 20 feature/improvement requests (in our issue tracker systems).

How to Track what Features are Being Implemented

When a new feature or improvement is scheduled, the 'fix-for' version will be indicated in the JIRA issue. This happens for the upcoming release only. We maintain roadmaps for more distant releases internally, but because these roadmaps are often pre-empted by changing customer demands, we do not publish them.

How Atlassian Chooses What to Implement

In every [major release](#) we *aim* to implement highly requested features, but it is not the only determining factor. Other factors include:

- **Customer contact:** We get the chance to meet customers and hear their successes and challenges at Atlassian Summit, Atlassian Unite, developer conferences, and road shows.
- **Customer interviews:** All product managers at Atlassian do customer interviews. Our interviews are not simply to capture a list of features, but to understand our customers' goals and plans.
- **Community forums:** There are large volumes of posts on [answers](#), of votes and comments on jira.atlassian.com, and of conversations on community forums like groups on LinkedIn.
- **Customer Support:** Our support team provides clear insights into the issues that are challenging for customers, and which are generating the most calls to support.
- **Atlassian Experts:** Our [Experts](#) provide insights into real-world customer deployments, especially for customers at scale.
- **Evaluator Feedback:** When someone new tries our products, we want to know what they liked and disliked and often reach out to them for more detail.
- **In product feedback:** The [JIRA Issue Collectors](#) that we embed our products for evaluators and our Early Access Program give us a constant pulse on how users are experiencing our product.
- **Usage data:** Are customers using the features we have developed?
- **Product strategy:** Our long-term strategic vision for the product.

How to Contribute to Feature Development

Influencing Atlassian's release cycle

We encourage our customers to vote on feature requests in JIRA. The current tally of votes is available online in our issue tracking system, <http://jira.atlassian.com>. Find out if your improvement request [already exists](#). If it does, please vote for it. If you do not find it, [create a new feature or improvement request](#) online.

Extending Atlassian Products

Atlassian products have powerful and flexible extension APIs. If you would like to see a particular feature implemented, it may be possible to develop the feature as a plugin. Documentation regarding the [plugin APIs](#) is available. Advice on extending either product may be available on the user mailing-lists, or at [Atlassian Answers](#).

If you require significant customisations, you may wish to get in touch with our [partners](#). They specialise in extending Atlassian products and can do this work for you. If you are interested, please [contact us](#).

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Patch Policy

Patch Policy

Atlassian will only provide software patches in extremely unusual circumstances. If a problem has been fixed in a newer release of the product, Atlassian will request that you upgrade your instance to fix the issue. If it is deemed necessary to provide a patch, a patch will be provided for the current release and the last maintenance release of the last major version only.

Patches are issued under the following conditions:

- The bug is critical (production application down or major malfunction causing business revenue loss or high numbers of staff unable to perform their normal functions).
AND
- A patch is technically feasible (i.e., it doesn't require a major architectural change)
OR
- The issue is a security issue, and falls under our [Security Patch Policy](#).

Atlassian does not provide patches for non-critical bugs.

Provided that a patch does not impact the quality or integrity of a product, Atlassian will ensure that patches supplied to customers are added to the next maintenance release. Customers [should watch](#) a filed bug in order to receive e-mail notification when a "Fix Version" is scheduled for release.

Patches are generally attached to the relevant <http://jira.atlassian.com> issue.

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Security Advisory Publishing Policy

Publication of Security Advisories

When a [critical severity](#) security vulnerability in an Atlassian product is discovered and resolved, Atlassian will inform customers through the following mechanisms:

- We will post a security advisory in the latest documentation of the affected product at the same time as releasing a fix for the vulnerability.
- We will send a copy of all posted security advisories to the **'Technical Alerts' mailing list** for the product concerned.
Note: To manage your email subscriptions and ensure you are on this list, please go to my.atlassian.com and click 'Communications Centre' near the top right of the page.
- If the person who reported the vulnerability wants to publish an advisory through some other agency, such as [CERT](#), we will assist in the production of that advisory and link to it from our own.

If you want to track non-critical severity security vulnerabilities, you need to monitor the issue trackers for the relevant products on <http://jira.atlassian.com>. For example, <https://jira.atlassian.com/browse/JRA> for JIRA and <https://jira.atlassian.com/browse/CONF> for Confluence. Security issues in trackers will be marked with a "security" label. All security issues will be listed in the release notes of the release where they have been fixed, similar to other bugs.

One of the ways to monitor updates to security issues is subscribing to the results of a [sample search](#) via email or RSS.

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Security Patch Policy

Product Security Patch Policy

Atlassian makes it a priority to ensure that customers' systems cannot be compromised by exploiting vulnerabilities in Atlassian products.

Scope

This page describes when and how we release security patches and security upgrades for our products. It does not describe the whole of disclosure process that we follow. It also excludes OnDemand, since OnDemand will always be patched by Atlassian without additional notifications.

Critical vulnerabilities

When a **Critical** security vulnerability is discovered by Atlassian or reported by a third party, Atlassian will do all of the following:

- Issue a new, fixed release for the current version of the affected product as soon as possible, usually in a few days.
- Issue a binary patch for the current release.
- Issue a binary patch for the latest maintenance release of the previous version of the product.
- Patches for older versions or releases normally will not be issued.

Patches will be attached to the relevant JIRA issue. You can use these patches as a "stop-gap" measure until you upgrade your installation in order to fully fix the vulnerability.

Non-critical vulnerabilities

When a security issue of a **High, Medium or Low** severity is discovered, Atlassian will do all of the following:

- Include the fix into the next scheduled release, both for the current and previous maintenance versions.
- Where practical, provide new versions of plugins or other components of the product that can be upgraded independently.

You should upgrade your installation in order to fix the vulnerability.

Other information

Severity level of vulnerabilities is calculated based on [Severity Levels for Security Issues](#).

Visit our general [Atlassian Patch Policy](#) as well.

Examples

Example 1: A critical severity vulnerability is found in a (hypothetical current release) JIRA 5.3.2. The last bugfix release in 5.2.x branch was 5.2.3. In this case, a patch will be created for 5.3.2 and 5.2.3. In addition, new bugfix releases, 5.3.3 and 5.2.4, which are free from this vulnerability, will be created in a few days.

Example 2: A high or medium severity vulnerability is found in the same release as in the previous example. The fix will be included into the currently scheduled releases 5.3.3 and 5.2.4. Release schedule will not be brought forward and no patches will be issued. If the vulnerability is in a plugin module, then a plugin upgrade package may still be supplied.

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Severity Levels for Security Issues

Severity Levels

Atlassian security advisories include a severity level. This severity level is based on our self-calculated CVSS score for each specific vulnerability. CVSS is an industry standard vulnerability metric. You can learn more about CVSS at [FIRST.org](#) web site.

CVSS scores are mapped into the following severity ratings:

- Critical
- High
- Medium
- Low

An approximate mapping guideline is as follows:

CVSS score range	Severity in advisory
0 – 2.9	Low
3 – 5.9	Medium
6.0 – 7.9	High
8.0 – 10.0	Critical

Below is a summary of the factors which illustrate types of vulnerabilities usually resulting in a specific severity level. Please keep in mind that this rating does not take into account details of your installation.

Severity Level: Critical

Vulnerabilities that score in the critical range usually have most of the following characteristics:

- Exploitation of the vulnerability results in root-level compromise of servers or infrastructure devices.
- The information required in order to exploit the vulnerability, such as example code, is widely available to attackers.
- Exploitation is usually straightforward, in the sense that the attacker does not need any special authentication credentials or knowledge about individual victims, and does not need to persuade a target user, for example via social engineering, into performing any special functions.

For critical vulnerabilities, is advised that you patch or upgrade as soon as possible, unless you have other mitigating measures in place. For example, if your installation is not accessible from the Internet, this may be a mitigating factor.

Severity Level: High

Vulnerabilities that score in the high range usually have some of the following characteristics:

- The vulnerability is difficult to exploit.
- Exploitation does not result in elevated privileges.
- Exploitation does not result in a significant data loss.

Severity Level: Medium

Vulnerabilities that score in the medium range usually have some of the following characteristics:

- Denial of service vulnerabilities that are difficult to set up.
- Exploits that require an attacker to reside on the same local network as the victim.
- Vulnerabilities that affect only nonstandard configurations or obscure applications.
- Vulnerabilities that require the attacker to manipulate individual victims via social engineering tactics.
- Vulnerabilities where exploitation provides only very limited access.

Severity Level: Low

Vulnerabilities in the low range typically have very little impact on an organisation's business. Exploitation of such vulnerabilities usually requires local or physical system access.

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Usage FAQ

- [Backing up Bamboo instances over 4GB](#)
- [Bamboo Database Schema](#)
- [Binding Bamboo to one IP address](#)
- [Can Bamboo build and test non-Java projects](#)
- [Can multiple plans share a common 3rd-party directory](#)
- [Changing Bamboo database settings](#)
- [Finding the Support Entitlement Number \(SEN\)](#)
- [How do I manually set the version of new Subversion workspaces](#)
- [Securing your repository connection](#)
- [Changing the remote agent heartbeat interval](#)
- [Cloning a Bamboo instance](#)
- [CVS Error logging in Bamboo](#)
- [Do I have to upgrade all remote agents for Bamboo Release 2.1.2](#)
- [Enable User Management debug logging in Bamboo](#)
- [Hibernate errors in logs after upgrading to Bamboo 2.0](#)
- [How do I construct a cron expression in Bamboo](#)
- [How do I disable SSH access to my elastic instances](#)
- [How do I shut down my elastic instances if I have restarted my Bamboo server](#)
- [How do I stop Bamboo from shutting itself down and restarting](#)
- [How do I stop the Bamboo server from automatically configuring my remote agent's capabilities](#)
- [JUnit parsing in Bamboo](#)
- [Known issues with CVS in Bamboo](#)
- [Monitoring and Profiling Bamboo](#)
- [Monitor Memory usage and Garbage Collection in Bamboo](#)
- [Moving Bamboo-Home of an agent](#)
- [Performing a thread dump.](#)
- [Removing Coverage plug-in data from the Bamboo database](#)
- [Restoring passwords to recover admin users](#)
- [Send Errors to stderr - Script Builder in Visual Studio WinXP to build Solutions Files](#)
- [Using Bamboo with Clover](#)
 - [Getting gcov results in Clover coverage summary](#)
- [Working with Sun JAVA libraries](#)
- [Bamboo indicates that my Ant or Maven builds failed, even though they were successful](#)
- [How can I pass bamboo variables to my build script](#)

Backing up Bamboo instances over 4GB

Due to limitations of the original ZIP file format, and the TrueZIP library used to generate ZIP files, it is not possible to [export a Bamboo instance](#) when the resulting ZIP file, or the original size of any of its components, is larger than 4GB. Instead, you will need to backup Bamboo manually. **We strongly recommend performing regular backups.**

To backup Bamboo manually:

1. Shut down Bamboo.
2. Copy the contents of your `<Bamboo-Home>` directory. You can delete bamboo's working copy folder `<bamboo-home>/xml-data/build-dir` beforehand to reduce the size. Zip this backup.
3. If you are using an [external database](#), use the database's native backup tool to backup your database (please consult your database documentation for further instructions). Alternatively, perform an SQL dump of your database.

To restore your Bamboo instance to a previous state:

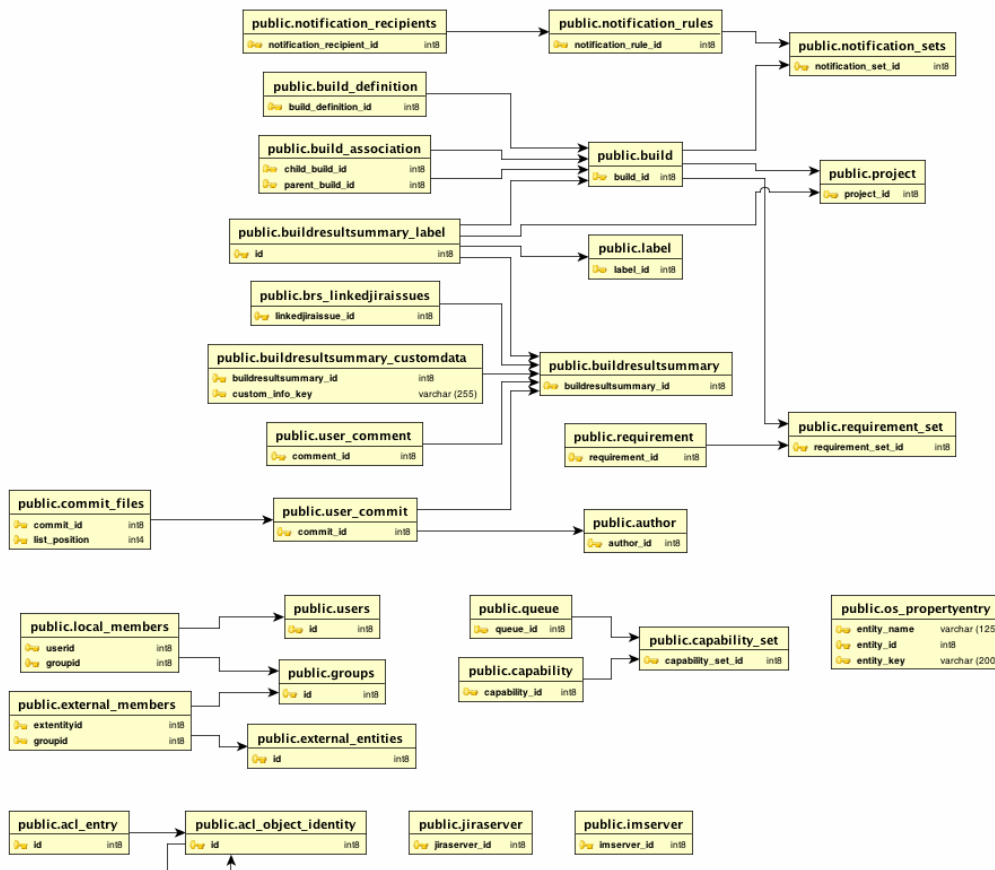
1. Edit the `../Bamboo-Install-Directory/webapp/WEB-INF/classes/bamboo.init.properties` file to point to your backed-up [Bamboo-Home](#) directory.
2. If you are using an external database, restore your database using the database's native backup tool.

Bamboo Database Schema



This documentation is now deprecated, if you want to investigate how Bamboo DB Schema please see [this kb](#).

Bamboo 2.1 Database schema



Binding Bamboo to one IP address

i These instructions apply to the Bamboo distribution (not EAR-WAR), which ships with the Jetty application server.

If you have installed Bamboo on a machine with multiple interfaces, and need to bind Bamboo to a single IP address, follow these instructions.

Step 1 — Instruct Bamboo to read its configuration from the `jetty.xml` file

By default Bamboo doesn't use the `jetty.xml` file to configure itself. You will need to [tell Bamboo to use it](#).

Step 2 — Edit the `Jetty.xml` file

Your `jetty.xml` file is located in `<Bamboo_Install_directory>/webapp/WEB-INF/classes/jetty.xml`.

Please note: YOUR_HOST_URL should be the same as the [Bamboo base URL](#) configured in Bamboo.

If you are using Bamboo 1.2.4:

Find the following section:

```
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Port">
        <SystemProperty name="jetty.port" default="8085"/>
      </Set>
      <Set name="Host">127.0.0.1</Set>
```

Change the last line as follows:

```
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Port">
        <SystemProperty name="jetty.port" default="8085"/>
      </Set>
      <Set name="Host">YOUR_HOST_URL</Set>
```

If you are using Bamboo 2.0

Find the following section:

```
<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.bio.SocketConnector">
      <Set name="Port">
        <SystemProperty name="jetty.port" default="8085"/>
      </Set>
      <!--<Set name="Host">127.0.0.1</Set-->
```

Uncomment the host property as follows:

```
<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.bio.SocketConnector">
      <Set name="Port">
        <SystemProperty name="jetty.port" default="8085"/>
      </Set>
      <Set name="Host">YOUR_HOST_URL</Set>
```

Step 3 — Restart Bamboo

If you have any [elastic agents](#) running, ensure that they are [shut down](#) before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become [orphaned from your Bamboo server](#).

Can Bamboo build and test non-Java projects

Bamboo can be ported to be used on any architecture and can build projects in virtually any language/script (Java, C++, ruby, perl, VB.net, bash, make and C# to name a few of many projects currently built with Bamboo).

Bamboo can execute any script/build that has a return code after the build process is completed. Ideally, you would configure a build tool (such as Maven or Ant) to build your code. Bamboo will then call on the build tool to build your project (depending on how your build process

is configured).

Regarding tests, Bamboo uses JUnit tests to integrate test results with Java and is capable of reading test results from any testing framework that outputs to a JUnit XML report.

Can multiple plans share a common 3rd-party directory

For example, you might have three repository directories, say, A, B, and C, where A is a common 3rd-party library. A is used across projects.

At this stage, Bamboo doesn't support having multiple checkout directories per build plan. However, you can work around this by setting these three directories up as separate Bamboo build plans - P_A, P_B and P_C.

To make this work, you will also need to specify as an argument to your build scripts for P_B and P_C the location of A, which will be something like this:

../Plan_key_for_A/

Using a set up like this, your library module (A) should only be checked out once across the Bamboo instance.

See also:

[Triggering a build when another build finishes](#)

Changing Bamboo database settings

The Bamboo database configuration is persisted in the <Bamboo-Home>/bamboo.cfg.xml file. You can change the database settings by editing this file, as detailed in the instructions below:

Changing the Bamboo database username and password.

If you want to change the database username and password, edit the following line,

```
<property name="hibernate.connection.password">YOUR_PASSWORD</property>
  <property
name="hibernate.connection.username">YOUR_USERNAME</property>
```

Changing the Bamboo database URL

If you want to change the database URL, edit the following line,

```
<property name="hibernate.connection.url">DATABASE_URL</property>
```

 *You need to restart the Bamboo application server for the changes to take effect. If you have any [elastic agents](#) running, ensure that they are [shut down](#) before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become [orphaned from your Bamboo server](#).*

Finding the Support Entitlement Number (SEN)

If you have a current Bamboo maintenance license under another account please supply the details of the licensee and the current Support Entitlement Number (SEN)

- Your Support Entitlement Number (SEN) is listed on the third page of your Atlassian Invoice.

OR

- Log into <http://my.atlassian.com> to find the SEN for a specific license

ATLASSIAN Products Studio About My Account

Licenses Quotes & Invoices My Profile Switch User

Welcome
Hey Donna McGahan, welcome to the Atlassian customer portal. You can manage your 4 licenses below.

Licenses Expand All | Collapse All

Click here to expand license info and find your SEN

Product
Bamboo Standard (Unlimited Local Agents, 1 Remote Agent): Demonstration
FishEye Unlimited (Unlimited Users): Commercial License
JIRA Enterprise: Demonstration

How do I manually set the version of new Subversion workspaces

You can manually set the version of any new Subversion workspaces created by Bamboo on checkout. Bamboo automatically upgrades any source code it checks out, to be compatible with a particular version of Subversion. If you use an older Subversion client to access the code checked out by Bamboo, you will need to force any new Subversion workspaces to be created with the SVN version that you wish to retain. Otherwise, if you then use an older Subversion client to access this code, any Bamboo builds on that code may fail.

If you want to prevent Bamboo from automatically upgrading any source code checked out, you will need to run Bamboo with the following [system property](#):

```
-Dbamboo.svn.wc.format=X.X
```

where X.X is the SVN version that you want to retain for your code. Valid values for this parameter are 1.3, 1.4, 1.5 and 1.6.

To change this parameter for your Bamboo instance:

- Add the parameter with a '-D' prefix and appropriate value, in your command line when starting Bamboo. e.g.
-Dbamboo.svn.wc.format=1.5, or

! Please note, setting this parameter will only affect any Subversion workspaces created after the parameter has been set. It will not change the version of any Subversion workspaces that have already been created. And the parameter needs to be set on the Bamboo server in case of an local build and remote agent in case of a remote agent build.

For example, setting this parameter to **1.5** tells Bamboo to:

- check out code to version 1.5 if no working copy exists, and
- not to automatically upgrade any already checked out code of an existing working copy to be compatible with Subversion 1.6.

Securing your repository connection

About this page

This page shows how to secure your bamboo server to source repository connection.

Subversion

svn+ssh

In your build plan you must specify the absolute path to the repository when using svn+ssh, for example `svn+ssh://<svnhost>/absolute/path/to/repository/root/your/module`

Using a key pair

The key pair is shared between your bamboo agent box (the bamboo server box in case of local agents) and the repository server box. Your [repository configuration](#) allows you to specify the location of a private key file that must be stored on the agent box.

The key pair has to be in PKCS12/OpenSSH format and the private key must be passphrase protected, otherwise a runtime exception is thrown by JDK security engine while opening the user key.

Linux and related

1. On the repository box generate the keypair

```
ssh-keygen -t rsa
```

2. add public key to ~/.ssh/authorized_keys

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

3. copy the private key to all the agent boxes **into a directory that is common to all agents (remote and local)** e.g.
/var/keys/ssh/id_rsa



For windows agents

Store the private key file in the same location **on the drive that the agent is started from**. For example you start your agent with

```
d:\bamboo-agent > java -jar atlassian-bamboo-agent-installer-xxx.jar ....
```

Then the key file must be in d:\var\keys\ssh\id_rsa

Windows

Private key should always be in OpenSSH format. On windows usually "putty" (plink) program is used that uses keys in its proprietary format (PPK - putty private key), this format is not supported by bamboo. The PuttyGen program may be used on Windows to convert key in PPK format to OpenSSH.

How to add the public key to the windows version of ~/.ssh/authorized_keys <<<< comment needed

Trouble shooting

You can test the svn+ssh connection from the command line.

First you need to tell the svn command line client which key file to use:

```
$ export SVN_SSH="ssh -i /absolute/path/to/private/key"
```

Then you can test the connection with

```
$ svn list svn+ssh://<svn-server>/Absolute/Path/To/Repository/[Module]
```

Changing the remote agent heartbeat interval

Remote agents periodically send a "heartbeat" signal to the Bamboo server. This is vital for tracking whether your remote agents are online or offline. The remote heartbeat is **asynchronous**, which means that if a remote agent goes offline and comes back online again it will reconnect instead of being shut down (as long as the same server is available).

However, you may wish to adjust the time parameters for the remote agent heartbeat, particularly if you have a lot of network activity already.



You need to be running **Bamboo 2.0.6** or above to adjust the following remote agent heartbeat parameters.

There are three configurable parameters on the bamboo server for the remote agent heartbeat:

- `bamboo.agent.heartbeatInterval` — This parameter governs the frequency of the heartbeat signal from the remote agents. This parameter is specified in seconds with the default being **5 seconds**.
- `bamboo.agent.heartbeatTimeoutSeconds` — This parameter governs how long the Bamboo server will wait before it times out an agent that it hasn't received a heartbeat signal from. A remote agent that has been timed out will be marked as 'Offline'. Any builds being run by agents which have timed out will be abandoned. This parameter is specified in seconds with the default being **60 0 seconds**.
- `bamboo.agent.heartbeatCheckInterval` — This parameter governs how often Bamboo checks for agents that have exceeded the heartbeat timeout specified in `bamboo.agent.heartbeatTimeoutSeconds`. This parameter is specified in seconds with the default being **30 seconds**.

Please read the [Configuring system properties](#) page for instructions on how to change a remote agent heartbeat parameter for your Bamboo server.

Cloning a Bamboo instance

In case you need to clone your production instance to a test/staging instance in order to prepare migrating to another database or upgrading Bamboo.

For example, you may want to transfer your current production snapshot to a test server as permitted in the [license agreement](#).



- We strongly recommend to duplicate Bamboo first and *then* apply changes such as upgrade or migrating to another database.
- If you are using JIRA or Crowd for user management, the URL of the Bamboo server may change when you clone the Bamboo instance, in which case you will need to edit that setting for the Bamboo application in JIRA/CROWD to match the new URL.

License

Development licenses are available for any Commercial or Academic license. [Create one](#) or [contact us](#) for help.

Clone production instance - standard

This is the simple and straight forward way to clone your instance

1. Export/Backup your current instance.
2. Copy zip across to new server.
3. Install **the same version** of Bamboo on new server.
4. Point Bamboo.home in <bamboo-install>/webapp/WEB-INF/classes/bamboo-init.properties to your **new** Bamboo home directory.
5. Start the new instance.
6. Complete the setup wizard, choose 'import existing data'.

Clone production instance - alternative

If your current instance has grown too large and export/import does not work you can still clone your instance using an alternative backup and restore strategy.

The purpose is to clone <bamboo-home> and make it available to the new test/clone instance.

1. Shutdown production Bamboo at a convenient time.
2. Create a backup:

embedded DB	external DB
(Optional) Reduce the size of this zip by deleting the xml-data/build-dir – this directory only contains working copies of checked out sources. Zip <bamboo-production-home> directory; the embedded database will be part of this zip.	(Optional) Reduce the size of this zip by deleting the xml-data/build-dir - this directory only contains working copies of checked out sources. Zip <bamboo-production-home> directory. Create a backup with the native tools provided by your DB.

3. Restart production Bamboo.
4. Transfer the home.zip to your cloned instance and unzip into <bamboo-clone-home>.
5. (External DB only) Create a new database for the cloned instance and import the db dump.
6. Edit <bamboo-clone-home>/bamboo.cfg.xml and <bamboo-clone-home>/xml-data/configuration/administration.xml and change the server names/ip addresses according to the new location.
7. (External DB only) Edit <bamboo-clone-home>/bamboo.cfg.xml and enter the new database connection details and credentials.
8. Point bamboo.home of your cloned instance to the unzipped <bamboo-clone-home> directory.
9. Start the Bamboo clone.

This should give you a perfectly cloned instance.

Your next steps

- If the new server has different locations for
 - JDKs
 - Ant
 - Maven
 - Perforce
 - Msbuild toolsadjust the settings in the [server capabilities settings](#) to match the locations on the new machine.
- From here you can [upgrade](#) if desired.
- After the upgrade you should be able to export your instance without problems and then migrate to another database for instance.

CVS Error logging in Bamboo

Currently, if the server throws an error during a CVS build in Bamboo versions 2.0.x, the application will hang with no indication of any

checkout/update problems. There is an open [JIRA issue](#) tracking this problem.

In order to further debug any CVS issues, you will need to turn up the CVS logging by passing in the `-DcvsClientLog=system` argument to Bamboo. Please edit `<bamboo-install>/bamboo.sh` accordingly.

Do I have to upgrade all remote agents for Bamboo Release 2.1.2

We have improved the availability and reliability of remote agents in this release, by adding a failover to reconnect agents when the network drops out.

By default, remote agents now use [ActiveMQ failover mechanism](#) to reconnect.

it's not essential to upgrade the agent jar. The agent will automatically download the changed code from the server.

The agent has a special classloader that actually downloads classes from the server. The JAR file on the agent only contains a handful of classes it needs to bootstrap itself.

Enable User Management debug logging in Bamboo

This page describes how to turn on user management debug logging.

Edit `<bamboo-install>/webapp/WEB-INF/classes/log4j.properties` and append the following lines

```
log4j.logger.bucket.user=DEBUG
log4j.category.com.atlassian.user=DEBUG
log4j.category.com.atlassian.bamboo.user.BambooUserManagerImpl=DEBUG
```

Restart bamboo.

Hibernate errors in logs after upgrading to Bamboo 2.0

If you are upgrading to Bamboo 2.0 from Bamboo 1.2.4 by pointing to your Bamboo-Home, you may see the following errors in your logs.

```
2008-02-21 09:13:39,890 ERROR \[main\] \[SchemaUpdate\] Unsuccessful: alter table
USER_COMMIT add constraint FKf8936c2ba958b29f foreign key (BUILDRESULTSUMMARY_ID)
references BUILDRESULTSUMMARY
2008-02-21 09:13:39,891 ERROR \[main\] \[SchemaUpdate\] Constraint already exists
in statement \[alter table USER_COMMIT add constraint FKf8936c2ba958b29f foreign
key (BUILDRESULTSUMMARY_ID) references BUILDRESULTSUMMARY\]
2008-02-21 09:13:39,892 ERROR \[main\] \[SchemaUpdate\] Unsuccessful: alter table
USER_COMMIT add constraint FKf8936c2bfe0c684f foreign key (AUTHOR_ID) references
AUTHOR
2008-02-21 09:13:39,892 ERROR \[main\] \[SchemaUpdate\] Constraint already exists
in statement \[alter table USER_COMMIT add constraint FKf8936c2bfe0c684f foreign
key (AUTHOR_ID) references AUTHOR\]
2008-02-21 09:13:39,893 ERROR \[main\] \[SchemaUpdate\] Unsuccessful: alter table
USER_COMMENT add constraint FK19da09cba958b29f foreign key (BUILDRESULTSUMMARY_ID)
references BUILDRESULTSUMMARY
2008-02-21 09:13:39,894 ERROR \[main\] \[SchemaUpdate\] Constraint already exists
in statement \[alter table USER_COMMENT add constraint FK19da09cba958b29f foreign
key (BUILDRESULTSUMMARY_ID) references BUILDRESULTSUMMARY\]
```

In Bamboo 2.0 we introduced compatibility with Oracle and MS SQL Server, this meant we had to make a few changes to the Bamboo Database schema and as a side-affect of these changes - you might notice hibernate warnings above in your atlassian-bamboo logs while Bamboo starts up. These warnings will not prevent your instance of Bamboo from working correctly, but will display every time Bamboo is started.

If you do wish to remove these warning messages, follow the steps below:

1. [Perform an export](#) from your current Bamboo 2.0 instance.
2. Re-install Bamboo 2.0 on a fresh instance.
3. Import the old exported data into your new Bamboo 2.0 instance.

This will remove the Hibernate Error messages while Bamboo starts up.

How do I construct a cron expression in Bamboo

Cron is a time-based job scheduler used in Unix/Linux computer operating systems with a unique and powerful terminology. A number of scheduling features in Bamboo, such as build expiry and elastic instance scheduling, require you to specify your requirements as a cron-based expression. For example, a cron expression such as `"0 0/30 9-19 ? * MON-FRI"` signifies that a scheduled event will be triggered every half an hour from 9am to 7pm, Monday to Friday.

A cron expression comprises of 6 mandatory and one optional field to specify a schedule. The fields in sequential order are: seconds,

minutes, hours, day-of-month, month, day-of-week and (optional) year, i.e.

<seconds> <minutes> <hours> <day-of-month> <month> <day-of-week> <year (optional)>

Each field can be expressed as an integer (e.g. **1, 2, 3**, etc) and special characters can be used in most fields as well (i.e. **' , - * / ? L W #**).

Bamboo uses **OpenSymphony's Quartz** to schedule cron tasks. The syntax it accepts may vary from other cron implementations. Please refer to the [Quartz CronTrigger Tutorial](#) documentation for further information on each of these parameters and more detailed examples.

How do I disable SSH access to my elastic instances

By default, SSH (Secure Shell) access is enabled for elastic instances, the first time that you use Elastic Bamboo. Access rules for the Amazon Elastic Compute Cloud (EC2) are managed by 'security groups' in the [Amazon Web Services Console](#). You can disable SSH access for your elastic instances by changing the EC2 access rules to remove the **'SSH'** Connection Method from the **'elasticbamboo'** security group.

For instructions on changing the EC2 access rules for Elastic Bamboo, please read the [Elastic Bamboo Security](#) document.

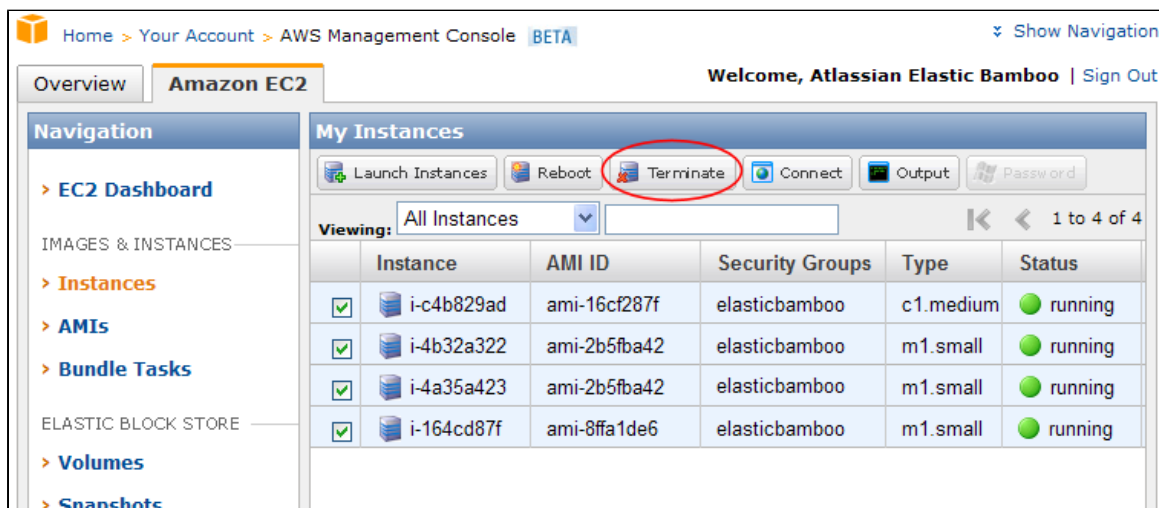
How do I shut down my elastic instances if I have restarted my Bamboo server

If you restart your Bamboo server without shutting down your elastic instances first, your elastic instances will continue to run. Your elastic instances will also be orphaned from your Bamboo server, and you will not be able to shut them down via Bamboo after your Bamboo server has restarted. You will need to terminate them via the Amazon Web Services (AWS) Console.

To shut down an elastic instance via the AWS Console:

1. Log in to the [AWS Console](#). The 'Amazon EC2' tab of the console should display.
2. Click the **Instances** link under the 'Images & Instances' section of the left navigation column. Your EC2 instances should be displayed.
3. Check the checkbox next to the instances that need to be terminated in the 'My Instances' panel. In most cases, it should be all instances unless you are running Elastic Bamboo on multiple Bamboo servers.
4. The buttons at the top of the 'My Instances' panel should become enabled. Click **Terminate** to terminate your instances.

Screenshot: Shutting down an elastic instance via the AWS Console



How do I stop Bamboo from shutting itself down and restarting

If your Bamboo server is shutting itself down and restarting multiple times during the day, you may be experiencing problems with Bamboo's service wrapper.

Symptom

Bamboo server is restarting it self a couple of times per day.
Wrapper reported in the log JVM seems to be hung and will be terminated.
e.g.

```
INFO    | wrapper    | 2009/01/28 15:24:34 | Wrapper Process has not received any CPU
time for 11 seconds.  Extending timeouts.
...
INFO    | jvm 1       | 2009/01/28 15:28:17 | 2009-01-28 15:28:17,299 INFO [ActiveMQ
ShutdownHook] [TransportConnector] Connector vm://bamboo Stopped
INFO    | jvm 1       | 2009/01/28 15:28:17 | 2009-01-28 15:28:17,721 INFO [ActiveMQ
ShutdownHook] [BrokerService] ActiveMQ JMS Message Broker (bamboo,
ID:BSYTEST5-4449-1233174172549-1:0) stopped
STATUS  | wrapper     | 2009/01/28 15:28:18 | <-- Wrapper Stopped
```

Explanation

Bamboo's service wrapper comes with a [timeout](#) that specifies the interval at which the JVM is pinged. A response is expected in that time. If the JVM is too busy, it will not respond to this ping in time.

`wrapper.ping.timeout` defines the timeout in seconds. 0 means that it will never time out. The default value of this setting is **30 seconds**.

Solution

Increase the timeout in the wrapper's configuration. To do this,

1. Edit `<Bamboo-Install>/conf/wrapper.conf`.
2. Add the following line to the end of the `wrapper.conf` file:

```
wrapper.ping.timeout=90
```

3. Restart Bamboo. If you have any [elastic agents](#) running, ensure that they are [shut down](#) before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become [orphaned from your Bamboo server](#).

How do I stop the Bamboo server from automatically configuring my remote agent's capabilities

The Bamboo server automatically detects and populates the capabilities that a remote agent should be configured with upon agent start up. If you have modified the agent capabilities, they will be reset by the server's automatic capability detection when the agent is next restarted.

You can override this by adding the following flag, `"-DDISABLE_AGENT_AUTO_CAPABILITY_DETECTION=true"`, to the Bamboo server. Read [Configuring system properties](#) for information on how to do this.

JUnit parsing in Bamboo

Bamboo can parse any test output that conforms to standard JUnit XML format. The implementation of this is pretty simple — Bamboo looks for specific tags in the JUnit XML output.

A failed JUnit XML report, that is successfully parsed by Bamboo.

```
<?xml version="1.0" encoding="UTF-8" ?>
<testsuite errors="0" tests="3" time="0.391" failures="1"
  name="com.atlassian.bamboo.repository.perforce.PerforceSyncCommandTest">
  <properties>
    <property value="Java(TM) 2 Runtime Environment, Standard Edition"
name="java.runtime.name"/>
    <property value="UnicodeBig" name="sun.io.unicode.encoding"/>
    .....
  </properties>
  <testcase time="0.001" name="testGeneratesCorrectP4CommandLine"/>
  <testcase time="0" name="testGettersReturnExpectedStuff"/>
  <testcase time="0.164" name="testUsingPerforceWhenNoFilesHaveChanged">
    <failure type="junit.framework.AssertionFailedError"
      message="Should not have any errors. [Perforce client error:,
Connect to server failed; ">
      junit.framework.AssertionFailedError: Should not have any errors.
[Perforce client error:, Connect to server
      failed; check $P4PORT., TCP connect to keg failed., keg: host unknown.]
expected:&lt;0&gt; but was:&lt;4&gt;
```

```
        at junit.framework.Assert.fail(Assert.java:47)
        at junit.framework.Assert.failNotEquals(Assert.java:282)
        at junit.framework.Assert.assertEquals(Assert.java:64)
        at junit.framework.Assert.assertEquals(Assert.java:201)
        at
com.atlassian.bamboo.repository.perforce.PerforceSyncCommandTest.testUsingPerforceWhenNoFilesHaveChanged(PerforceSyncCommandTest.java:60)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:585)
        at junit.framework.TestCase.runTest(TestCase.java:154)
        at junit.framework.TestCase.runBare(TestCase.java:127)
        at junit.framework.TestResult$1.protect(TestResult.java:106)
        at junit.framework.TestResult.runProtected(TestResult.java:124)
        at junit.framework.TestResult.run(TestResult.java:109)
        at junit.framework.TestCase.run(TestCase.java:118)
        at junit.framework.TestSuite.runTest(TestSuite.java:208)
        at junit.framework.TestSuite.run(TestSuite.java:203)
        at sun.reflect.GeneratedMethodAccessor17.invoke(Unknown Source)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:585)
        at
org.apache.maven.surefire.battery.JUnitBattery.executeJUnit(JUnitBattery.java:242)
        at
org.apache.maven.surefire.battery.JUnitBattery.execute(JUnitBattery.java:216)
        at org.apache.maven.surefire.Surefire.executeBattery(Surefire.java:215)
        at org.apache.maven.surefire.Surefire.run(Surefire.java:163)
        at org.apache.maven.surefire.Surefire.run(Surefire.java:87)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:585)
        at
org.apache.maven.surefire.SurefireBooster.runTestsInProcess(SurefireBooster.java:313)
        at
org.apache.maven.surefire.SurefireBooster.run(SurefireBooster.java:221)
        at
org.apache.maven.test.SurefirePlugin.execute(SurefirePlugin.java:371)
        at
org.apache.maven.plugin.DefaultPluginManager.executeMojo(DefaultPluginManager.java:412)
        at
org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeGoals(DefaultLifecycleExecutor.java:534)
        at
org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeGoalWithLifecycle(DefaultLifecycleExecutor.java:475)
        at
org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeGoal(DefaultLifecycleExecutor.java:454)
        at
org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeGoalAndHandleFailures(DefaultLifecycleExecutor.java:306)
        at
```

```
org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeTaskSegments(DefaultLife
cycleExecutor.java:273)
    at
org.apache.maven.lifecycle.DefaultLifecycleExecutor.execute(DefaultLifecycleExecuto
r.java:140)
    at org.apache.maven.DefaultMaven.doExecute(DefaultMaven.java:322)
    at org.apache.maven.DefaultMaven.execute(DefaultMaven.java:115)
    at org.apache.maven.cli.MavenCli.main(MavenCli.java:256)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:2
5)
    at java.lang.reflect.Method.invoke(Method.java:585)
    at org.codehaus.classworlds.Launcher.launchEnhanced(Launcher.java:315)
    at org.codehaus.classworlds.Launcher.launch(Launcher.java:255)
    at
org.codehaus.classworlds.Launcher.mainWithExitCode(Launcher.java:430)
    at org.codehaus.classworlds.Launcher.main(Launcher.java:375)
</failure>
<system-out>
    PerforceSyncCommand.command: /usr/local/bin/p4
</system-out>
```

```
</testcase>
</testsuite>
```

[Click here](#) to download the XML report.

A passed JUnit XML report, that is successfully parsed by Bamboo.

```
<?xml version="1.0" encoding="UTF-8" ?>
<testsuite errors="0" skipped="0" tests="1" time="0.045" failures="0"
name="com.atlassian.bamboo.labels.LabelManagerImplTest">
  <properties>
    <property value="Java(TM) 2 Runtime Environment, Standard Edition"
name="java.runtime.name"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/i386"
name="sun.boot.library.path"/>
    <property value="1.5.0_07-b03" name="java.vm.version"/>
    <property value="Sun Microsystems Inc." name="java.vm.vendor"/>
    <property value="http://java.sun.com/" name="java.vendor.url"/>
    <property value=":" name="path.separator"/>
    <property value="Java HotSpot(TM) Client VM" name="java.vm.name"/>
    <property value="sun.io" name="file.encoding.pkg"/>
    <property value="US" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="Java Virtual Machine Specification"
name="java.vm.specification.name"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN"
name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="sun.awt.X11GraphicsEnvironment" name="java.awt.graphicsenv"/>
    <property
value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core"
name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed"
name="java.endorsed.dirs"/>
    <property value="i386" name="os.arch"/>
    <property value="/tmp" name="java.io.tmpdir"/>
    <property value="Sun Microsystems Inc." name="java.vm.specification.vendor"/>
    <property value="Linux" name="os.name"/>
    <property value="/opt/java/tools/maven2/bin/m2.conf" name="classworlds.conf"/>
    <property value="ISO-8859-1" name="sun.jnu.encoding"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/i386.."
name="java.library.path"/>
    <property value="Java Platform API Specification"
name="java.specification.name"/>
    <property value="49.0" name="java.class.version"/>
    <property value="HotSpot Client Compiler" name="sun.management.compiler"/>
    <property value="2.6.15-1.1833_FC4smp" name="os.version"/>
    <property value="/home/bamboo" name="user.home"/>
    <property value="Australia/Sydney" name="user.timezone"/>
    <property value="sun.print.PSPrinterJob" name="java.awt.printerjob"/>
    <property value="ISO-8859-1" name="file.encoding"/>
    <property value="1.5" name="java.specification.version"/>
    <property value="bamboo" name="user.name"/>
    <property value="/opt/java/tools/maven2/boot/classworlds-1.1.jar"
name="java.class.path"/>
    <property value="1.0" name="java.vm.specification.version"/>
    <property value="32" name="sun.arch.data.model"/>
    <property value="/usr/java/jdk1.5.0_07/jre" name="java.home"/>
    <property value="Sun Microsystems Inc." name="java.specification.vendor"/>
    <property value="en" name="user.language"/>
    <property value="mixed mode, sharing" name="java.vm.info"/>
    <property value="1.5.0_07" name="java.version"/>
```

```
<property value="/usr/java/jdk1.5.0_07/jre/lib/ext" name="java.ext.dirs"/>
<property value="Sun Microsystems Inc." name="java.vendor"/>
<property value="/opt/java/tools/maven2" name="maven.home"/>
<property value="/home/bamboo/.m2/repository" name="localRepository"/>
<property value="/" name="file.separator"/>
<property value="http://java.sun.com/cgi-bin/bugreport.cgi"
name="java.vendor.url.bug"/>
<property value="little" name="sun.cpu.endian"/>
<property value="UnicodeLittle" name="sun.io.unicode.encoding"/>
<property value="" name="sun.cpu.isalist"/>
</properties>
```

```
<testcase time="0.045" name="testBAM1436"/>
</testsuite>
```

[Click here](#) to download the XML report.

[Click here for the AntXmlResultParser.java](#) file which contains the Bamboo code for parsing JUnit XML output.

For those interested in the XUnit XML Schema, please see [this document](#).

Known issues with CVS in Bamboo

Bamboo uses CVS `rlog` command - this lets you perform a CVS update on your local working directory without checking out your project.



CVS Error logging in Bamboo

Currently, if the server throws an error during a CVS build in Bamboo versions 2.0.x, the application will hang with no indication of any checkout/update problems. There is an [open JIRA issue](#) tracking this problem.

In order to further debug any CVS issues, you will need to turn up the CVS logging by passing in the `-DcvsClientLog=system` system argument to Bamboo.

1) Incompatibility with CVS servers 1.11.1 and below

Support for the `rlog` command 1.11.1p and performing a CVS `rlog` command returns the following error:

```
-cvs [rlog aborted]: server does not support rlog
```

2) Incompatibility with CVS server version 1.11.x when using "." to denote the root module to be checked out.

The CVS `rlog` command fails if you are using CVS version 1.11.x, with the following error.

```
INFO    | jvm 1    | 2008/05/15 14:19:10 | E cvs: recurse.c:642: do_recursion:
Assertion `strstr (repository, "/./") == ((void *)0)' failed.
INFO    | jvm 1    | 2008/05/15 14:19:10 | error
```

Please upgrade your CVS version to 1.12.x to get around this issue.

3) CVS Checkout format

Due to prior issues, Bamboo will checkout all files (including text files) from the CVS server as binary, however post Bamboo 2.1.2 this behaviour can be changed via a system parameter. To do this restart Bamboo with the following parameter (*if you have any [elastic agents](#) running, ensure that they are [shut down](#) before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become [orphaned from your Bamboo server](#)*).

```
-DCVS_CHECKOUT_BINARY_FORMAT=false
```

Post 2.1.5 this has been replaced with a more flexible option

```
-DCVS_CHECKOUT_FORMAT=BINARY
```

Option	Command Options	Behaviour
BINARY (Default)	-b	forces all files to be checked out in binary and won't convert any line endings
TEXT	-kv	forces all files to be checked out as text and converts all line endings (even Binary files)
NONE		lets CVS decide whether or not to convert line endings

For further reference, on configuring Bamboo start-up options [see this document](#)

Monitoring and Profiling Bamboo

This page helps you to set up profiling for bamboo. Profiling information can be useful to monitor bamboo's performance, memory consumption and the server's CPU load

On this page

- [Profiling with JMX](#)
 - [Enabling the profiler](#)
 - [Monitoring and Controlling the Profiler with JConsole](#)
- [Profiling with Yourkit](#)
 - [Installing Yourkit](#)
 - [Enabling the profiler](#)

Profiling with JMX

Enabling the profiler

JMX is Sun's [native java platform monitor](#) . JConsole can be used to visualise the profiling data.

To enable JMX add `-Dcom.sun.management.jmxremote` (or `-Dcom.sun.management.jmxremote.port=<portNum>` for remote monitoring) as a command line argument when starting bamboo.

For Unix : add the parameter to the `RUN_CMD` line in `bamboo.sh`

For Windows : add the parameter to the `.\conf\wrapper.conf` file as `wrapper.java.additional.4=-D...`

Restart Bamboo.



Does not work when bamboo is run as a windows service

At the moment profiling via JMX only works when bamboo is run as a console application on windows. Running bamboo as a windows service with profiling is currently not supported.

Monitoring and Controlling the Profiler with JConsole

Please refer to [Sun's documentation](#)

Profiling with Yourkit

Installing Yourkit

[Yourkit](#) is an alternative (commercial) java profiler.

Follow the [installation instructions](#) for your platform.

You do not need a license if you just run the profiling agent with bamboo.

Enabling the profiler

Add `-agentlib:yjpagent=onexit=snapshot` to the command line in `bamboo.sh` or as an additional parameter to `wrapper.conf`

Then follow the instructions on [enabling the profiler manually](#) for your platform.

Restart Bamboo.

Monitor Memory usage and Garbage Collection in Bamboo

A simple way to do this is to turn on garbage collection and heap dump on out of memory.

Parameters

Please [add the following parameters](#) to Bamboo

```
-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -verbose:gc -Xloggc:/path/to/gc.log
```

Note: Remember to substitute `/path/to/gc.log` with a meaningful file path on your server.

GC log file location

The garbage collection traces and the heap dumps are in `<bamboo-install>/gc.log`.

Additional Note

The `-XX:+PrintGCTimeStamps` flag, prints when GCs happen relative to the start of the application.

Some helpful links:

http://blogs.atlassian.com/developer/2007/10/plugging_leaks_in_confluence.html

<http://www.oracle.com/technetwork/articles/javase/gcportal-136937.html>

Moving Bamboo-Home of an agent

To move an agent's Bamboo-Home -

1. Move the Bamboo-Home of the agent, to the intended location.
2. Edit the <Bamboo-Agent-Home>/bamboo-agent.cfg.xml file, find the following line -

```
<buildWorkingDirectory>/YOUR_OLD_DIRECTORY/xml-data/build-dir</buildWorkingDirectory>
<buildArtifactDirectory>/YOUR_OLD_DIRECTORY/xml-data/artifacts</buildArtifactDirectory>
```

3. Point the working directory and the artifact directory to the new Bamboo-Home.
4. Start your Agent with -Dbamboo.home=your_new_agent_home and point to your new Bamboo-Agent-Home.

Performing a thread dump.

If Bamboo stops responding, or is performing poorly, you should create a thread dump to help Atlassian determine the cause of the problem.

This will show the state of each thread in the JVM, including a stack trace and information about what locks that thread is holding and waiting for.

Linux (and Solaris and other Unixes) Users

Find the process id of the JVM and issue the command:

Use the **ps** command to get list of all processes.

```
kill -3 <pid>
```

Note: This will not kill your server (so long as you included the "-3" option, no space in between). The thread dump will be printed to Bamboo's standard output.



Please note that some application servers (like tomcat) redirect stdout (to catalina.out for instance).

Jstack (any Platform with an JAVA JDK)

Sun JDK 1.5 and above ship with native tool called jstack to perform thread dump. To use the tool find the Process ID and execute the command:

```
jstack <ProcessID>
```



If you run your Atlassian product via wrapper (as a service) on Windows, you may encounter this error, 'Not enough storage is available to process this command'. See the suggestions in [this KB article](#) for workarounds.

Java VisualVM (any Platform with an JAVA JDK)

Oracle JDK has a native tool jvisualvm to perform thread dumps (and much more). To use the tool execute the command:

```
jvisualvm
```

Find Bamboo process ({{com.atlassian.bamboo.server.Server}}) and execute "Thread Dump" option available from a context menu.

Thread Dump Tools

- [Samurai](#)
- [Thread Dump Analyzer TDA](#)

Removing Coverage plug-in data from the Bamboo database

The third-party [Coverage plug-in](#) for Bamboo stores very large amounts of data in the Bamboo database. There are two consequences of this:

1. Using the Coverage plug-in with an embedded Bamboo database may result in poor performance, OutOfMemoryErrors, and/or Bamboo start-up failures; and
2. After installing the Coverage plug-in, you may encounter difficulties with Bamboo's import, export and backup features, such as OutOfMemoryErrors and corrupted export and backup files.

Precautionary Measures

To mitigate the risk of these problems, Atlassian makes the following recommendations to users of the Coverage plug-in:

1. Atlassian strongly recommends that you [migrate to a supported external database](#) before installing this plug-in; and
2. Once you have installed the Coverage plug-in, Atlassian strongly recommends that you regularly [backup your bamboo-home and external database using external tools](#), as the plug-in may interfere with the reliability of Bamboo's built-in backup feature. We intend to address the [underlying issue](#) in a future release of Bamboo.

Recovery Procedure

If an instance of Bamboo is configured with an embedded database and the Coverage plug-in is failing for the reasons described above, this can be rectified by removing the Coverage plug-in's data from the database, using the following procedure.

On Linux, Mac OS X and other Unix-like platforms:

1. Shut down Bamboo.
2. Execute the following commands in a shell, substituting *bamboo-home* with the path to your Bamboo home directory:

```
cd bamboo-home/database

grep "^INSERT INTO BUILDRESULTSUMMARY_CUSTOMDATA .*'coverage\'." defaultdb.script | gzip
> coverage.sql.gz

mv defaultdb.script defaultdb.script.backup_with_coverage

gzip defaultdb.script.backup_with_coverage

gunzip -c defaultdb.script.backup_with_coverage.gz | grep -v "^INSERT INTO
BUILDRESULTSUMMARY_CUSTOMDATA .*'coverage\'." > defaultdb.script
```

3. Start Bamboo.

On Microsoft Windows:

For assistance, please [raise a Bamboo support request](#).

Restoring passwords to recover admin users

Use this document if you are unable to login as administrator or have forgotten your password and do not have Mail Server configured, to manually replace administrator passwords.

Follow the instructions for either the [Embedded Database](#) or [External Database](#). If you have not configured a database, use the Embedded instructions.

Embedded Database Instructions

Stage One - Identify Administrator

This guide assumes that the first user added was an administrator. If this is not the case, search for the admin username and find their user id number, then modify their password hash instead.

1. Shutdown Bamboo
2. In your Bamboo home directory, open `\database\defaultdb.script` file in a text editor
3. Search for the text:

```
INSERT INTO USERS VALUES(1
```

To find the administrator login entry:

```
INSERT INTO USERS VALUES(1, 'USERNAME', 'PASSWORD_HASH')
```

Where the 1 is the user id number, and USERNAME and PASSWORD_HASH are actual values.
As an example, my table entry for user admin with password admin looks like this;

```
INSERT INTO USERS  
VALUES(1, 'admin', 'x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPwx10Nxp3Y3pB37A==', 'admin@admin.com', '2007-08-14 11:26:18.504000000', 'admin')
```

1. This step makes admin the administrator's password. Bamboo does not store passwords in plain text in the database, but uses hashes computed from the original password. The hash for the characters admin is below:

```
x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPwx10Nxp3Y3pB37A==
```

Paste the admin password hash between the " characters of their existing PASSWORD_HASH. The new administrator login entry should look like:

```
INSERT INTO USERS  
VALUES(1, 'USERNAME', 'x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPwx10Nxp3Y3pB37A==', 'EMAIL', 'DATE_TIME', 'FULL_NAME')
```

Where USERNAME is the administrator username.

2. Save the file
3. Start up Bamboo
4. Login with the administrator username and password admin

External Database Instructions

Stage One - Identify User

The first user added is always an admin. To restore your password you simply need to update the password hash in the USERS table with the admin hash

Connect to your database using a database admin tool such as [DBVisualiser](#). Please download a database admin tool now if you do not have one installed already. Once installed, connect to your database and retrieve the list of administrator usernames with:

```
select * from USERS where ID=1
```

This command should list all users who belong to Bamboo-Admin user group.

Stage Two - Replace Administrator Password

Bamboo does not store passwords in plain text in the database, but uses hashes computed from the original password. You instead cut and paste a hash, rather than the plain password, over the existing password. Below is the hash for the password admin

```
x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPwx10Nxp3Y3pB37A==
```

To change the password to admin for a given username:

1. Shutdown Bamboo
2. Connect to your database. run this SQL on your database: select * from USERS where NAME='admin' If you are using LDAP integration for user management (not only authentication) then your admin user will be in a different table. The SQL to run is:

```
update USERS set PASSWORD =  
'x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPwx10Nx  
p3Y3pB37A==' where NAME = 'USER_NAME_FROM_STAGE_ONE'
```

3. Start Bamboo
4. Login with your username and your password is now admin

Bamboo Password Hashing

- Before Bamboo 4.3.x the pure hashing method was used for which the hash of 'admin' is:

```
x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPwx10Nx  
p3Y3pB37A==
```

- From Bamboo 4.3.x the salted hash is used which can be recognized by the PKCS. The hash for 'admin' using the new method is:

```
{PKCS5S2}Bpkvn//DVjxjm/gSwKsLjDIt06e2Bfj9RHA5Z8mfMhMVGJkl9itnpzKST+BcHeFf
```

- However, the old hash will still work for all Bamboo versions when manually inserted into the DB.

Send Errors to stderr - Script Builder in Visual Studio WinXP to build Solutions Files

To display an Error Summary for erroneous builds in bamboo build summary is not available for the Script Builder - going through the build logs seems tedious.

There is a section named "Error summary" which collects all errors during the build process that are printed to **stderr**. For example a build script

```
#!/bin/bash  
echo "ERROR build xyz failed" >&2
```

would print this message into the build summary section. It is up to you to insert the appropriate messages into your build script.

Problem

The actual problem is devenv.com/msbuild not being very helpful: both build tools only append to **stdout** stream, even in the case of warnings/errors during the build.

Solution

I solved the issue by writing a simple Ruby script that invokes the build tool and filters the stdout stream for any warnings and errors via regexp; the matching warning/error lines are then echoed to **stderr** and Bamboo picks them up nicely.

build script.ry

```
pipe = IO.popen("devenv.com #{${*[0]}} /Rebuild ")
errors = 0
warnings = 0
while line = pipe.gets
if line =~ /^.* : .* error .*/
$stderr.puts line
errors += 1
elsif line =~ /^.* : warning .*/
$stderr.puts line
warnings += 1
else
$stdout.puts line
end
end
exit errors > 0 ? 1 : 0
```

Related Pages

Knowledge Base - (BSP-1381) Script Builder Display build errors in Error Summary

Using Bamboo with Clover

Getting Started

One-click Clover Integration

Clover has been seamlessly integrated with Bamboo from Bamboo 2.4 and later. Clover reports can be activated in the [Builder configuration screen](#). Please see [Enabling the Clover add-on # Automatic Clover integration](#) for further details.

Would you like to view Clover Code Coverage for this plan?

Integration Options

☒ Use Clover to collect Code Coverage for this build.
Clover is a code coverage tool reports how well tested your code is and also highlights parts of code that require more testing. For more information visit: [Atlassian Clover](#)

☒ Automatically integrate Clover into this build.

☐ Clover is already integrated into this build and a clover.xml file will be produced. For more information visit: [Atlassian Clover](#) or view the [Online Documentation](#).

☒ Generate a Clover Historical Report
Include coverage trends and class movers in the Clover HTML report. [More info](#).
(Will only work consistently if this plan is run on a single agent and no clean checkout is performed.)

☒ Generate a JSON report
JSON makes it very easy to integrate Clover data into a web-page. [Learn how](#).

Global Clover license has been configured in administration panel. To override use option below.

☐ Use plan-defined Clover license key
Override globally defined Clover license and provide dedicated license for this plan.

To configure Clover activity refer to [Clover Reference Guides](#) for your builder:

- [Clover for Ant](#)
- [Clover for Maven 2](#)

Classic Clover Integration

To use Clover with Bamboo, you need to:

1. Integrate Clover with your build and ensure that HTML and XML reports are generated:
 - [Clover-for-Ant Installation Guide](#)
 - [Clover-for-Maven 2 and 3 Installation Guide](#)
2. Ensure that there are tests present in your [build plan](#) that generate test results in JUnit test report format.
3. Configure where Bamboo can find Clover reports:
 - see [Enabling the Clover add-on # Manual Clover integration](#)

For further details, please see [Configuring tasks](#).

Common Problems

Q: I have managed to get Clover statistics displayed in numerical form for each build, but the graphs do not show a history of these statistics?

A: The history of Clover is displayed over time periods (e.g. a day, a week, a month), and the minimum data point is per day. The Clover coverage will not display data that is less than a day old.

Q: Will the Bamboo/Clover integration run on failed builds?

A: Before Bamboo version 1.2.1, Bamboo would only report Clover coverage for successful builds. As of Bamboo 1.2.1, Bamboo will report Clover coverage regardless of the build outcome.

Getting gcov results in Clover coverage summary



This feature is not officially supported by Atlassian. It is being maintained by open source community, feel free to contribute.

Description

Clover does not support code coverage for C/C++. However, it is possible to display C/C++ coverage statistics on "Clover" tab on "Job Summary" and "Plan Summary" pages. In order to get this working:

- create a task in which gcov is used and produces coverage file
- create a task in which python script (see references below) converts gcov data to clover.xml file
- enable Clover on Miscellaneous tab on Job Configuration page
 - enable "Use Clover to collect code coverage for this build"
 - select option "Clover is already integrated into this build and a clover.xml will be produced."
 - enter path to clover.xml file

References

Source code for Python script performing conversion is kept in Mercurial [bamboo-gcov-plugin](https://bitbucket.org/atlassian/bamboo-gcov-plugin) repository on bitbucket.org:

```
hg clone ssh://hg@bitbucket.org/atlassian/bamboo-gcov-plugin
```

Discussion about Clover schema on Atlassian Answers:

- <https://answers.atlassian.com/questions/68875/clover-xml-schema>

Working with Sun JAVA libraries

Due to licensing restrictions, we are not allowed to re-distribute native SUN libraries through our maven2 public repositories.

If you are developing plugins for Bamboo or building Bamboo from source, you might need `javax.mail` and `javax.transaction:jta:j`ar for Bamboo to build successfully. The relevant POMs for this look something like this:

```
.....
<dependency>
  <groupId>javax.mail</groupId>
  <artifactId>mail</artifactId>
  <version>1.3.2</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>jta</groupId>
  <artifactId>jta</artifactId>
  <version>1.0.1</version>
  <scope>compile</scope>
</dependency>
.....
```

Before building, please install the Sun JAR's into your local Maven2 repositories by following the instructions below.

To install the `javax.mail` JAR into your local Maven2 repository:

1. Download the `javax.mail` Jar from [Sun's website](https://javaee.github.io/javamail/).
2. Install it on our your local machine by entering the following command in a terminal:

```
mvn install:install-file -DgroupId=javax.mail -DartifactId=mail
-Dversion=1.3.3 -Dpackaging=jar -Dfile=YOUR/PATH/TO/FILE
```

To install `javax.transaction:jta:jar` JAR into your local Maven2 repository:

1. Download the `javax.transaction:jta:jar` Jar from [Sun's website](#).
2. Install it on your local machine by entering the following command in a terminal:

```
mvn install:install-file -DgroupId=javax.transaction -DartifactId=jta
-Dversion=1.0.1B -Dpackaging=jar -Dfile=/path/to/file
```

Bamboo indicates that my Ant or Maven builds failed, even though they were successful



Please note this Bamboo functionality relates only to the Maven Task and Ant Task outputs.

If your plan's build logs indicate that your Maven or Ant builds are passing but Bamboo is reporting them as failed (or vice-versa), it could be that:

- Bamboo is not finding 'BUILD SUCCESS' in your build logs
- Bamboo is finding 'BUILD FAILED' in your build logs when it should not be doing so. (This marker is not used in Maven.)
- Your builds are returning a non-zero return code. (For example, the build log will indicate Build process for 'ABC Application - XYZ Build' returned with return code = 1.)

If your builds produce atypical or non-standard output, you can make Bamboo check for text other than 'BUILD SUCCESS' or 'BUILD FAILED' in your build logs. An additional system property is available to specify how far back in the logs Bamboo checks for these text markers.

System Property	Description	Default Value
<code>atlassian.bamboo.builder.successMarker</code>	Specifies the text (or string) that Bamboo looks for in the build log to determine if the build was successful	BUILD SUCCESS
<code>atlassian.bamboo.builder.failedMarker</code>	Specifies the text (or string) that Bamboo looks for in the build log to determine if the build failed	BUILD FAILED
<code>SUCCESS_MESSAGE_LINES</code>	Specifies the number of lines from the end of the builder log in which to check for the values of <code>atlassian.bamboo.builder.successMarker</code> or <code>atlassian.bamboo.builder.failedMarker</code> .	250

For instructions on how to configure a system property, please refer to the [Configuring system properties](#) page.

How can I pass bamboo variables to my build script

[Bamboo global and build specific variables](#) can be referred to in build scripts or maven pom.xml. Bamboo variables are not directly available in the builder execution context however.

They can be passed as parameters to the builder.

Maven

For example, you may want your Maven 2 version to be determined by Bamboo. In Maven 2 pom.xml you may have:

```
...
<groupId>com.atlassian.boo</groupId>
<artifactId>boo-test</artifactId>
<packaging>jar</packaging>
<version>1.1.${bambooBuildNumber}-SNAPSHOT</version>
...
```


You can then specify the following in the '**Goal**' field of your build plan:

```
clean package -DbambooBuildNumber=${bamboo.buildNumber}
```

When the command runs, Bamboo will replace the `buildNumber` with the actual number (e.g. 1102), which will be passed to the underlying Maven build to use. The command will then produce a jar that looks like this: `boo-test-1.1.1102-SNAPSHOT.jar`.

Ant

You can pass bamboo variables as ant parameters along with ant targets like

```
clean test -Dbuild.key=${bamboo.buildKey}
```

In your ant build script just refer to this variable

```
...
<echo message="bamboo.buildKey = ${build.key}"/>
...
```

Contributing to the Bamboo documentation

Would you like to share your Bamboo hints, tips and techniques with us and with other Bamboo users? We welcome your contributions.

Blogging your technical tips and guides

Have you written a blog post describing a specific configuration of Bamboo or a neat trick that you have discovered? Let us know, and we will link to your blog from our documentation. [More....](#)

Contributing documentation in other languages

Have you written a guide to Bamboo in a language other than English, or translated one of our guides? Let us know, and we will link to your guide from our documentation. [More....](#)

On this page:

- [Blogging your technical tips and guides](#)
- [Contributing documentation in other languages](#)
- [Updating the documentation Itself](#)
 - [Getting permission to update the documentation](#)
 - [Our style guide](#)
 - [How we manage community updates](#)

Related pages:

- [Tips of the Trade](#)
- [Author Guidelines](#)
- [Atlassian Contributor License Agreement](#)

Updating the documentation Itself

Have you found a mistake in the documentation, or do you have a small addition that would be so easy to add yourself rather than asking us to do it? You can update the documentation page directly

Getting permission to update the documentation

Please submit the [Atlassian Contributor License Agreement](#).

Our style guide

Please read our short [guidelines for authors](#).

How we manage community updates

Here is a quick guide to how we manage community contributions to our documentation and the copyright that applies to the documentation:

- **Monitoring by technical writers.** The Atlassian technical writers monitor the updates to the documentation spaces, using RSS feeds and watching the spaces. If someone makes an update that needs some attention from us, we will make the necessary

changes.

- **Wiki permissions.** We use wiki permissions to determine who can edit the documentation spaces. We ask people to sign the [Atlassian Contributor License Agreement \(ACLA\)](#) and submit it to us. That allows us to verify that the applicant is a real person. Then we give them permission to update the documentation.
- **Copyright.** The Atlassian documentation is published under a Creative Commons CC BY license. Specifically, we use a [Creative Commons Attribution 2.5 Australia License](#). This means that anyone can copy, distribute and adapt our documentation provided they acknowledge the source of the documentation. The CC BY license is shown in the footer of every page, so that anyone who contributes to our documentation knows that their contribution falls under the same copyright.

Tips of the Trade

Below are some links to external blog posts and articles containing technical tips and instructions on setting up and using Bamboo. This page presents an opportunity for customers and community authors to share information and experiences.

The references here are technical 'how to' guides written by bloggers who use Bamboo. For feature tours, solution tours and other information about continuous integration, please refer to the [Atlassian website](#) and to our [evaluator resources](#).



Please be aware that these are external blogs and articles.

Most of the links point to external sites, and some of the information is relevant to a specific release of Bamboo. Atlassian provides these links because the information is useful and relevant at the time it was written. Please check carefully whether the information is still relevant when you read it, and whether it is relevant to your version of Bamboo. [Unless explicitly stated](#), Atlassian does not offer support for third-party extensions or plugins. The information in the linked blog posts has not been tested or reviewed by Atlassian. We recommend that you test all solutions on a **test** server before trying them on your production site.

On this page:

- [Lightning fast notification](#)
- [Automating the staging and production deployments](#)
- [Continuous Integration for Ruby](#)
- [Continuous Integration Goodness for your Ruby Project](#)
- [Bamboo JMeter Aggregator - Getting the most from performance builds](#)
- [Bamboo plugins for Git and GitHub](#)
- [Secure Installation of Bamboo](#)

Build Management

[Lightning fast notification](#)

- By: John Ferguson Smart, on the 'Atlassian Blog'
- About: Using IM as a notification system to keep developers up to date on the new deployments for their various projects
- Date: 15 April 2009
- Related documentation: [Working with Instant Messenger \(IM\) notifications](#)

[Automating the staging and production deployments](#)

- By: John Ferguson Smart, on the 'Atlassian Blog'
- About: Moving your builds to general availability (GA) and production deployments with Maven, JIRA and Bamboo
- Date: 6 May 2009
- Related documentation: [JiraVersions Plugin](#)

Non-Java Languages

[Continuous Integration for Ruby](#)

- By: John Ferguson Smart, on the 'Atlassian Blog'
- About: A Continuous Integration environment that runs Ruby builds and tests on Bamboo, and automates the deployment and installation on a remote test machine
- Date: 20 May 2009
- Related documentation: [Can Bamboo build and test non-Java projects](#)

[Continuous Integration Goodness for your Ruby Project](#)

- By: Nick Sieger, on the 'Nick Sieger' blog
- About: Running Ruby builds and tests on Bamboo
- Date: 6 Jan 2007
- Related documentation: [Can Bamboo build and test non-Java projects](#)

Performance Builds

[Bamboo JMeter Aggregator - Getting the most from performance builds](#)

- By: James Roper, on the 'Atlassian Blog'
- About: Using the Bamboo JMeter Aggregator plugin to manage the data produced by your performance builds
- Date: 21 May 2009
- Related documentation: [Bamboo JMeter Aggregator Plugin](#)

Repositories

Bamboo plugins for Git and GitHub

- By: Ken Olofsen, on the 'Atlassian Blog'
- About: Using Bamboo with Git and Github
- Date: 2 May 2009
- Related documentation: [Specifying the source repository](#)

Installation

Secure Installation of Bamboo

- By: Stéphane Bagnier, on the 'Antelink Blog'
- About: Part of a series about the complete installation of the Atlassian suite behind a proxy with SSL everywhere
- Date: 14 December 2010
- Related documentation: [Bamboo installation guide](#)



Have you written a technical tip for Bamboo?

Add a comment to this page, linking to your blog post or article. We will include it if the content fits the requirements of this page.



Feedback?

Your first port of call should be the author of the linked blog post. If you want to let us know how useful (or otherwise) a linked post is, please add a comment to this page.

Other Sources of Information

[Atlassian website](#)
[Atlassian forums](#)
[Atlassian blog](#)
[Bamboo plugins](#)

Bamboo Documentation in Other Languages

Below are some links to Bamboo documentation written in other languages. In some cases, the documentation may be a translation of the English documentation. In other cases, the documentation is an alternative guide written from scratch in another language. This page presents an opportunity for customers and community authors to share documentation that they have written in other languages.



Please be aware that these are external guides.

Most of the links point to external sites, and some of the information is relevant to a specific release of Bamboo. Atlassian provides these links because the information is useful and relevant at the time it was written. Please check carefully whether the information is still relevant when you read it, and whether it is relevant to your version of Bamboo. The information in the linked guides has not been tested or reviewed by Atlassian.

On this page:

- No guides yet

None

No guides yet

We do not yet have any guides to link here. Be the first to suggest one!

Adding Your Own Guide to this Page

Have you written a guide for Bamboo in another language? Add a comment to this page, linking to your guide. We will include it if the content fits the requirements of this page.

Giving Feedback about One of the Guides

If you have feedback on one of the guides listed above, please give the feedback to the author of the linked guide.

If you want to let us know how useful (or otherwise) one of these guides is, please add a comment to this page.

Other Sources of Information

[Bamboo Documentation Home](#)
[Atlassian website](#)
[Atlassian blog](#)
[Bamboo plugins](#)

Bamboo resources

Resources for Evaluators

- [Free Trial](#)
- [Feature Tour](#)

Resources for Administrators

- [Bamboo forum at Atlassian Answers](#)
- [Bamboo Knowledge Base](#)
- [Bamboo FAQ](#)
- [Tips of the Trade](#)
- [Guide to Installing an Atlassian Integrated Suite](#)
- [The big list of Atlassian gadgets](#)

Resources for Developers

- [Bamboo Developer Documentation](#)
- [API documentation](#)
- [Developer topics on Atlassian Answers](#)

Downloadable Documentation

- [Bamboo documentation in PDF, HTML or XML formats](#)

Plugins

- [Atlassian Marketplace](#)

IDE Connectors

- Use the [Atlassian Connector for Eclipse](#) or the [Atlassian Connector for IntelliJ IDEA](#) to work with your Bamboo builds right there in your development environment. Do you use [JIRA](#), [Crucible](#) or [FishEye](#) too? With the connector you can manage your issues and code reviews within your IDE, or move quickly between the IDE and a FishEye view of your source repository. **Hint:** The Atlassian IDE Connectors are free.

Support

- [Atlassian Support](#)
- [Support Policies](#)

Training

- [Atlassian Training](#)

Forums

- [Bamboo forum at Atlassian Answers](#)
- [Bamboo developers forum](#)

Mailing Lists

- Visit <http://my.atlassian.com> to sign up for mailing lists relating to Atlassian products, such as technical alerts, product announcements and developer updates.

Feature Requests

- [Issue Tracker and Feature Requests for Bamboo](#)

Glossary

activity log

agent

agent-specific capability

artifact

authors in Bamboo

build

build activity

build duration

build log

build queue

build result

build telemetry

capability

child

committer

custom capability

default repository

elastic agent

elastic Bamboo

elastic block store

elastic image

elastic instance

executable

favourites

global permission

job

label

local agent

parent

permission

plan

plan permission

projects in Bamboo

queue

reason

remote agent

remote agent supervisor

requirement

shared capability

stage

stock images

task

triggering

watcher

activity log

Every [plan](#) has an *activity log*. An *activity log* is a temporary display of the latest output from the plan's most recent [build log](#).

agent

A Bamboo *agent* is a service that can run [job builds](#). There are two types of Bamboo agents:

- *local agents* run as part of the Bamboo server.
- *remote agents* run on computers, other than the Bamboo server, that run the [remote agent](#) tool. An [elastic agent](#) is a remote agent that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#).

 Local agents run in the Bamboo server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.

Each agent has a defined set of [capabilities](#) and can only run builds for jobs whose [requirements](#) match the agent's capabilities.

agent-specific capability

An *agent-specific capability* is a [capability](#) that applies to one agent only. Note that the value of an agent-specific capability will override the value of a *shared capability* of the same name (if one exists).

See [Agents and capabilities](#) and [Configuring capabilities](#) for more information.

artifact

Artifacts are files created by a [job build](#) (e.g. JAR files). Artifact definitions are used to specify which artifacts to keep from a build and are configured for individual [jobs](#).

See [Configuring artifact sharing between jobs](#).

authors in Bamboo

An *author* is any person who checks in code to a repository that is associated with a Bamboo [plan](#). An author need not be a Bamboo user.

See [Generating reports on selected authors](#).

build

A build is the execution of either a [plan](#) or a [job](#). The execution of a plan is referred to as a 'plan build' and that of a job is a 'job build'.

build activity

Build activity is the number of builds that occur in a given period of time.

build duration

Build duration is the total time taken to execute a [plan](#) — that is, the time taken to compile the code and run all of the plan's tests.

Variations in a plan's build *duration* can be over time.

build log

Every [build](#) has a *build log*. A *build log* is a permanent record of all the output generated by compiling the [job's](#) source-code and executing

the tests.

build queue

The Bamboo *build queue* controls the sequence of *builds*. When a plan submits a build to the build queue, the build will wait in the build queue until a suitable [agent](#) is available to run the build.

The build queue is displayed on the **Current Activity** tab of the [Dashboard](#).

build result

Every completed build has a *build result*:

- 'Successful' — the code compiled, with or without errors, and all tests completed successfully.
- 'Failed' — either the code did not compile, or at least one test failed.
- 'Incomplete' — the build was not completed, e.g. it may have been stopped manually.

Additionally,

- if the build result is 'Failed', and the previous build result was 'Successful', the build is said to be 'Broken'.
- if the build result is 'Successful', and the previous build result was 'Failed', the build is said to be 'Fixed'.

build telemetry

Build telemetry is the insight provided by Bamboo's dynamic reports, charts and collation of build metrics. Build telemetry helps identify trends across [build plans](#) and across [authors](#) — not just focusing on the results of a single build.

capability

A *capability* is a feature of an [agent](#). A capability can be defined on an agent for:

- an executable (e.g. Maven)
- a JDK
- a Version Control System client application (e.g. Git)
- a custom capability. This is a key-value property which defines a particular characteristic of an agent (e.g. 'operating.system=WindowsXP' or 'fast.builds=true').

Capabilities typically define the path to an executable that has already been installed, and must be defined in Bamboo before Bamboo or its agents can make use of those.

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

See [Configuring capabilities](#) for more information.

child

A *child* is a [plan](#) which gets triggered when another plan completes a build. See [Setting up plan build dependencies](#).

committer

A *committer* is the Bamboo user(s) who committed code to a particular build (i.e. someone who committed code after the previous build was checked out by Bamboo).

Administrators can configure a plan's [notifications](#) to be sent to the build's committer(s).

custom capability

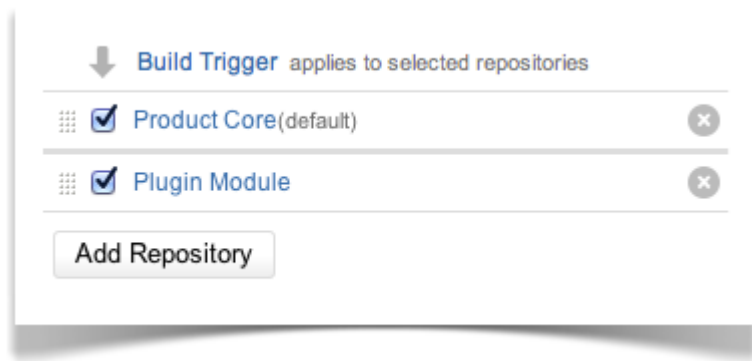
Custom capabilities can be used to control which [jobs](#) will be built by a particular agent, since agent [capabilities](#) are required to match job [requirements](#). For example, if the builds for a particular job should only run in a Windows environment, you could create a custom capability 'operating.system=WindowsXP' for the appropriate agent(s), and specify it as a requirement for this job.

- To create a new custom capability in your Bamboo system, see [Defining a new custom capability](#).
- To specify a job's requirement for a custom capability, see [Configuring a job's requirements](#).

default repository

The first repository in the list of plan repositories is the [Plan's Default Repository](#). The default repository will be automatically checked out by any new [job](#) created.

Repository specific [Plan Variables](#), such as *repository.revision.number*, will point to the default repository of a Plan. To address a specific repository, you must add the name of the repository to the end of the variable as follows: *repository.revision.number.product_core*.



elastic agent

An *elastic agent* is a [remote agent](#) that runs in the [Amazon Elastic Compute Cloud \(EC2\)](#). An elastic agent process runs in an [elastic instance](#) of an [elastic image](#). An elastic agent inherits its capabilities from the [elastic image](#) that it was created from.

elastic Bamboo

Elastic Bamboo allows you to use computing resources from the [Amazon Elastic Compute Cloud \(EC2\)](#) to run builds. Elastic Bamboo uses a [remote agent AMI \(Amazon Machine Image\)](#) to create [instances of remote agents](#) in the Amazon EC2.

elastic block store

The Amazon [Elastic Block Store \(EBS\)](#) provides 'EBS volumes' which can attach to EC2 instances. EBS volumes (and the 'EBS snapshots' created from these volumes) provide persistent storage for your [elastic instances](#).

If you have relatively static resources required for building your Bamboo jobs (such as, source code checkouts and Maven repository artifacts), you can add these to an EBS volume. From this volume, you can create an EBS snapshot, which effectively records the 'state' of an EBS volume at a given point in time.

elastic image

An *elastic image* is an Amazon Machine Image (AMI) that is stored in one of Amazon data centres for use with the [Elastic Bamboo feature](#). An elastic image is used to create [elastic instances](#), which in turn create [elastic agents](#). Conceptually, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system.

Each elastic image registered with the Amazon Web Services (AWS) has its own unique identifier, known as an **AMI ID**.

You can [associate multiple elastic images](#) with a Bamboo server. One [default shared image](#) is maintained by Atlassian in AWS, and is available to all Elastic Bamboo users.

You can also [create your own custom elastic images](#).

elastic instance

An *elastic instance* is a running instance of an [elastic image](#). One elastic instance is created whenever an elastic image is started. Hence, starting one elastic image multiple times, results in the creation of multiple elastic instances. Each time an elastic instance is created, one [elastic agent](#) is created on that instance.

Conceptually, an elastic instance can be thought of as a computer. The elastic agent's processes are run on this computer and the elastic image is the boot hard drive. Unlike computers, however, elastic instances are temporary and stateless. When an elastic instance is shut down:

- Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) will not persist
- Any customisations to the instance itself will also be lost.

✓ The Amazon [Elastic Block Store](#) can provide persistent storage for your elastic instances.

executable

An *executable* is an external program that Bamboo uses during the build process. Generally, executables compile source code to generate compiled executable files (referred to as [artifacts](#) in Bamboo). [Ant](#), [Maven](#), [MSBuild](#) or [PHPUnit](#) are just some examples of executables that can be used as part of your build process.

New executables can be defined as [capabilities](#) in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a [task](#).

See [Defining a new executable capability](#).

favourites

Each Bamboo user can nominate their *favourite* plans — that is, the plans they work with the most.

Each user's favourites are displayed on the 'My' page of the [Dashboard](#). Bamboo administrators can also configure each plan to send [build result notifications](#) to users who have nominated the plan as one of their favourites (these users are known as the plan's 'watchers').

global permission

A *global permission* is the ability to perform a particular operation in relation to Bamboo as a whole. See [Granting global permissions to users or groups](#).

See also [plan permission](#).

job

A Bamboo *job* is a single build unit within a [plan](#). One or more jobs can be organised into one or more [stages](#). The jobs in a stage can all be run at the same time, if enough Bamboo agents are available. A job is made up of one or more [tasks](#).

A job:

- Processes a series of one or more [tasks](#) that are run **sequentially** on the **same agent**.
- Controls the order in which tasks are performed.
- Collects the [requirements](#) of individual tasks in the [job](#), so that these requirements can be matched with agent [capabilities](#).
- Defines the [artifacts](#) that the build will produce.
- Can only use artifacts produced in a previous stage.
- Specifies any [labels](#) with which the [build result](#) or build artifacts will be tagged.

Each new plan created in Bamboo contains at least one job known as the 'Default Job'.

Projects and plans can only be configured by Bamboo administrators (see [Creating a plan](#)).

label

A *label* is a convenient way to tag and group [build results](#) that are logically related to each other. Labels can also be used to define [RSS feeds](#) and to control [build expiry](#).

Labels can be applied to build results automatically, by specifying the label(s) in a plan (note that only Bamboo administrators can do this). Labels can also be [applied ad hoc](#) to build results by Bamboo users.

local agent

See [agent](#).

parent

A *parent* is a [plan](#) which triggers another plan to build whenever it completes a build. See [Setting up plan build dependencies](#).

permission

See [plan permission](#) and [global permission](#).

plan

A *plan* defines everything about your continuous integration build process in Bamboo.

A plan:

- Has a single [stage](#), by default, but can be used to group [jobs](#) into multiple stages.
- Processes a series of one or more stages that are run **sequentially** using the **same** repository.
- Specifies the default repository.
- Specifies how the build is [triggered](#), and the triggering dependencies between the plan and other plans in the project.
- Specifies [notifications](#) of [build results](#).
- Specifies who has [permission](#) to view and configure the plan and its jobs.
- Provides for the definition of [plan variables](#).

Every plan belongs to a [project](#).

Projects and plans can only be configured by Bamboo administrators (see [Creating a plan](#)).

plan permission

A *plan permission* is the ability to perform a particular operation on a [plan](#) and its [jobs](#). For each plan, different permissions can be granted to particular groups and/or users.

See [Configuring a plan's permissions](#) and [Granting plan permissions in bulk](#).

See also [global permission](#).

projects in Bamboo

A *project* is a collection of *plans*. Projects enable you to easily group and identify plans which are logically related to each other. They are especially useful when generating [reports](#) across multiple plans.

A project:

- Has one, or more, [plans](#).
- Provides [reporting](#) (using the wallboard, for example) across all plans in the project.
- Provides links to other applications.

Projects are created from the create plan screen. Select New Project from the project drop down when [creating a new plan](#).

queue

See [build queue](#).

reason

A build's *reason* is the way in which the build was triggered.

Triggering in Bamboo allows plan builds to be started automatically. Bamboo has the following trigger methods:

- **Polling the repository for changes** — Bamboo polls the source repository for changes, either periodically or according to a schedule. This ensures that a plan build only runs when code has changed in the plan's source repository.
- **Repository triggers the build when changes are committed** — Requires that your source repository is configured to fire an event to Bamboo. This has the advantage of placing minimal load on your Bamboo server.
- **Cron-based scheduling** — Builds are run according to a schedule, regardless of whether any code changes have occurred. This can allow a team to structure the day according to a predictable schedule.
- **Single daily build** — The build is run at a specified time every day.

For more information, see [Triggering builds](#).

remote agent

See [agent](#) and the [Bamboo remote agent installation guide](#).

remote agent supervisor

A *remote agent supervisor* is an application that is installed alongside a Bamboo remote agent, by default. The remote agent supervisor is an implementation of the [Java Service Wrapper](#).

The remote agent supervisor monitors remote agents on the machine that it is installed on. If any remote agent crashes, the remote agent supervisor will automatically attempt to restart it. If communications are lost with the Bamboo server, the remote agent will shut itself down and wait for the remote agent supervisor to restart it.

The remote agent supervisor will run on the following operating systems:

- **Linux:**
 - x86
 - x86_64
 - IA64
 - PPC 64 bit (*but not 32 bit*)
- **Mac OSX:**
 - all architectures
- **Solaris:**
 - x86
 - x86_64 (running in 32 bit mode)
 - IA64 (running in 32 bit mode)
 - SPARC (both 32 bit and 64 bit)
- **Windows:**
 - 32 bit
 - 64 bit

requirement

A *requirement* is specified in a [job](#) or a [task](#). A requirement specifies a [capability](#) that an [agent](#) must have for it to [build](#) that job or task. A job inherits all of the requirements specified in its tasks.

Together, capabilities and requirements control which agents can execute builds for particular [jobs](#). Each job can only be built by agents whose capabilities match the job's requirements. See [Configuring a job's requirements](#) for more information.

shared capability

Shared capabilities are inherited by all applicable agents, that is, (shared) local server capabilities are inherited by all local agents, and shared remote capabilities are inherited by all remote agents. Note, however, that the value of a shared capability will be overridden by the value of an *agent-specific capability* of the same name (if one exists).

See [Agents and capabilities](#) and [Configuring capabilities](#).

stage

Stages group (or 'map') [jobs](#) to individual steps within a [plan's build process](#). For example, you may have an overall plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo stages to represent each of these steps.

A stage:

- Has a single [job](#), by default, but can be used to group multiple jobs.
- Processes its jobs in **parallel**, on **multiple agents** (where available).
- Must successfully complete all its jobs before the next [stage](#) in the [plan](#) can be processed.
- May produce [artifacts](#) that can be made available for use by a subsequent stage.

Each new plan created in Bamboo contains at least one stage (for the [default job](#)) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

stock images

Atlassian maintains public 'default' [elastic images](#), currently they are available for the following operating systems:

- Amazon Linux
- Windows (introduced in Bamboo 3.4)

Bamboo's [Elastic Bamboo](#) feature uses these images by default. In your list of [elastic image configurations](#), this image will have '**(stock image)**' appended to its name.

On this page:

- [Amazon Linux stock image](#)
- [Windows stock image](#)
- [Notes](#)

Amazon Linux stock image

The Amazon Linux 'default image' uses:

- the [Amazon Linux](#) (a CentOS derivative) operating system.
- the Bamboo elastic agent.*

and has the following default packages/capabilities:

Default packages/capabilities	Path/value
Builders	
Ant 1.8.2	/opt/ant-1.8.2
Grails 1.2.2	/opt/grails-1.2.5
Grails 1.3.1	/opt/grails-1.3.7
Grails 2.0.1	/opt/grails-2.0.1
Grails 2.1	/opt/grails-2.1
Maven 2.0 (Maven 2.x) (<i>version 2.0.11</i>)	/opt/maven-2.0.11
Maven 2.1 (Maven 2.x) (<i>version 2.1.0</i>)	/opt/maven-2.1.0
Maven 2.2 (Maven 2.x) (<i>version 2.2.1</i>)	/opt/maven-2.2.1
Maven 3.0 (Maven 3.x) (<i>version 3.0.4</i>)	/opt/maven-3.0.4
JDKs	
JDK 1.5 (<i>version 5u22</i>)	/opt/jdk-5 (<i>for Java Home</i>)
JDK 1.6 (<i>version 6u31</i>)	/opt/jdk-6 (<i>for Java Home</i>)
Oracle JDK 1.7 (<i>version 7u3</i>)	/opt/jdk-7 (<i>for Java Home</i>)
OpenJDK 1.7 (<i>version 7u2</i>)	/opt/openjdk-7 (<i>for Java Home</i>)
Others	
Bash (Command)	/bin/bash
Git	/usr/bin/git
Mercurial 1.4 Executable (<i>version 1.8.4</i>)	/usr/bin/hg
PHPUnit 3.4	/usr/bin/phpunit-3.4
PHPUnit 3.7	/usr/bin/phpunit-3.7

* The agent jar also contains the libraries required to connect to Subversion and CVS.

Windows stock image

The Windows 'stock image' is built from:

- the Windows 2008 Server R2 64bit operating system, with all updates applied.
- the Bamboo elastic agent.*

and has the following default packages/capabilities:

Default packages/capabilities	Path/value
Builders	
Ant 1.8.2	C:\opt\ant-1.8.2
Maven 2.0 (Maven 2.x) (<i>version 2.0.11</i>)	C:\opt\maven-2.0.11
Maven 2.1 (Maven 2.x) (<i>version 2.1.0</i>)	C:\opt\maven-2.1.0
Maven 2.2 (Maven 2.x) (<i>version 2.2.1</i>)	C:\opt\maven-2.2.1
Maven 3.0 (Maven 3.x) (<i>version 3.0.3</i>)	C:\opt\maven-3.0.3
MSBuild 2.0 (32bit)	C:\Windows\Microsoft.NET\Framework\v2.0.50727\MSBuild.exe
MSBuild 2.0 (64bit)	C:\Windows\Microsoft.NET\Framework64\v2.0.50727\MSBuild.exe
MSBuild 3.5 (32bit)	C:\Windows\Microsoft.NET\Framework\v3.5\MSBuild.exe
MSBuild 3.5 (64bit)	C:\Windows\Microsoft.NET\Framework64\v3.5\MSBuild.exe
MSBuild 4.0 (32bit)	C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe
MSBuild 4.0 (64bit)	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe
NAnt 0.91	C:\opt\nant-0.91
JDKs	
JDK 1.6 (<i>version 6u33</i>)	C:\Program Files\Java\jdk1.6.0_33
JDK 1.7 (<i>version 7u05</i>)	C:\Program Files\Java\jdk1.7.0_05
Browsers	
Firefox (<i>FF10</i>)	C:\Program Files (x86)\Mozilla Firefox
Internet Explorer (<i>IE9</i>)	C:\Program Files (x86)\Internet Explorer
Others	
Git	C:\Program Files (x86)\Git\bin\git.exe
Mercurial 2.0 Executable (<i>version 2.0</i>)	C:\Program Files\TortiseHg\hg.exe

* The agent jar also contains the libraries required to connect to Subversion and CVS.

Notes

 Be aware that the default packages/capabilities listed above may change with each major release of Bamboo. There is a new default image (with its own **AMI ID**) for each new version of Bamboo. However, older default images will still be available for use.

task

A task:

- Is a small discrete unit of work, such as source code checkout, executing a Maven goal, running a script, or parsing test results.
- Is run sequentially within a [job](#) on a Bamboo working directory.

Tasks may make use of an [executable](#) if required. Once a task is defined in the Bamboo system, it can then be specified in jobs by a plan administrator. A job can be configured to execute a number of tasks, on the same working directory. For example, before executing a Maven goal, the user could substitute specific files within the working directory, substitute version numbers, checkout source repositories or execute a script.

triggering

Triggering in Bamboo allows plan builds to be started automatically. Bamboo has the following trigger methods:

- **Polling the repository for changes** — Bamboo polls the source repository for changes, either periodically or according to a schedule. This ensures that a plan build only runs when code has changed in the plan's source repository.
- **Repository triggers the build when changes are committed** — Requires that your source repository is configured to fire an event to Bamboo. This has the advantage of placing minimal load on your Bamboo server.
- **Cron-based scheduling** — Builds are run according to a schedule, regardless of whether any code changes have occurred. This can allow a team to structure the day according to a predictable schedule.
- **Single daily build** — The build is run at a specified time every day.

For more information, see [Triggering builds](#).

watcher

A plan's *watchers* are the Bamboo users who have marked this plan as one of their [favourites](#). Administrators can configure a plan's [notifications](#) to be sent to the plan's watchers.