Contents

Getting started with Bamboo ................................................................. 9
Understanding the Bamboo CI Server .................................................... 9
Getting started with Java and Bamboo ................................................. 11
Getting started with .NET and Bamboo ............................................... 16
Getting started with PHP and Bamboo ................................................ 20
Getting started with Ruby and Bamboo ............................................... 20
Using the Bamboo dashboard .............................................................. 20
Using Bamboo ..................................................................................... 22
Configuring plans ................................................................................ 23
Creating a plan ..................................................................................... 25
Creating a new plan ............................................................................. 26
Cloning an existing plan ..................................................................... 27
Importing a Maven 2 project ............................................................... 28
Managing plans ................................................................................... 30
Editing a plan's configuration ............................................................... 30
Configuring a plan's permissions ......................................................... 31
Disabling or deleting a plan ................................................................ 32
Labelling a plan ................................................................................... 33
Defining plan variables ....................................................................... 34
Using stages in a plan ......................................................................... 35
Creating a stage ................................................................................... 36
Deleting a stage ................................................................................... 37
Editing a stage .................................................................................... 38
Rerunning a failed stage ..................................................................... 39
Using plan branches ........................................................................... 40
Connecting to code repositories ......................................................... 45
Specifying the source repository ......................................................... 46
Bitbucket - Mercurial ........................................................................... 48
CVS ....................................................................................................... 50
Configuring source code management triggers for CVS ................. 53
Git ......................................................................................................... 56
GitHub ................................................................................................ 59
Mercurial ............................................................................................. 61
Upgrading remote agents for Mercurial .............................................. 64
Perforce ............................................................................................... 65
Using Perforce with Bamboo - limitations and workarounds ..... 68
Subversion .......................................................................................... 70
Configuring source code management triggers for Subversion .... 73
Configuring a shared source repository .............................................. 75
Configuring jobs and tasks ................................................................. 75
Configuring jobs ................................................................................ 76
Creating a job ..................................................................................... 77
Editing a job ....................................................................................... 78
Configuring a job's requirements ...................................................... 79
Configuring a job's build artifacts ...................................................... 80
Configuring miscellaneous settings for a job ................................... 85
Configuring automatic labelling of job build results .................... 87
Viewing a job's Maven dependencies ............................................... 88
Disabling or deleting a job ................................................................. 89
Deleting a job's current working files .............................................. 90
Configuring tasks ............................................................................... 90
Checking out code ............................................................................. 93
Configuring a builder task ................................................................. 94
Ant ....................................................................................................... 95
Custom command executable .......................................................... 97
Grails .................................................................................................. 99
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working with builds</td>
<td>130</td>
</tr>
<tr>
<td>Polling the repository for changes</td>
<td>135</td>
</tr>
<tr>
<td>Repository triggers the build when changes are committed</td>
<td>137</td>
</tr>
<tr>
<td>Cron-based scheduling</td>
<td>139</td>
</tr>
<tr>
<td>Single daily build</td>
<td>140</td>
</tr>
<tr>
<td>Triggering a plan build manually</td>
<td>141</td>
</tr>
<tr>
<td>Triggering a plan build when code is updated</td>
<td>143</td>
</tr>
<tr>
<td>Triggering a plan build based on a schedule</td>
<td>144</td>
</tr>
<tr>
<td>Setting up plan build dependencies</td>
<td>144</td>
</tr>
<tr>
<td>Dependency blocking strategies</td>
<td>146</td>
</tr>
<tr>
<td>Working with build results</td>
<td>147</td>
</tr>
<tr>
<td>Viewing a build result</td>
<td>148</td>
</tr>
<tr>
<td>Deleting the results of a plan build</td>
<td>149</td>
</tr>
<tr>
<td>Working with comments</td>
<td>150</td>
</tr>
<tr>
<td>Commenting about a build result</td>
<td>150</td>
</tr>
<tr>
<td>Viewing comments about a build result</td>
<td>151</td>
</tr>
<tr>
<td>Viewing code check-in comments</td>
<td>153</td>
</tr>
<tr>
<td>Working with labels</td>
<td>153</td>
</tr>
<tr>
<td>Labelling a build result</td>
<td>154</td>
</tr>
<tr>
<td>Removing a label from a build result</td>
<td>154</td>
</tr>
<tr>
<td>Viewing labelled build results</td>
<td>154</td>
</tr>
<tr>
<td>Viewing popular labels</td>
<td>155</td>
</tr>
<tr>
<td>Viewing the code changes that triggered a build</td>
<td>155</td>
</tr>
<tr>
<td>Viewing a build's artifacts</td>
<td>156</td>
</tr>
<tr>
<td>Viewing a build log</td>
<td>157</td>
</tr>
<tr>
<td>Viewing the metadata for a build result</td>
<td>158</td>
</tr>
<tr>
<td>Viewing test statistics for a job</td>
<td>159</td>
</tr>
<tr>
<td>Configuring build results expiry for a plan</td>
<td>160</td>
</tr>
<tr>
<td>Viewing test results for a build</td>
<td>162</td>
</tr>
<tr>
<td>Viewing a test's history</td>
<td>163</td>
</tr>
<tr>
<td>Reordering jobs in the build queue</td>
<td>164</td>
</tr>
<tr>
<td>Stopping an active build</td>
<td>164</td>
</tr>
<tr>
<td>Quarantining failing tests</td>
<td>165</td>
</tr>
<tr>
<td>Getting feedback</td>
<td>166</td>
</tr>
<tr>
<td>Notifications</td>
<td>167</td>
</tr>
<tr>
<td>Displaying the wallboard</td>
<td>167</td>
</tr>
<tr>
<td>Configuring notifications for a plan and its jobs</td>
<td>169</td>
</tr>
<tr>
<td>Configuring Bamboo to send SMTP Email</td>
<td>173</td>
</tr>
<tr>
<td>Configuring Bamboo to use Instant Messaging</td>
<td>176</td>
</tr>
<tr>
<td>Working with Instant Messenger (IM) notifications</td>
<td>186</td>
</tr>
<tr>
<td>Subscribing to RSS feeds</td>
<td>187</td>
</tr>
<tr>
<td>Reporting</td>
<td>188</td>
</tr>
<tr>
<td>Viewing build statistics for all users</td>
<td>188</td>
</tr>
<tr>
<td>Viewing build results for an author</td>
<td>189</td>
</tr>
<tr>
<td>Generating reports on selected authors</td>
<td>190</td>
</tr>
</tbody>
</table>
Generating reports across multiple plans .................................................. 193
Viewing the Clover code-coverage for a job .............................................. 198
Viewing the Clover code-coverage for a build ........................................... 199
Integrating Bamboo with Atlassian applications ......................................... 200
Integrating Bamboo with JIRA................................................................. 201
  Linking JIRA issues to a build .............................................................. 203
  Viewing linked JIRA issues .................................................................... 204
Integrating Bamboo with Confluence ......................................................... 206
Integrating Bamboo with FishEye .............................................................. 207
Managing your user profile ....................................................................... 208
  Changing your password ....................................................................... 208
  Changing your notification preferences ................................................... 209
  Associating your author name with your user profile ................................. 209
Administering Bamboo ............................................................................. 210
  Administering plans ............................................................................. 211
    Moving plans to a different project ..................................................... 211
    Modifying multiple plans in bulk ....................................................... 214
  Monitoring job builds ........................................................................... 215
    Configuring the hanging build event ................................................... 215
    Configuring the build queue timeout event ........................................ 217
    Disabling build monitoring .................................................................. 218
  Configuring concurrent builds ................................................................ 219
System settings ......................................................................................... 219
  Viewing Bamboo's system information .................................................... 220
  Updating your Bamboo license details .................................................... 221
  Specifying Bamboo's title ..................................................................... 222
  Specifying Bamboo's URL ..................................................................... 222
  Logging in Bamboo .............................................................................. 223
  Enabling GZIP compression ................................................................... 226
  Enabling Bamboo's Remote API ............................................................. 226
  Configuring system properties ............................................................... 227
  Finding Your Bamboo Support Entitlement Number (SEN) ....................... 228
  Configuring Gravatar support .................................................................. 230
Agents and capabilities ............................................................................. 230
  About agents and capabilities ............................................................... 231
  Viewing Bamboo's agents ..................................................................... 234
Configuring agents ................................................................................... 243
  Creating a local agent .......................................................................... 244
  Editing an agent's details ..................................................................... 245
  Disabling or deleting an agent ............................................................... 246
  Viewing an agent .................................................................................. 247
    Viewing an agent's capabilities ............................................................ 247
    Viewing the jobs that an agent can build .......................................... 249
    Determining which agents can build which jobs .................................. 250
    Viewing an agent's system properties ................................................. 251
  Monitoring agent status ........................................................................ 252
Configuring capabilities ........................................................................... 254
  Viewing a capability's agents and jobs .................................................... 255
  Configuring a new executable ............................................................... 257
    Viewing your executable capabilities .................................................. 258
    Configuring an agent-specific executable capability ............................. 261
    Configuring a shared executable capability ......................................... 261
  Configuring a new JDK ......................................................................... 265
    Configuring an agent-specific JDK capability ....................................... 265
    Configuring a shared JDK capability ................................................... 266
    Viewing your JDK capabilities ............................................................ 268
  Configuring a new version control capability ......................................... 269
  Configuring a new custom capability ...................................................... 270
    Configuring an agent-specific custom capability .................................. 270
    Configuring a shared custom capability .............................................. 271
  Modifying and deleting capabilities ....................................................... 272
  Renaming a capability ............................................................................ 273

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
Remote agents ................................................................. 274
Disabling and enabling remote agents support .................. 274
Working with Elastic Bamboo ......................................... 275
About Elastic Bamboo .................................................... 275
  Elastic Bamboo Costs ................................................. 278
Getting started with Elastic Bamboo ................................. 278
Configuring Elastic Bamboo ......................................... 281
  Generating your AWS Private Key File and Certificate File .... 284
  Configuring elastic instances to use the EBS ..................... 286
Managing Elastic Bamboo ............................................. 293
  Managing your elastic images ..................................... 293
  Managing your elastic instances ................................ 311
  Managing your elastic agents .................................... 320
Running job builds using Elastic Bamboo ......................... 325
Disabling Elastic Bamboo ............................................ 327
Users and permissions ................................................. 327
Managing users .......................................................... 328
  Creating a user ...................................................... 329
  Changing a user's password or details ......................... 329
  Deleting or deactivating a user .................................. 330
  Granting administration rights to a user ....................... 331
  Changing usernames ................................................ 332
Connecting to external user directories ......................... 334
  Integrating Bamboo with Crowd ................................. 334
  Integrating Bamboo with LDAP .................................. 336
Managing groups ....................................................... 351
  Creating a group .................................................... 351
  Deleting a group .................................................... 352
  Changing members of groups ..................................... 352
Managing permissions ................................................. 353
  Granting plan permissions in bulk ............................... 354
  Granting global permissions to users or groups ............... 355
  Allowing anonymous access to Bamboo ....................... 357
  Global security and permission properties ..................... 358
    Allowing public signup ......................................... 358
    Displaying full details about users ......................... 359
    Using Captcha for failed logins .............................. 359
Plugins ................................................................. 360
  Installing a plugin ................................................ 362
  Upgrading your existing plugins ................................ 364
  Checking plugin compatibility for Bamboo upgrades ......... 365
  Configuring plugins ............................................... 366
    Enabling the Clover plugin .................................... 367
  Disabling or enabling a plugin .................................. 368
  Viewing the plugin audit log .................................... 369
  Viewing your installed plugins ................................ 370
  Uninstalling a plugin ............................................ 371
  Plugin blacklist .................................................... 371
Data and backups ..................................................... 372
  Locating important directories and files ....................... 372
  Specifying Bamboo's working directory ....................... 375
  Viewing your database connection details ..................... 375
  Moving your Bamboo data to a different database .......... 376
  Optimising or re-indexing data ................................ 376
  Specifying a backup schedule ................................... 377
  Exporting data for backup ....................................... 379
  Importing data from backup ..................................... 380
  Configuring global build results expiry ....................... 382
Security ............................................................... 385
  Agent authentication ............................................. 385
  Elastic Bamboo Security ........................................ 387
  Bamboo cookies ................................................... 389
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How can I pass bamboo variables to my build script</td>
<td>616</td>
</tr>
<tr>
<td>Bamboo resources</td>
<td>617</td>
</tr>
<tr>
<td>Contributing to the Bamboo Documentation</td>
<td>618</td>
</tr>
<tr>
<td>Tips of the Trade</td>
<td>619</td>
</tr>
<tr>
<td>Bamboo Documentation in Other Languages</td>
<td>622</td>
</tr>
<tr>
<td>Glossary</td>
<td></td>
</tr>
<tr>
<td>activity log</td>
<td>623</td>
</tr>
<tr>
<td>agent</td>
<td>625</td>
</tr>
<tr>
<td>agent-specific capability</td>
<td>625</td>
</tr>
<tr>
<td>artifact</td>
<td>625</td>
</tr>
<tr>
<td>author</td>
<td>625</td>
</tr>
<tr>
<td>build</td>
<td>626</td>
</tr>
<tr>
<td>build activity</td>
<td>626</td>
</tr>
<tr>
<td>build duration</td>
<td>626</td>
</tr>
<tr>
<td>build log</td>
<td>626</td>
</tr>
<tr>
<td>build queue</td>
<td>626</td>
</tr>
<tr>
<td>build result</td>
<td>626</td>
</tr>
<tr>
<td>build strategy</td>
<td>627</td>
</tr>
<tr>
<td>build telemetry</td>
<td>628</td>
</tr>
<tr>
<td>capability</td>
<td>628</td>
</tr>
<tr>
<td>child</td>
<td>629</td>
</tr>
<tr>
<td>committer</td>
<td>629</td>
</tr>
<tr>
<td>custom capability</td>
<td>629</td>
</tr>
<tr>
<td>elastic agent</td>
<td>629</td>
</tr>
<tr>
<td>elastic bamboo</td>
<td>629</td>
</tr>
<tr>
<td>elastic block store</td>
<td>629</td>
</tr>
<tr>
<td>elastic image</td>
<td>630</td>
</tr>
<tr>
<td>elastic instance</td>
<td>630</td>
</tr>
<tr>
<td>executable</td>
<td>630</td>
</tr>
<tr>
<td>favourites</td>
<td>630</td>
</tr>
<tr>
<td>global permission</td>
<td>631</td>
</tr>
<tr>
<td>Job</td>
<td>631</td>
</tr>
<tr>
<td>label</td>
<td>631</td>
</tr>
<tr>
<td>local agent</td>
<td>631</td>
</tr>
<tr>
<td>permission</td>
<td>631</td>
</tr>
<tr>
<td>Plan</td>
<td>631</td>
</tr>
<tr>
<td>plan permission</td>
<td>632</td>
</tr>
<tr>
<td>project</td>
<td>632</td>
</tr>
<tr>
<td>queue</td>
<td>632</td>
</tr>
<tr>
<td>reason</td>
<td>632</td>
</tr>
<tr>
<td>remote agent</td>
<td>633</td>
</tr>
<tr>
<td>remote agent supervisor</td>
<td>633</td>
</tr>
<tr>
<td>requirement</td>
<td>633</td>
</tr>
<tr>
<td>shared capability</td>
<td>633</td>
</tr>
<tr>
<td>Stage</td>
<td>634</td>
</tr>
<tr>
<td>Stock images</td>
<td>634</td>
</tr>
<tr>
<td>Task</td>
<td>636</td>
</tr>
<tr>
<td>triggering</td>
<td>637</td>
</tr>
<tr>
<td>watcher</td>
<td>637</td>
</tr>
</tbody>
</table>
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.

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

What is Bamboo?

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 traditionally performed to release a software application, including building and functional testing, tagging releases, assigning versions, and deploying and activating the new version in production.

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 a 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.

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

<table>
<thead>
<tr>
<th>Level</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>• Has one, or more, plans.</td>
</tr>
<tr>
<td></td>
<td>• Provides reporting (wallboard) across all plans in the project.</td>
</tr>
<tr>
<td></td>
<td>• Provides links to other applications.</td>
</tr>
</tbody>
</table>
Plan

- Has a single stage, by default, but can be used to group jobs into multiple stages.
- The stages in a plan are processed serially/consecutively.
- All jobs in a stage must succeed before the next stage can be processed.
- Specifies the default repository.
- Specifies how the build is triggered, and triggering dependencies between a plan and other plans in the project.
- Specifies notifications of build results.
- (Provides for the definition of plan variables.)

Stage

- Has a single job, by default, but can be used to group multiple jobs.
- The jobs within a stage may be processed in parallel.
- The artifacts produced in a stage can be made available for use by a subsequent stage.

Job

- Has one or more tasks.
- 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.
- Can only use artifacts produced in a previous stage.

Task

- Tasks are units of work, such as source code checkout, executing a Maven goal, running a script, or parsing test results.

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.

ℹ️ 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.

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.

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

<table>
<thead>
<tr>
<th>Source Repository</th>
<th>Git</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository URL</strong></td>
<td>The URL of Git repository.</td>
</tr>
<tr>
<td><strong>Branch</strong></td>
<td>The name of the branch (or tag) containing source code.</td>
</tr>
<tr>
<td><strong>Authentication Type</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Use shallow clones</strong></td>
<td>Fetches the shallowest commit history possible. Do not use if your build depends on full repository history.</td>
</tr>
</tbody>
</table>

### 3. Build strategy

The **build strategy** for a plan lets us choose how Bamboo gets triggered to run the build.

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

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

See [Triggering builds](#) for details.

### 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**

Build File

**build.xml**

Target

**clean test**

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


**Maven 3.0...**

Maven 3.x Configuration

Task Description

Executable

**Maven 3**

Goal

**clean test**

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.


**Grails...**
Grails Configuration

Task Description

Executable

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.


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](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 [Configuring 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. Build strategy
  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.

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.
3. Build strategy

The build strategy for a plan lets us choose how Bamboo gets triggered to run the build.

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

Choose Build Strategy > Repository triggers the build..., and optionally, specify an IP address for the repository server.

See Triggering builds for details.

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.

### NAnt...

#### NAnt Configuration

<table>
<thead>
<tr>
<th>Task Description</th>
</tr>
</thead>
</table>

**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.


### Visual Studio...

#### Visual Studio Configuration

<table>
<thead>
<tr>
<th>Task Description</th>
</tr>
</thead>
</table>

**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.


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 which 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 appears 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.
- ⏹️ This plan has been disabled.

Screenshot: Bamboo Dashboard - 'All Plans' tab

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
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.

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
Configuring 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
Administrering Bamboo
Release notes
Bamboo security advisories
Bamboo documentation downloads

Configuring plans

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

Plans:

- are organised into one or more stages;
- have one or more jobs in each stage;
• contain a single 'Default job' in a single stage, straight after creating a new plan;
• define default settings for what gets built by jobs in the plan (i.e. the 'default source repository');
• define how the plan's build is triggered;
• define who will be notified of the job's build result;
• define who has permission to view and perform various actions on the plan and its jobs.

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:

• Creating a plan
• Managing plans
• Defining plan variables
• Using stages in a plan
• Using plan branches

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.

Screenshot: The Plan Summary page
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:

1. Click Create Plan in the top navigation bar.
2. 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
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:

1. Click Create Plan in the menu bar, and then click Create a New Plan.
2. Complete the following sections:

<table>
<thead>
<tr>
<th>Plan Details</th>
<th>Either select an existing project to which this plan will belong, or click New Project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Repositories</td>
<td>See the following for details: Bitbucket - Mercurial, CVS, Git, GitHub, Mercurial, Perforce, Subversion</td>
</tr>
<tr>
<td>Build Strategy</td>
<td>Choose how Bamboo should trigger builds. See Triggering builds.</td>
</tr>
</tbody>
</table>

Step 2. Configure tasks for the plan
When a new plan is created, a default job is also created as part of the plan. As part of creating your plan, you need to configure the tasks for the default job. You can 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 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
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 build strategy (defined above). To enable this plan, select the Yes please! check box.
5. Click Create to create the plan. The 'Plan Summary' page will be displayed.

When you next return to the Dashboard, your new plan (and new project, if applicable) will be displayed on the All Plans tab. Bamboo will also automatically run an initial build for your new plan, unless you have opted for the Manual & dependent builds only build strategy.

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)

Notes
- You can configure Bamboo to automatically start an initial build for a new plan, even if you have selected the Manual & dependent builds only build strategy. 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. Start the Create Plan wizard as described on Creating a plan 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 build strategy.
5. 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, unless the plan you cloned uses the Manual & dependent builds only build strategy.

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 a new plan, even if the plan you cloned uses the Manual & dependent builds only build strategy. 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

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. Start the Create Plan wizard as described on Creating a plan and click Import a Maven 2 Project. Maven 2 needs to be installed on the Bamboo server machine; see Configuring a new executable.
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 build strategy.

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, unless the plan you cloned uses the Manual & dependent builds only build strategy.

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**: Subversion
- **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
- **Password**: (Optional) The password required by the subversion username

[Import] [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 build strategy, but you can change this by editing this plan. You can configure Bamboo to automatically start an initial build for a new plan, even if you have selected the Manual & dependent builds only build strategy.
strategy. To do so, add the `fire.initial.build.for.manual.strategy` to your `bamboo.cfg.xml` file as described in Configuring system properties.

---

Managing plans

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

Editing a plan's configuration

To edit an existing plan:

1. Click Dashboard.
2. Click the All Plans tab.
3. Click the plan in the list and then choose Actions > Configure Plan.
4. Click a tab to edit 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.
   - 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.

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
Configuring a plan's permissions

The process for changing the permissions for a particular plan are described below. Note that, for ongoing ease of management, it is recommended that you grant permissions to groups rather than to individual users.

To change plan permissions:

1. Navigate to the configuration for the desired plan, as described on Editing a plan's configuration.
2. Click the Permissions tab.
3. You can change 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.

Related pages:
- Editing a plan's configuration
- Granting plan permissions in bulk
- Managing permissions

<table>
<thead>
<tr>
<th>User category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logged in Users</td>
<td>Users who are logged in to Bamboo.</td>
</tr>
<tr>
<td>Anonymous Users</td>
<td>Users who are not logged in to Bamboo.</td>
</tr>
</tbody>
</table>
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: Plan Permissions

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
- Exporting data for backup
- Disabling or deleting a job
- Stopping an active build

### 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 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](#)

To label a build result:

1. Navigate to the desired plan.
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](#)

**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 [Using global, plan or build-specific 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 [Triggering a plan build manually](#).

**Related pages:**
- [Defining global variables](#)
- [Using global, plan or build-specific variables](#)
- [Triggering a plan build manually](#)

Before you begin:

- Note that plan variables override global variables with the same name.
To define a plan variable:

1. Navigate to the configuration for the desired plan, as described on Editing a plan's configuration.
2. Click the Variables tab.
3. 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

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. Stages have the following characteristics:

- A stage may contain one or more jobs;
- Depending on the availability of Bamboo agents, all jobs in a stage can be processed in parallel;
- Stages are processed consecutively within a plan, one at a time;
- All jobs in a stage must be built and succeed before Bamboo builds any jobs in the next stage. If any job fails in a stage, no further stages in the plan will be processed and the plan's build will fail.

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
- Deleting a stage
- Editing a stage
- Rerunning a failed 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).

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. Stages have the following characteristics:

- A stage may contain one or more jobs;
- Depending on the availability of Bamboo agents, all jobs in a stage can be processed in parallel;
- Stages are processed consecutively within a plan, one at a time;
- All jobs in a stage must be built and succeed before Bamboo builds any jobs in the next stage. If any job fails in a stage, no further stages in the plan will be processed and the plan's build will fail.
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
- Editing a job
- Configuring a job's build artifacts

**To create a stage within a plan:**

1. Navigate to the stages for a plan, as described in Using stages in a plan.
2. Click Configuration in the left navigation panel and then Create Stage on the right. The 'Create a new Stage' dialog will appear.
3. Complete the form and click Create. For information about manual stages, see Configuring a stage (About manual stages).
4. (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**

**Deleting a stage**

Before you begin, please see the Notes for this page.

**To delete a stage from a plan:**

1. Navigate to the stages for a plan, as described in Using stages in a plan.
2. Click Configuration in the left navigation panel.
3. Click the cog (⚙️) icon for the relevant stage and choose Delete Stage.
4. Click Confirm to delete the stage. Note that a deleted stage cannot be recovered.
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.

**Editing a stage**

To edit a stage for a plan:

1. Navigate to the stages for a plan, as described in Using stages in a plan.
2. Click **Configure** in the left navigation panel.
3. 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**
1. 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](#). The failed stage will be displayed in 'Failed Stage' area on the Summary tab.
2. Click the button with the stage name 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](#).
- If you want to keep track of the failure reason, you might want to add a comment to the build result to record it. The existing build result will be overwritten (Bamboo will not create a new build) and the previous failure reason will not be retained.

**Related pages:**

- [Editing a plan's configuration](#)
- [Configuring jobs](#)
- [Using stages in a plan](#)
Using plan branches

Plan branches are used to represent a branch in your version control repository, where the plan branch uses 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. This guarantees that their new code functions correctly before sharing the changes with the rest of the team.

However, changes made on a branch may not be built and tested by Bamboo unless the developer sets up a new build plan, or clones an existing plan that is configured to build the new branch. When using plan branches, any new branch that is created is automatically built and tested using the build configuration of the plan that plan branches was enabled for. Optionally, changes from the feature branch can be merged back to the "master" (e.g. trunk, default or mainline branch).

Note that you need to 'favourite' a plan branch in order to receive notifications about build results for it.

Activating plan branching

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

Whichever way the plan branch is created, the default plan maintains the structure and configuration of its branch plans. However, you can override the following settings in a branch plan:

- the source repository, so that the branch uses a different repository from the default plan.
- the plan variables, so that the branch uses different variables from the default plan.

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

You can click Detect Branches to get Bamboo to check for branches in the specified repository for the plan.

**Auto branching**

Use auto branching for Git and Mercurial repositories. For other repository types, use manual branching.

**To have Bamboo automatically create a branch plan whenever the repo branches:**

1. Go to the Branches tab in the configuration pages for the plan you wish to branch.
2. Select Enabled, under 'Automatic Branch Detection'.
3. Enter a regular expression to specify the repo branch names for which plan branches will be created. See the Java documentation on regular expressions.
4. Select Enabled, under 'Automatic Branch Clean-up', if required.
5. Click Save.

**Manual branching**

Use manual branching for all supported repository types. You can use auto branching for Git and Mercurial 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.
3. Select from the available VCS branches, then click Create.
4. Select Override default's Repository in the Source Repository tab if you want the branch plan to use a different repository from the default plan. See Specifying the source repository and it's child pages for details.
5. Click the Variables tab to override the plan variables of the default plan.

*Screenshot: The Plan Summary page for a branch, showing the 'branches' menu.*
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 master repo is only updated with changes in the branch that have built successfully.
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. Select Branch Integration 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 only if you want those changes merged back into your branch once the build completes successfully.
5. Click Save.
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. Select Branch Integration 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 only if you want your changes pushed to the other repo once the build completes successfully.
5. Click Save.
Limitations with plan branches

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

<table>
<thead>
<tr>
<th>Action</th>
<th>Limitations</th>
</tr>
</thead>
</table>
| Auto plan branching | • Can only be used with Git and Mercurial 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.)                                      |

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:
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 SCMs. For details, please refer to the following pages (and see Notes below):

- Bitbucket - Mercurial
- CVS
- Git
- GitHub
- Mercurial
- Perforce
- Subversion

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

On this page:

- Viewing the source repository for a plan
- Viewing the source repository for a job
- Notes

Related pages:

- Connecting to code repositories
- Checking out code
- Configuring a shared source repository
- Configuring plans

Screenshot: Configuring a job — Subversion Source Repository
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
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](#).

**Notes**

- A number of source repositories are supported 'out of the box', as described on the **Supported platforms** 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.

**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**.

---

*(Image of Bamboo UI interface with Bitbucket source repository settings)*
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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Repositories</td>
<td>Retrieves all repositories you have explicit permissions to access from Bitbucket when you click Load Repositories.</td>
</tr>
<tr>
<td>URL</td>
<td>Allows you to specify a Repository URL (eg. <a href="http://bitbucket.org/sindbad/ogre">http://bitbucket.org/sindbad/ogre</a>). To choose a branch to work with, other than the default branch, click Load Branches.</td>
</tr>
<tr>
<td>Branch</td>
<td>Pick a branch if you want to check out code from a branch other than the default branch.</td>
</tr>
</tbody>
</table>

Advanced Options

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command timeout</td>
<td>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.</td>
</tr>
<tr>
<td>Verbose logs</td>
<td>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.</td>
</tr>
<tr>
<td>Enable Quiet Period</td>
<td>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.</td>
</tr>
<tr>
<td>Include/Exclude Files</td>
<td>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.</td>
</tr>
<tr>
<td>Exclude Changesets</td>
<td>Enter a regular expression to match the commit messages for changesets that should not start a build.</td>
</tr>
</tbody>
</table>
### Web Repository

| **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)  

| **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/’).

**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-tests/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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVS Root</td>
<td>The full path to your CVS repository root (e.g. <code>pserver:ver.me@cvs.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 <a href="#">Using global, plan or build-specific variables</a>).</td>
</tr>
<tr>
<td><strong>TIP</strong></td>
<td>If you are importing a Maven 2 Project, this location should contain your project's <code>pom.xml</code> file.</td>
</tr>
</tbody>
</table>
| 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.  
*(Only available when configuring an existing plan)*
<table>
<thead>
<tr>
<th><strong>Module</strong></th>
<th>Type the name of the CVS module that contains the source-code.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>a. Create a module (e.g. allbuilds).</td>
</tr>
<tr>
<td></td>
<td>b. Define an ampersand module with the same name. (The ampersand module can be empty.)</td>
</tr>
<tr>
<td></td>
<td>c. In the <strong>Module</strong> field, enter the following: allbuilds allbuilds &amp;project2 &amp;project2 &amp;project3</td>
</tr>
<tr>
<td><strong>Version of module</strong></td>
<td>The version of the module that Bamboo should build:</td>
</tr>
<tr>
<td></td>
<td>• <strong>HEAD</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>Branch/Tag</strong> – supply the name of the branch or tag.</td>
</tr>
<tr>
<td></td>
<td>Note that you can use global variables in this field (see Using global, plan or build-specific variables).</td>
</tr>
</tbody>
</table>

**Advanced Options**

<table>
<thead>
<tr>
<th><strong>Include/Exclude Files</strong></th>
<th>Allows you to specify the files that Bamboo should, or should not, use to detect changes.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enter into <strong>File Pattern</strong> 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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Exclude Changesets</strong></th>
<th>Enter a regular expression to match the commit messages for changesets that should not start a build.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web Repository</strong></td>
<td>If your repository can be viewed in a web browser, select the repository type.</td>
</tr>
<tr>
<td></td>
<td>This allows links to relevant files to be displayed in the 'Code Changes' section of a build result.</td>
</tr>
<tr>
<td><strong>Generic Web Repository</strong></td>
<td>- <strong>Web Repository URL</strong> – the URL of the repository.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Web Repository Module</strong> — the particular repository required for this plan or job, if the Web Repository URL above points to multiple repositories.</td>
</tr>
</tbody>
</table>
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/').

---

**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** build strategy has been specified 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).
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:

```bash
cvs -d cvsroot-to-your-repository checkout CVSROOT
```

where:

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

> Using `-d` overrides the any `$CVSROOT` environment variable setting.

The following files should be checked out:

- `CVSROOT/checkoutlist`
- `CVSROOT/commitinfo`
- `CVSROOT/config`
- `CVSROOT/cvswrappers`
- `CVSROOT/editinfo`
- `CVSROOT/loginfo`
- `CVSROOT/modules`
- `CVSROOT/notify`
- `CVSROOT/rcsinfo`
- `CVSROOT/taginfo`
- `CVSROOT/verifymsg`

**Step 2. Install the pre-commit trigger**

Add a line like the following example’s to the `CVSROOT/commitinfo` pre-commit trigger file. The `CVSROOT/com
mitinfo 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
```

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 CVSROOT/loginfo post-commit trigger file. The CVSROOT/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 %{}
http://bamboo-base-url MOO-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.
- %{} is how CVS tells the postCommitBuildTrigger.sh script which directory it is committing.
- MOO-KEY is the key of the Bamboo plan to be executed.

**Step 4. Save the changes back to CVS**

Commit the changes you made to the CVSROOT/commitinfo and CVSROOT/loginfo 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**

The instructions on this page describe how to configure Bamboo to use a Git source repository.

**Related pages:**

- **Specifying the source repository**
- **GitHub**

**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 (e.g.: `git://github.org/atlassian/bamboo-git-plugin.git`).

Valid URLs are of the form:
- `git://host.xz[:port]/path/to/repo.git`
- `ssh://[user@]host.xz[:port]/path/to/repo.git`
- `[user@]host.xz:path/to/repo.git`
- `http[s]://host.xz[:port]/path/to/repo.git`
- `/path/to/repo.git`
- `file:///path/to/repo.git`

### 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.
<table>
<thead>
<tr>
<th><strong>Include/Exclude Files</strong></th>
<th>Allows you to specify the files that Bamboo should, or should not, use to detect changes. Enter into <strong>File Pattern</strong> a <a href="https://www.example.com">regular expression</a> to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See <a href="https://www.example.com">sub page</a> for examples.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exclude Changesets</strong></td>
<td>Enter a <a href="https://www.example.com">regular expression</a> to match the commit messages for changesets that should not start a build.</td>
</tr>
<tr>
<td><strong>Web Repository</strong></td>
<td>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 <a href="https://www.example.com">Code Changes</a> section of a build result.</td>
</tr>
</tbody>
</table>
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/').

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

<table>
<thead>
<tr>
<th><strong>Use submodules</strong></th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command timeout</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Verbose logs</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Enable Quiet Period</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Include/Exclude Files</strong></td>
<td>Allows you to specify the files that Bamboo should, or should not, use to detect changes. Enter into <strong>File Pattern</strong> a <strong>regular expression</strong> to match the files that Bamboo includes or excludes. The regex pattern must match the file path in the repository. See <a href="#">sub page</a> for examples.</td>
</tr>
<tr>
<td><strong>Exclude Changesets</strong></td>
<td>Enter a <strong>regular expression</strong> to match the commit messages for changesets that should not start a build.</td>
</tr>
<tr>
<td><strong>Web Repository</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
**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/').

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:

<table>
<thead>
<tr>
<th>Repository URL</th>
<th>The full path to your Mercurial repository (eg: git://bitbucket.org/atlassian/bamboo-git-plugin.git)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Valid URLs are of the form:</td>
</tr>
<tr>
<td></td>
<td>• local/filesystem/path[#revision]</td>
</tr>
<tr>
<td></td>
<td>• file://local/filesystem/path[#revision]</td>
</tr>
<tr>
<td></td>
<td>• http[s]://[user[:pass]@]host[:port]/[path][#revision]</td>
</tr>
<tr>
<td></td>
<td>• ssh://[user[:pass]@]host[:port]/[path][#revision]</td>
</tr>
<tr>
<td></td>
<td>(for further references visit Mercurial documentation)</td>
</tr>
<tr>
<td>Branch</td>
<td>The name of the relevant branch (or tag) you want to work on. Leave empty to work on default branch.</td>
</tr>
<tr>
<td>Username</td>
<td>The username (if any) required to access the repository.</td>
</tr>
<tr>
<td>Authentication</td>
<td>Password</td>
</tr>
<tr>
<td></td>
<td>Keyfile with passphrase</td>
</tr>
<tr>
<td></td>
<td>Keyfile without passphrase</td>
</tr>
<tr>
<td></td>
<td>Default Mercurial credentials</td>
</tr>
</tbody>
</table>

Advanced Options

<table>
<thead>
<tr>
<th>Command timeout</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbose logs</td>
<td>Turns on --verbose and --debug options in hg commands and passes the output to build logs. Use that option if you encounter problems with Mercurial in Bamboo.</td>
</tr>
<tr>
<td>Location of POM file</td>
<td>Type the path to your project's pom.xml file which is relative to the root of your Mercurial Repository URL (defined above).</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><em>(Only available when importing a Maven 2 project)</em></td>
</tr>
<tr>
<td>Disable repository caching</td>
<td>Select this option to enable subrepositories support.</td>
</tr>
<tr>
<td>Enable Quiet Period</td>
<td>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.</td>
</tr>
<tr>
<td>Include/Exclude Files</td>
<td>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.</td>
</tr>
<tr>
<td>Exclude Changesets</td>
<td>Enter a regular expression to match the commit messages for changesets that should not start a build.</td>
</tr>
<tr>
<td>Web Repository</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Mercurial Web Repository – select one of the following viewer schemes:</td>
</tr>
<tr>
<td></td>
<td>• BitBucket Web Repository Scheme (if you use BitBucket)</td>
</tr>
<tr>
<td></td>
<td>• Default Web Repository Scheme (hgserve) (Mercurial's own default web server)</td>
</tr>
</tbody>
</table>
FishEye — specify the URL and other details for the repository:

- **FishEye URL** — the URL of your FishEye repository (e.g. 'https://atlas-eye.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/').

### 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-tests/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 remote repository:

*Screenshot: Choosing the 'Keyfile with passphrase' option for a plan or job*
If you need to access 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 among 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:

<table>
<thead>
<tr>
<th>Port</th>
<th>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.</th>
</tr>
</thead>
</table>
### 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 `//<clientname>/<workspace_mapping>/`

... For details please see the Perforce User's Guide.

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.
<table>
<thead>
<tr>
<th>Web Repository</th>
<th>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.</th>
</tr>
</thead>
</table>
| **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. |
| **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/'). |

**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.

<table>
<thead>
<tr>
<th>On this page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Running builds on multiple remote agents or machines</td>
</tr>
<tr>
<td>2. Using Perforce Overlay and Exclusionary Mappings in Bamboo</td>
</tr>
</tbody>
</table>

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 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.

ℹ️ 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.

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:
Consequently, no changes to files in //clientName/depot/Prj/... 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

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:

<table>
<thead>
<tr>
<th><strong>Repository URL</strong></th>
<th>The location of your Subversion repository e.g. <a href="http://svn.collab.net/repos/svn/trunk">http://svn.collab.net/repos/svn/trunk</a> Note that you can use global variables in this field (see Using Global or Build-specific Variables).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>(Optional) The Subversion username (if any) required to access the repository.</td>
</tr>
<tr>
<td>Authentication Type</td>
<td>Password – choose this option if you want to authenticate with a username and password.</td>
</tr>
</tbody>
</table>

If you are importing a Maven 2 Project, this location should contain your project's pom.xml file.
| 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.  
<p>| 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. <em>(Only available when configuring an existing plan.)</em> |</p>
<table>
<thead>
<tr>
<th><strong>Include/Exclude Files</strong></th>
<th>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 <a href="#">sub page</a> for examples.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exclude Changesets</strong></td>
<td>Enter a regular expression to match the commit messages for changesets that should not start a build.</td>
</tr>
<tr>
<td><strong>Web Repository</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Generic Web Repository</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Web Repository URL – the URL of the repository.</td>
</tr>
<tr>
<td></td>
<td>• Web Repository Module — the particular repository required for this plan or job, if the Web Repository URL above points to multiple repositories.</td>
</tr>
</tbody>
</table>
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/').

### 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.

### 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:

```bash
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.tmpl` file in the `hooks/` directory, name it `post-commit` and make it executable:

```bash
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.

```bash
/path-to-your-bamboo-installation/scripts/svn-triggers/postCommitBuildTrigger.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

**Configuring jobs and tasks**

The following pages contain information on configuring jobs and tasks for your Bamboo plans. If you are looking for information about Bamboo builds, please see Working with builds.

• Configuring jobs
• Configuring tasks
• Using global, plan or build-specific variables

Screenshot: Configuring tasks for a job
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 is made up of one or more *tasks*. A job defines:

- what gets built (i.e. the source code repository) — this can be a custom, job-specific source repository or the ‘default source repository’ of the plan that contains this job;
- which *agent capabilities* are required for the build (based on job-specific requirements and requirements of the job’s tasks);
- what *tasks* make up the job and the order in which they are executed;
- what *artifacts* the job’s build will produce;
- 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).

**Related pages:**

- Creating a job
- Editing a job
- Disabling or deleting a job
- Viewing a job's Maven dependencies
- Configuring a test task

**To navigate to a job:**

1. Click Dashboard and then the All Plans tab.
2. Click the latest build in the 'Builds' column, for the plan that has the job. The plan's 'Build Result Summary' page will be displayed.
3. Click the name of the desired job in the left panel, to see the 'Job Result Summary' page.

Creating a job
This page describes how to create a Bamboo job in a stage.

- 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.
- Creating a new job allows you to define a single unit of execution within a stage, including what gets built and what builder to use.

**Related pages:**
- Configuring plans
- Using stages in a plan
- Editing a job
- Disabling or deleting a job

To create a new job:
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 Configuration under 'Stages & Jobs' in the left panel.
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 Editing a job.

**Screenshot: Cloning an existing job**
Editing a job

To edit an existing job in a Bamboo plan:

1. Navigate to the job, as described on Configuring jobs.
2. Choose Actions > Configure Job.
3. Click the 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 (from step 3).
   - **Post Actions** — see Configuring automatic labelling of build results (from step 3).
   - **Miscellaneous** — see Configuring miscellaneous settings for a job.

**Screenshot: The Tasks tab of a job's configuration pages**
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:

- **Executable capability** — Every task has one 'Executable' requirement. A job inherits the 'Executable' requirements of its tasks.
- **JDK capability** — A task has either one 'JDK' requirement or none, depending on the task’s 'Executable'. A job inherits the 'JDK' requirements of its tasks.
- **Custom capability** — A job can have multiple 'custom' requirements (or none). Before you can specify a custom requirement in your job, you must first define that custom requirement in your Bamboo system (see Configuring a new custom 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 'operating.system=WindowsXP' for the appropriate agent(s), and specify it as a requirement for this job.
- **Perforce capability** — A job will have a 'Perforce' requirement automatically specified, if Perforce was selected as the source repository.

On this page:

- Specifying extra requirements for a job
- Viewing current capable agents

Related pages:

- Editing a job
- Configuring tasks
- Viewing an agent's capabilities
- Viewing a capability's agents and jobs

Specifying extra requirements for a job
A job's will inherit the requirements of its tasks by default. However, you can specify extra requirements for the job, in addition to the task requirements.

To specify extra requirements for a job:

1. Navigate to the desired job's configuration pages, as described on Editing a job.
2. Click the Requirements tab (see screenshot above). 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**

<table>
<thead>
<tr>
<th>Job Details</th>
<th>Tasks</th>
<th>Requirements</th>
<th>Artifacts</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Requirements</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

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

**Screenshot: Artifact definitions for a job**

<table>
<thead>
<tr>
<th>Job Details</th>
<th>Tasks</th>
<th>Requirements</th>
<th>Artifacts</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Artifacts Definitions

Jobs may create Artifacts you want to keep or share with others. Jobs (e.g. Clover, Agile, JAR files, etc). Artifact Definitions allow you to define which Artifacts you care about.

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Copy Pattern</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributions</td>
<td>target/checkout/distribution/standalone/target/install</td>
<td><strong>/*.jar</strong></td>
<td>Unshare</td>
</tr>
<tr>
<td>Source</td>
<td>target/checkout/distribution/source/target</td>
<td>*.zip</td>
<td>Unshare</td>
</tr>
<tr>
<td>WAR</td>
<td>target/checkout/distribution/war/target/</td>
<td>*.war</td>
<td>Unshare</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Destination directory</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angry Nerd - Stable Website</td>
<td>Root of Working Directory</td>
<td>Edit</td>
</tr>
</tbody>
</table>

### 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

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.

On this page:
- Sharing artifacts between jobs
- Sharing artifacts between Maven jobs (Beta)
- Notes

Related pages:
- Viewing a build's artifacts
- Configuring a job's build artifacts

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. Tick 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.

Screenshot: Creating an artifact dependency

Sharing artifacts between Maven jobs (Beta)

Before you begin:
- Maven artifact sharing is currently in beta. We recommend that you do not use it with any critical production systems.
- Maven artifact sharing is not supported for Maven 1.

About Maven artifact sharing

Maven artifact sharing works by producing new remote repositories when you run a plan. These are temporary equivalents of the deployment repositories configured within the project's pom.xml files. For example, if the project deploys its repository to the 'mycorp-private' repository at http://repository.example.com/ then Bamboo will create a 'mycorp-private' repository for each Plan result hosted on the Bamboo server.

When a job produces a Maven artifact, it is deployed to this repository. A subsequent job that consumes an artifact created by this process will resolve it from the repository within Bamboo. This process of deploying and
resolving artifacts can then be repeated, as necessary across subsequent jobs in later stages. Once a build has completed, the temporary repositories are removed to conserve disk space, unless specified otherwise (see configuration instructions below).

The diagram below shows an example of how artifact sharing works. Two jobs, Job A and Job B, share an artifact in the plan, MyPlan. Job A builds the artifact then deploys it to the temporary remote Maven repository. Job A triggers Job B. Job B resolves the artifact, builds, then deploys the artifact back to the repository.

Diagram: Maven artifact sharing example

Configuring Maven artifact sharing

Artifact sharing is configured differently when using Maven 2 or 3 as a builder for your jobs. Artifacts are deployed to and resolved from a Maven repository, rather than copied from agent to agent. You will need to change your pom.xml file, as well as configure the relevant jobs to set up artifact sharing.

Before you begin:

- Your Bamboo URL must be specified correctly for Maven artifact sharing to work correctly. For example, do not use localhost as part of your Bamboo URL. For more information, see Specifying Bamboo's URL.

To share an artifact between two Maven jobs in different stages:

1. Edit your pom.xml file and add the following plugin definition:
<build>
  <plugins>
    <plugin>
      <groupId>com.atlassian.bamboo.maven.sharing</groupId>
      <artifactId>bamboo-artifact-sharing-maven-plugin</artifactId>
      <version>3.0-i5</version>
      <executions>
        <execution>
          <id>sharing</id>
          <goals>
            <goal>share</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>

• Note, the version number specified above (<version>) must match the version of Bamboo you are using. This version number will be compatible across minor releases of Bamboo, unless specified otherwise in the relevant upgrade.

2. Navigate to the plan that you want to enable artifact sharing for, and click the Miscellaneous tab (see screenshot below).

3. Select the Enabled checkbox.

4. If you don't want to keep the Maven artifact once the plan build has completed, select Expire Maven Repository Artifacts.

   Note that each plan builds may result in hundreds of megabytes of Maven artifacts. Ensure that you have sufficient disk space to accommodate this.

5. Configure artifact sharing between the desired jobs. That is,
   • If a job needs to deploy an artifact (i.e. produce the artifact), mark the artifact as shared in the relevant artifact definition.
   • If a job needs to resolve an artifact (i.e. consume the artifact), create an artifact dependency for the artifact as described above.

Screenshot: Enabling Maven artifact sharing on the Miscellaneous tab for a plan

<table>
<thead>
<tr>
<th>Maven Artifact Sharing (Beta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
</tr>
<tr>
<td>Enables support for Maven Artifact Passing (Beta)</td>
</tr>
<tr>
<td>Expire Maven Repository Artifacts</td>
</tr>
<tr>
<td>Expire the Maven Repository Artifacts once the plan has finished executing</td>
</tr>
</tbody>
</table>

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 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
To configure the miscellaneous settings for a job:

1. Navigate to the desired job, as described in Editing a job.
2. Edit the desired settings as follows:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Override default hanging build detection</strong></td>
<td>Override the <strong>default build hanging detection</strong> settings. These settings determine when a build hung event is thrown (e.g. you can <strong>configure your notifications</strong> to trigger from this event).</td>
</tr>
<tr>
<td><strong>Build Time Multiplier</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Log Quiet Time</strong></td>
<td>The amount of time since Bamboo last recorded an entry in the <strong>build log</strong> for a build. The 'Expected Build Time' and 'Log Quiet Time' must both be exceeded for Bamboo to throw the build hung event.</td>
</tr>
<tr>
<td><strong>Build Queue Timeout</strong></td>
<td>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 <strong>Configuring the build queue timeout event</strong>).</td>
</tr>
<tr>
<td><strong>NCover output will be produced</strong></td>
<td><strong>Do not select this option. NCover</strong> is a code coverage tool that supports .NET projects.</td>
</tr>
<tr>
<td><strong>Use Clover to collect Code Coverage for this build</strong></td>
<td>Select this check box if:</td>
</tr>
<tr>
<td></td>
<td>• This job will be building a Java or Groovy-based project using a builder such as Ant, Maven or Grails.</td>
</tr>
<tr>
<td></td>
<td>• You are running <strong>Atlassian Clover</strong> and want to collect code coverage data to view from within Bamboo (see <strong>Viewing the Clover code-coverage for a build</strong>).</td>
</tr>
<tr>
<td><strong>Automatically integrate Clover into this build</strong></td>
<td><strong>Generate a Clover Historical Report</strong> — shows the current coverage results compared with previous Clover code coverage reports.</td>
</tr>
<tr>
<td></td>
<td><strong>Generate a JSON report</strong> — gives the Clover results in a format ready for embedding into applications or external report views.</td>
</tr>
</tbody>
</table>

You will also need to insert a Clover license (evaluation licenses are available) into the field provided. See **Enabling the Clover plugin**.
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:**

- Editing a job

### 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 Editing a job.
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

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:

1. Navigate to the desired job, as described on Configuring jobs.
2. Click the desired build result number in the 'Recent History' of the Job Summary.
3. 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
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.

---

**Disabling a job**

To disable a job:

1. From the dashboard, choose the plan which has the job, and then, in the Plan Navigator, click on the job to be disabled.
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. From the dashboard, choose the plan which has the job, and then, in the Plan Navigator, click on the job to be deleted.
2. Choose Actions > Configure Job.
3. 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.

To delete a job's current working files:

1. Access the job whose current working files you want to delete. To do this:
   a. Click Dashboard and then the All Plans tab.
   b. Click the name of the plan for the job. The 'Plan Summary' page will be displayed.
   c. Choose Actions > Configure Plan.
   d. Click the name of the job you wish to access from under the 'Stages & Jobs' section.
      - If necessary, expand the appropriate stage to access your job.
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 an operation that 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

---

**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
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*
Notes

- **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.
- **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 Configuring a new JDK.

Checking out code

Bamboo uses the 'Source Code Checkout' task to check out repositories into the working directory on the agent.

Using this task it is 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. Select the desired repository from the **Repository** list. 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.

4. Specify a Checkout Directory location, which is where the contents of the selected repository will be checked out to when the task executes.

5. Click Add Repository, at the bottom of the 'Task' screen, to add another repository to the task.

6. Click Save.

Screenshot: Configuring a Source Code Checkout task

Notes
- A number of source repositories are supported 'out of the box', as described on the Supported Platforms 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
Builder tasks execute a job's builds using build tools. Build tools are used to automate build processes. The range of features for a build tool differ significantly from one build tool to the next. Generally, build tools compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Build tools typically have their own form of scripting language and the ability to manage dependencies correctly.

Related pages:
- Configuring a test task

See the following pages for more information on configuring specific builder tasks:
Ant
This page describes how to configure a Bamboo task that uses an Ant executable.

See Configuring a builder task for an overview of Bamboo builder tasks.

Related pages:
- Configuring tasks
- Editing a job

To configure an Ant task:
1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired Ant task, or click Add Task and then Ant if creating a new task.
3. Complete the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Description</td>
<td>A description of the task, which is displayed in Bamboo.</td>
</tr>
<tr>
<td>Executable</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Build File</strong></td>
<td>The name of your existing build file (e.g. <code>build.xml</code>). You can include variables (see Using Global or Build-specific Variables).</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td>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>-D java.awt.headless=true</code>). Multiple Ant targets can be specified with a space-delimited list. You can also include variables (see Using Global or Build-specific Variables).</td>
</tr>
<tr>
<td><strong>Build JDK</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

4. If required, specify environment variables and working directory settings:

<table>
<thead>
<tr>
<th><strong>Environment Variables</strong></th>
<th><em>(Optional)</em> 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=&quot;-Xms200m -Xmx700m&quot;).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Working Sub Directory</strong></td>
<td><em>(Optional)</em> 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.</td>
</tr>
</tbody>
</table>

5. If applicable, select **The build will produce test results** and specify the 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/cvs-module>`.

6. Click **Save**.
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.
To configure a command task:

1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired Command task, or click Add Task and then Command if creating a new task.
3. Complete the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Description</td>
<td>A description of the task, which is displayed in Bamboo.</td>
</tr>
<tr>
<td>Executable</td>
<td>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.</td>
</tr>
<tr>
<td>Argument</td>
<td>(Optional) The relevant argument to pass to the command. Note that arguments which contain spaces must be quoted. You can include variables (see Using global, plan or build-specific variables).</td>
</tr>
</tbody>
</table>

4. If required, specify environment variables and working directory settings:

<table>
<thead>
<tr>
<th>Environment Variables</th>
<th>(Optional) 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=&quot;-Xms200m -Xmx700m&quot;).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working Sub Directory</td>
<td>(Optional) 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.</td>
</tr>
</tbody>
</table>

5. Click Save.
Grails
This page describes how to configure a Bamboo task that uses a Grails executable.

Bamboo supports Grails versions 1.2.x, 1.3.x, and 2.x.

**Related pages:**
- Configuring tasks
- Editing a job
- Configuring an Agent-specific JDK capability

**To configure a Grails task:**

1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired Grails task, or click Add Task and then Grails if creating a new task.
3. Complete the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Description</td>
<td>A description of the task, which is displayed in Bamboo.</td>
</tr>
<tr>
<td>Executable</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 Using global, plan 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.

4. If required, specify environment variables and working directory settings:

Environment Variables

(Optional) 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

(Optional) 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. If applicable, select The build will produce test results and 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/<cvs-module>.
6. Click **Save**.
Grails Configuration

Task Description

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

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?

Specify custom results directories

**/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...
Maven
This page describes how to configure a Bamboo task that uses a Maven executable. Apache Maven is a tool used for building and managing Java-based projects.

### Related pages:
- Configuring tasks
- Editing a job
- Viewing a job's Maven dependencies
- Configuring an Agent-specific JDK capability

To configure a Maven task:

1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired Maven task, or click Add Task and then a Maven option (e.g. Maven 2.x) if creating a new task.
3. Complete the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task Description</strong></td>
<td>A description of the task, which is displayed in Bamboo.</td>
</tr>
<tr>
<td><strong>Executable</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
| **Goal**             | The Maven goal that Bamboo will execute.  
  - You can use `-D` to define one or more JVM parameters. For example, `-Djava.awt.headless=true` will pass the parameter `java.awt.headless` with a value of `true`.  
  - Multiple maven goals can be specified, separated with a 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** | *(Optional: Maven 2.x and later only)* The path to your Maven project file, relative to the working sub directory specified. If this is not specified, Maven will use the pom.xml in the root of the working sub directory. |
4. If required, specify environment variables and working directory settings:

<table>
<thead>
<tr>
<th>Environment Variables</th>
<th>(Optional) 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=&quot;-Xms200m -Xmx700m&quot;).</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Working Sub Directory</th>
<th>(Optional) 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.</th>
</tr>
</thead>
</table>

5. If applicable, select The build will produce test results and choose one of the following:

<table>
<thead>
<tr>
<th>Look in the standard test results directory</th>
<th>Specify custom results directories</th>
<th>Specify the alternative directory, relative to the root directory, where test results will be created. You can use Ant-style patterns such as <em>/test-reports/</em>.xml. Bamboo requires test results to be in JUnit XML format. <strong>⚠️</strong> For jobs that use CVS, the root directory is &lt;bamboo-home&gt;/xml-data/build-dir/JOB_KEY/&lt;cvs-module&gt;.</th>
</tr>
</thead>
</table>

6. Click Save.
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.

Build 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.
MSBuild
This page describes how to configure a Bamboo task to use an MSBuild executable.

**Related pages:**
- Configuring tasks
- Editing a job

**To configure an MSBuild task:**

1. Navigate to the task configuration 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 MSBuild if creating a new task.
3. Complete the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Description</td>
<td>A description of the task, which is displayed in Bamboo.</td>
</tr>
<tr>
<td>Executable</td>
<td>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.</td>
</tr>
<tr>
<td>Project File</td>
<td>The name of the solution, project file or MSBuild project to execute, e.g. ExampleSolution.sln. You can include variables (see Using global, plan or build-specific variables).</td>
</tr>
<tr>
<td>Options</td>
<td>The MSBuild command line options that you want to include. You can also include variables (see Using global, plan or build-specific variables).</td>
</tr>
</tbody>
</table>

4. If required, specify environment variables and working directory settings:
### Environment Variables

*(Optional)* 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

*(Optional)* 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.
NAnt

This page describes how to configure a Bamboo task to use a NAnt executable.

To configure a NAnt task:

1. Navigate to the task configuration 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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Description</td>
<td>A description of the task, which is displayed in Bamboo.</td>
</tr>
<tr>
<td>Executable</td>
<td>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.</td>
</tr>
<tr>
<td>Build File</td>
<td>The relevant file name (e.g. default.build). You can include variables (see Using global, plan or build-specific variables).</td>
</tr>
<tr>
<td>Targets</td>
<td>The NAnt target that you want Bamboo to execute. for example: run. You can also include variables (see Using global, plan or build-specific variables).</td>
</tr>
<tr>
<td>Options</td>
<td>The NAnt command line options that you want to include. You can also include variables (see Using global, plan or build-specific variables).</td>
</tr>
</tbody>
</table>

4. If required, specify environment variables and working directory settings:

| Environment Variables | (Optional) 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 | (Optional) 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.
Script
This page describes how to configure a Bamboo task to use a script executable.

**Related pages:**
- Configuring tasks
- Editing a job

**To configure a script task:**
1. Navigate to the task configuration 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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Description</td>
<td>A description of the task, which is displayed in Bamboo.</td>
</tr>
</tbody>
</table>
### 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 [Using global, plan or build-specific 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 [Using global, plan or build-specific variables](#)).

4. If required, specify environment variables and working directory settings:

#### Environment Variables

*(Optional)* 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

*(Optional)* 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

This page describes how to configure a Bamboo task to use a Visual Studio (devenv.exe) executable.

**Related pages:**
- Configuring tasks
- Editing a job

**To configure a Visual Studio task:**

1. Navigate to the task configuration 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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Description</td>
<td>A description of the task, which is displayed in Bamboo.</td>
</tr>
<tr>
<td>Executable</td>
<td>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.</td>
</tr>
</tbody>
</table>
### 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 Using global, plan or build-specific variables).

### Options

Specify any Visual Studio command-line options that you want to include (e.g. /build Debug). You can also include variables (see Using global, plan or build-specific 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

(Optional) 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

(Optional) 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.
Configuring a test task
Test tasks in Bamboo parse test data, and may run tests, using a particular testing framework.

Please note:

- 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. However, you can configure a builder task to not produce test results and use a test task to parse the test data instead. For example, you may want to set up one JUnit Parser task to parse test data for a number of Maven tasks after they have executed.
- .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. NUnit Parser), if you want test results from the builder task to be parsed.

Related pages:
- Configuring a builder task

See the following pages for more information on configuring specific test tasks:
JUnit Parser
This page describes how to configure a JUnit Parser executable for a Bamboo task.

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.

An executable is a program external to Bamboo used to automate processes. 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.

Related pages:
- Configuring tasks
- Editing a job
- Configuring a test task

Configuring a JUnit Parser executable for a task
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 JUnit executable for a task:

1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired JUnit Parser task, or click Add Task and then JUnit Parser if creating a new task.
3. Update the task settings:
   - Task Description — Enter a description of the task, for display in Bamboo.
   - 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 */test-report s/.xml. 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 <bamboo-home>/xml-data/build-dir/JOB_KEY/<cvs-module>.
4. Click Save.

MBUnit Parser
This page describes how to configure a MBUnit parser executable for a Bamboo task.

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.
An executable is a program external to Bamboo used to automate processes. 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.

### Related pages:
- Configuring tasks
- Editing a job
- Configuring a test task

### Configuring a MBUnit Parser executable for a 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. MBUnit Parser), if you want test results from the builder task to be parsed.

To configure a MBUnit Parser executable for a task:

1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired MBUnit Parser task, or click Add Task and then MBUnit Parser if creating a new task.
3. Update the task settings:
   - Task Description — Enter a description of the task, for display in Bamboo.
   - 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/](http://www.mbunit.com/).
4. Click Save.

### MSTest Parser

This page describes how to configure a MSTest parser executable for a Bamboo task.

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.

An executable is a program external to Bamboo used to automate processes. 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.

### Related pages:
- Configuring tasks
- Editing a job
- Configuring a test task

### Configuring a MSTest Parser executable for a 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), if you want test results from the builder task to be parsed.

To configure a MSTest Parser executable for a task:
1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired MSTest Parser task, or click **Add Task** and then **MSTest Parser** if creating a new task.
3. Update the task settings:
   - **Task Description** — Enter a description of the task, for display in Bamboo.
   - **MBUnit 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 Runner**

This page describes how to configure a **MSTest Runner** executable for a Bamboo task. The MSTest Runner executable runs and parses tests for .NET builds.

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.

An executable is a program external to Bamboo used to automate processes. 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.

**Configuring a MSTest Runner executable for a 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), if you want test results from the builder task to be parsed.

**To configure a MSTest Runner executable for a task:**

1. Navigate to the task configuration for the job (this will be the default Job if creating a new plan).
2. Click the name of the desired MSTest Runner task, or click **Add Task** and then **MSTest Runner** if creating a new task.
3. Update the task settings:

   **Related pages:**
   - **Configuring tasks**
   - **Editing a job**
   - **Configuring a test task**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Description</td>
<td>A description of the task, for display in Bamboo.</td>
</tr>
<tr>
<td><strong>Executable</strong></td>
<td>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.</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Environment Variables</strong></td>
<td>Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS=&quot;-Xmx256m -Xms128m&quot;.</td>
</tr>
<tr>
<td><strong>Container</strong></td>
<td>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).</td>
</tr>
<tr>
<td><strong>Test Metadata</strong></td>
<td>The path to the Test Metadata file relative to the working directory. For example, &quot;MyApp\MyApp.vsmdi&quot;</td>
</tr>
<tr>
<td><strong>Result Filename</strong></td>
<td>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).</td>
</tr>
<tr>
<td><strong>Run Configuration</strong></td>
<td>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).</td>
</tr>
</tbody>
</table>
NUnit Parser
This page describes how to configure a NUnit Parser executable for a Bamboo task.

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.

An executable is a program external to Bamboo used to automate processes. 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.
Configuring a NUnit Parser executable for a 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), if you want test results from the builder task to be parsed.

To configure a NUnit Parser executable for a task:

1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired NUnit Parser task, or click Add Task and then NUnit Parser if creating a new task.
3. Update the task settings:
   - Task Description — Enter a description of the task, for display in Bamboo.
   - 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 Runner

This page describes how to configure a NUnit Runner executable for a Bamboo task.

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.

An executable is a program external to Bamboo used to automate processes. 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.

Configuring a NUnit Runner executable for a task

Before you begin:

- .NET builder tasks in Bamboo (e.g. NUnit Runner) 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.

To configure a NUnit Runner executable for a task:

1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired NUnit Runner task, or click Add Task and then NUnit Runner if creating a new task.
3. Update the task settings:

Related pages:
- Configuring tasks
- Editing a job
- Configuring a test task

Related pages:
- Configuring tasks
- Editing a job
- Configuring a test task
<table>
<thead>
<tr>
<th><strong>Setting</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task Description</strong></td>
<td>A description of the task, which gets displayed in Bamboo.</td>
</tr>
<tr>
<td><strong>Executable</strong></td>
<td>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 <strong>add other executables</strong>, if required.</td>
</tr>
<tr>
<td><strong>NUnit Test Files</strong></td>
<td>The name of an assembly (.dll), Visual Studio project (.csproj), or NUnit Test Suite (.nunit) to test. See <a href="http://www.nunit.org/">http://www.nunit.org/</a>.</td>
</tr>
<tr>
<td><strong>Result Filename</strong></td>
<td>The name to be used for the XML results file.</td>
</tr>
<tr>
<td><strong>Tests to Run</strong></td>
<td>The name of the test case, test fixture or namespace to run.</td>
</tr>
<tr>
<td><strong>Test Categories to Include</strong></td>
<td>Specify one or more test categories, separated by commas, to be included in the test run.</td>
</tr>
<tr>
<td><strong>Test Categories to Exclude</strong></td>
<td>Specify one or more test categories, separated by commas, to be excluded from the test run. Exclusions take precedence over inclusions.</td>
</tr>
<tr>
<td><strong>Command Line Options</strong></td>
<td>Specify any command line options or switches you wish to include when running NUnit.</td>
</tr>
</tbody>
</table>

4. Click **Save**.

For more information on NUnit, see [http://www.nunit.org/](http://www.nunit.org/).
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.

An executable is a program external to Bamboo used to automate processes. 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.

**PHPUnit**

This page describes how to configure a PHPUnit executable for a Bamboo task.

When a new plan is created, you can configure the tasks for the plan's default job.

An executable is a program external to Bamboo used to automate processes. 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.
Configuring a PHPUnit executable for a task

Before you begin:

- To use this executable, you will need to install PHPUnit and reference the path to your PHP command-line interpreter, (e.g. /usr/bin/phpunit on Ubuntu).

To configure a PHPUnit executable for a task:

1. Navigate to the task configuration for the job (this will be the default job if creating a new plan).
2. Click the name of the desired command task, or click Add Task and then a PHPUnit option (e.g. 'PHPUnit') if creating a new task.
3. Update the task settings:
   - Task Description --- Enter a description of the task, for display in Bamboo.
   - 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.
4. If required, update the system environment variables and working directory settings:

<table>
<thead>
<tr>
<th>Environment Variables</th>
<th>(Optional) 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=&quot;-Xms200m -Xmx700m&quot;).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working Sub Directory</td>
<td>(Optional) 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.</td>
</tr>
</tbody>
</table>

5. Update the following build parameters:
   - 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 — Enter the location of the file to record JUnit 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 — Enter the location of the directory to record HTML coverage data.
6. Click **Save**.

**Using global, plan or build-specific 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).
- **System variables** also apply across your entire Bamboo instance and inherit their values from system or environment variables of the same name.

### On this page:
- Specifying variables
- Using variables
- Examples of variables usage
- Specifying capabilities as variables

### Related pages:
- Defining global variables
- Defining plan variables
- Triggering a plan build manually
- Configuring plugins

**Specifying variables**

**Global variables**

See [Defining global variables](#) for information on defining global variables.

The usage format for all global variables is:

\[${\text{bamboo.globalVarName}}\]

**Plan variables**

See [Defining plan variables](#) for information on defining plan variables. You can override a plan variable for a build, if you have triggered the build manually. For details, see [Triggering a plan build manually](#).

The usage format for all plan variables is:

\[${\text{bamboo.varName}}\]

**Build-specific variables**

The following build-specific variables are also available by default:
<table>
<thead>
<tr>
<th>Build-specific variable</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildKey</td>
<td>Bamboo property</td>
<td>The plan key for the current job, e.g. BAM-MAIN-JOBX</td>
</tr>
<tr>
<td>buildNumber</td>
<td>Bamboo property</td>
<td>The Bamboo build number, e.g. 1 23</td>
</tr>
<tr>
<td>buildPlanName</td>
<td>Bamboo property</td>
<td>The Bamboo plan name e.g. Some Project name – Some plan name</td>
</tr>
<tr>
<td>buildTimeStamp</td>
<td>Bamboo property</td>
<td>The time when build was started in ISO 8601 format e.g. 2010-01-01T01:00:00.000+0 1:00</td>
</tr>
<tr>
<td>buildForceCleanCheckout</td>
<td>Bamboo property</td>
<td>Whether the &quot;Force Clean Build&quot; option was used, values: true/false</td>
</tr>
<tr>
<td>build.working.directory</td>
<td>Bamboo property</td>
<td>The working directory that the build is being executed on</td>
</tr>
<tr>
<td>manualBuildTriggerReason.usern ame</td>
<td>Bamboo property</td>
<td>The user who triggered the manual build</td>
</tr>
<tr>
<td>repository.revision.number</td>
<td>Plugin</td>
<td>The revision number</td>
</tr>
<tr>
<td>repository.previous.revision.numb er</td>
<td>Plugin</td>
<td>The previous revision number (might not exist if for example is initial build)</td>
</tr>
<tr>
<td>custom.svn.revision.number</td>
<td>Plugin</td>
<td>(For Subversion only) The revision number</td>
</tr>
<tr>
<td>custom.svn.lastchange.revision.numb er</td>
<td>Plugin</td>
<td>(For Subversion only) The last changed revision number</td>
</tr>
<tr>
<td>custom.svn.username</td>
<td>Plugin</td>
<td>(For Subversion only) User name used for repository authentication</td>
</tr>
<tr>
<td>repository.svn.repositoryUrl</td>
<td>(For Subversion only) The repository url</td>
<td></td>
</tr>
<tr>
<td>custom.cvs.last.update.time</td>
<td>Plugin</td>
<td>(For CVS only) The last updated timestamp</td>
</tr>
<tr>
<td>custom.cvs.last.update.time.label</td>
<td>Plugin</td>
<td>(For CVS only) 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 '_'</td>
</tr>
</tbody>
</table>
### System variables

- System variables also apply across your entire Bamboo instance and inherit their values from system or environment variables of the same name.

The usage format for all build-specific variables is:

```
${bamboo.varName}
```

### 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.

### Using variables

Variables can be used in the following fields of your build plan:

<table>
<thead>
<tr>
<th>Field</th>
<th>Global</th>
<th>Build-specific</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal (for Maven builders only) — see Configuring tasks</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
### Build File (for Ant and NAnt builders only)
- see Configuring tasks

### Target (for Ant and NAnt builders only)
- see Configuring tasks

### Options (for NAnt builders only)
- see Configuring tasks

### Script (for Scripts only)
- see Configuring tasks

### Argument (for Scripts and Custom Commands only)
- see Configuring tasks

### Environment Variables
- see Configuring tasks

### Repository URL (for Subversion repositories only)
- see Specifying the source repository

### Web Repository URL (for Subversion, CVS and Perforce repositories)
- see Specifying the source repository

### CVS Root (for CVS repositories only)
- see Specifying the source repository

### Branch name (for CVS repositories only)
- see Specifying the source repository

#### Examples of variables usage

**Maven example**

For example, you may want your Maven 2 version to be determined by Bamboo. In Maven 2 `pom.xml` you may have:
You can then specify the following in the **Goal** field of your build plan:

```xml
<groupId>com.atlassian.boo</groupId>
<artifactId>boo-test</artifactId>
<packaging>jar</packaging>
<version>1.1.1102-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 example**

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,
### 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,

<table>
<thead>
<tr>
<th>Capability Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom</td>
<td>${bamboo.capability.&lt;capability_key&gt;}</td>
</tr>
<tr>
<td>JDK</td>
<td>${bamboo.capability.system.jdk.&lt;jdk_label&gt;}</td>
</tr>
</tbody>
</table>
| Builder         | ${bamboo.capability.system.builder.<builder_type>.<builder_label>}
|                 | e.g. ${bamboo.capability.system.build.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.

#### 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 Using global, plan or build-specific 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 Triggering a plan build manually.

### Related pages:
- Using global, plan or build-specific variables
- Defining plan variables
- Triggering 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

### 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 Configuring jobs and tasks.

- Viewing a plan's build information
- Triggering builds
- Setting up plan build dependencies
Viewing a plan's build information

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

Plans:

- are organised into one or more stages;
- have one or more jobs in each stage;
- contain a single 'Default job' in a single stage, straight after creating a new plan;
- define default settings for what gets built by jobs in the plan (i.e. the 'default source repository');
- define how the plan's build is triggered;
- define who will be notified of the job's build result;
- define who has permission to view and perform various actions on the plan and its jobs.

Every plan belongs to a project.

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

To view a plan's details:

1. 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.
2. Click a tab to view information about the plan:

<table>
<thead>
<tr>
<th>Tab</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan Summary</td>
<td>Information about the plan, as shown in the diagram below.</td>
</tr>
<tr>
<td>Branches</td>
<td>The branch plans for this plan.</td>
</tr>
<tr>
<td>Recent Failures</td>
<td>Information about recent failures of the plan, including the builds that failed, links to the build results, time taken to fix, etc.</td>
</tr>
<tr>
<td>History</td>
<td>The full history of builds of the plan.</td>
</tr>
<tr>
<td>Quarantined Tests</td>
<td>Failing test's results that have been disconnected from the build results.</td>
</tr>
<tr>
<td>Issues</td>
<td>View the JIRA issues linked to builds of your plan. (Only displayed if your administrator has integrate Bamboo with JIRA.)</td>
</tr>
<tr>
<td>Configuration</td>
<td>View and edit the plan's configuration. (Only displayed if you are an administrator for the plan.)</td>
</tr>
</tbody>
</table>
3. Use the **Actions** menu to access common functions for the plan, such as **Run Plan, Disable Plan** and **Configure Plan.** (This menu is only displayed if you are an **administrator** for the plan.)

**Diagram: Plan Summary (annotated)**

---

**Triggering builds**

There are several methods by which Bamboo can 'trigger' (i.e. begin executing) a plan's build:

**Build Strategy**

- **Code updated** — ensures that a plan build only occurs when something changes in the plan's source repository (which may affect the outcome of a plan build) i.e. whenever one or more authors check in code.
- **Scheduled build** — can allow a team to structure the day according to a predictable schedule. Scheduled builds are run regardless of whether or not any code changes have occurred i.e. at scheduled times or specified time intervals.
- **Manual build** — allows you to ensure that builds are *only* triggered manually (or by the successful builds of other plans on which this plan is dependent).
- **Initial clean build** — i.e. immediately after a new plan has been created.

**Build Dependency**

- **Dependency** — a build can be triggered whenever another plan's build is successfully completed.

The trigger method for a plan's build is listed in the 'Reason' column on the **dashboard**.

---

**On this page:**

- [Considerations for choosing a build strategy](#)
- [Build dependencies and build strategies](#)
- [Conditional build triggers](#)
Considerations for choosing a build strategy

This table lists Bamboo’s available ‘build strategies’, which determine how the execution of a plan (i.e. a build) is triggered. You set a Build Strategy option on the Plan Details tab of the plan configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
</tr>
</thead>
</table>
| Code is updated                             | Polling the Repository for changes                                                                                                              | • This is a ‘pull strategy’.  
• This is the simplest option.  
• But it does mean that your SCM must service a ‘check out’ or ‘update’ command whenever it is polled, even if no code has changed in the repository. |
|                                            | Bamboo will 'poll' the source code repository for changes based on either of the following:                                                       |                                                                                                          |
|                                            | • set intervals (periodically)  
• a schedule.                                                                                                                                     |                                                                                                          |
|                                            | If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.                                              |                                                                                                          |
|                                            | See Polling the repository for changes.                                                                                                         |                                                                                                          |
| Repository triggers the build when changes are committed | Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan. | • This is a ‘push strategy’.  
• This option minimises server load as message events are sent only when code changes to this repository are committed.  
• But you must configure your source code management system to send message events to Bamboo about code changes in this repository. |
|                                            | See Repository triggers the build when changes are committed.                                                                                  |                                                                                                          |
### Scheduled

<table>
<thead>
<tr>
<th>Scheduled</th>
<th>Cron Based Scheduling</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cron Based Scheduling</td>
<td>Bamboo will trigger a build of this plan based on a cron expression. See <a href="#">Cron-based scheduling</a>.</td>
<td>Manual</td>
</tr>
<tr>
<td>Single daily build</td>
<td>Bamboo will trigger a build of this plan once per day at a specified time. See <a href="#">Single daily build</a>.</td>
<td>Manual</td>
</tr>
<tr>
<td>Manual</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a <strong>build dependency</strong>. See <a href="#">Triggering a plan build manually</a>.</td>
<td>Manual</td>
</tr>
</tbody>
</table>

- Allows you to run builds based on a schedule.
- 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 repository.
- 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 repository.
- This option is suitable if a build of this plan will fail, perhaps due to source code problems or failing tests.
- This frees up Bamboo agents to build other plans which are less likely to fail.

### Build dependencies and build strategies

Build dependencies are "parent-child" relationships between plan builds, which can be configured so that a series of plan's builds are triggered. See [Setting up plan build dependencies](#).

The page Setting up Build Dependencies does not exist.

### 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 that must build successfully before this plan can be built. See also [Setting up plan build dependencies](#).

**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 **Plan Details** tab.
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.

<table>
<thead>
<tr>
<th>Trigger Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Trigger Conditions" /></td>
</tr>
</tbody>
</table>

**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, a build of your plan is triggered.

**On this page:**

- [Periodic polling for repository changes](#)
- [Scheduled polling for repository changes](#)

**Related pages:**

- Triggering a plan build when code is updated
- Triggering a plan build based on a schedule

**Periodic polling for repository changes**

**To trigger a plan’s build periodically:**

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 Plan Details tab.
4. Choose **Build Strategy > Polling the Repository for changes**.
5. Select **Periodically** from the ‘Polling Strategy’ options.
6. Use **Polling Frequency** to specify how often (in seconds) Bamboo should check the repository for changes.
7. Click **Save**.

**Screenshot: Polling the repository for changes**
Scheduled polling for repository changes

You can use the Schedule Editor to set up a polling schedule for your repository. Note, this is a schedule for polling your repository: a plan build will be be triggered only if there are source code changes. See also how to schedule for directly triggering plans.

The schedule can be daily (times per day), weekly (days per week), monthly (days per month) or based on a Cron expression.

To trigger a plan's build based on a schedule:

1. Navigate to the repository settings for desired Plan, as described on Specifying the source repository.
2. From the Build Strategy list, choose Polling the Repository for changes.
3. Choose Scheduled from the Polling Strategy options. The Schedule field will display the last configured schedule, e.g. 'Daily at 5:00 am'.
4. Click the edit icon (📝) next to the Schedule field to display the Schedule Editor.
5. Use the Schedule Editor to set up the polling schedule.
   - Note that 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, 0 0 1 ? * 1#2. For information on Cron expressions, see this FAQ: How do I construct a cron expression in Bamboo.
6. Click Save.

Screenshots: Scheduling polling for changes
1. Daily

2. Week

Repository triggers the build when changes are committed

"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 trigger a build on code check in.

On this page:

1. Configuring your source repository
2. Configuring Bamboo to trigger a build on code check in

Related pages:

- Triggering a plan build when code is updated
- Polling the repository for changes

1. Configuring your source repository

To configure 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...

Edit the Subversion repository's hooks/post-commit trigger file with something like:
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 trigger a build on code check in

Before you begin:

- Triggering a build when there is no update — Bamboo will ignore the build triggers, if the local working copy and the repository copy have the same revision numbers. When testing your build triggers, please check that the local working copy is not the latest version - in which case, no further action will be taken.

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 Plan Details tab.
4. Choose Build Strategy > Repository triggers the build when changes are committed.
5. Only enter an IP address in Trigger IP Address if you want Bamboo to receive post-commit notifications 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 the Trigger IP Address field.

6. Click Save.

Screenshot: Build Strategy – repository triggers the build when changes are committed

Cron-based scheduling

"Cron-based scheduling" allows you to run builds based on a schedule. This schedule can be configured using the Schedule Editor or by using a cron expression.

The schedule can be daily (times per day), weekly (days per week), monthly (days per month) or based on a cron expression.

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 Plan Details tab.
4. Choose Cron Based Scheduling from the Build Strategy list (under ‘Build Strategy’).
5. Click the edit icon ( ) next to the current schedule to display the Schedule Editor.
6. 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.

7. Click **Save**.

**Screenshots: Schedule Editor options**

<table>
<thead>
<tr>
<th>Daily</th>
<th>Days per Week</th>
<th>Days per Month</th>
<th>Cron Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval: once per day</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>At: 12:00</td>
<td></td>
<td></td>
<td>00 <strong>a</strong> 0 0 ? * *</td>
</tr>
<tr>
<td>Sunday</td>
<td></td>
<td>The 1st day of every month</td>
<td></td>
</tr>
<tr>
<td>Monday</td>
<td></td>
<td>The first Sunday of every month</td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Single daily build**

A "Single daily build" runs at a time of your choice. This is particularly suitable for builds that take a long time to complete.

**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 **Plan Details** tab.
4. Choose **Build Strategy** > **Single daily build** (see screenshot below).
5. Specify the time of day at which the build should run in **Build Time**. Use hh:mm format, with a 24-hour clock.
6. Click **Save**.

**Screenshot: Build Strategy – Single daily build**
Triggering a plan build manually
Bamboo allows you to ensure that builds are only triggered manually or by parent builds of dependent builds.

You can also run a customised build, which allows you to do the following:

- 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.

If you want to trigger builds automatically rather than manually, you can trigger builds based on code changes or based on a schedule.

Starting a plan build manually
To start a plan build manually:

1. Locate the relevant plan on the Dashboard.
2. Click the Run icon.

On this page:
- Starting a plan build manually
- Starting a parameterised plan build
- Starting a plan build with manual stages
- Configuring a plan's build to be triggered only manually or as a dependency

Related pages:
- Stopping an active build
- Defining plan variables
- Triggering builds

Starting a parameterised plan build
If you trigger a plan build manually, you can override any global variables or plan variables by substituting your own parameters.

Before you begin:

- You cannot run a parameterised plan build if it is triggered automatically.
- You can only run a parameterised plan build if you have defined global variables for Bamboo or plan variables for your plan.
To start a parameterised plan build:

1. Navigate to the Plan Summary (not Plan configuration) for the desired plan.
2. Choose Run Plan > Run Customised Plan.
3. Override the desired global and plan variables in the appropriate tab of the 'Run Customised Build' dialog. Note, the 'Plan Variables' and 'Global Variables' tabs will not appear if you do not have such variables defined.
4. Click Run.

Starting a plan build with manual stages

If you manually trigger a plan build that has manual stages, you can select the manual stages to run.

To start a plan build, manually selecting the stages to run:

1. Navigate to the Plan Summary (not Plan configuration) for the desired plan.
2. Choose Run Plan > Run Customised Plan.
3. Click the Stages tab in the 'Run Customised Build' dialog, and select the desired manual stages to run.
4. Click Run.
Configuring a plan's build to be triggered only manually or as a dependency

You can specify that a plan should only ever be built manually or triggered by other builds. This is useful for:

- Broken builds — If a build is broken, you may want to temporarily specify Manual & dependent builds only. This means that a failing build will not be triggered frequently and hence will not take up time and processing power when other builds could be running.
- Dependent builds — If you specify that a build should run when another build successfully finishes, you may want to prevent it from running at other times. You can achieve this by specifying Manual & dependent builds only.

To allow a plan's build to be triggered only manually or as a dependency:

1. Navigate to the repository settings for the desired plan, as described on Specifying the source repository.
2. Select Manual & dependent builds only from the Build Strategy list (see screenshot below).
3. Click Save.

Screenshot: Plan Configuration — Build Strategy: Manual builds only

Triggering a plan build when code is updated

"Triggering a Plan's build when code is updated" ensures that a Plan build only occurs when something changes in the Plan's source repository (which may affect the outcome of a Plan build). There are two ways to trigger a Plan's build when code is updated:

- **Pull strategy** — See Polling the repository for changes.
- **Push strategy** — "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'). See Repository triggers the build when changes are committed.

You can also trigger builds based on a schedule or manually, rather than when code is updated.

**Triggering a plan build based on a schedule**

"Triggering a Plan build based on a schedule" can allow a team to structure the day according to a predictable schedule. Note that scheduled builds are run regardless of whether or not any code changes have occurred. There are two ways to schedule a build:

- **Single Daily Build** — A "single daily build" runs at a time of your choice. This is particularly suitable for builds that take a long time to complete. See Single daily build.
- **Cron-Based Scheduling** — The page Cron Based Scheduling does not exist. See Cron-based scheduling.

You can also trigger builds based on code changes or manually, rather than on a schedule.

**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**. In turn, every time a **Acme – Plugin** plan build completes successfully, you may want to run builds for some additional plans (e.g. **Acme – Functional Tests** and **Acme – Unit Tests**). In this case, **Acme – Plugin** is a parent of the **Acme – Functional Tests** and **Acme – Unit Tests** plans, as well as being a child of the **Acme – Core** plan.

**On this page:**

- Manual dependency management
- Automatic dependency management with Maven 2
- Dependency blocking
- Notes

**Manual dependency management**

To trigger a build when another build finishes:

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. Locate the 'Manual Dependency Management' section — all plans in your Bamboo system (other than the one you are configuring) are listed in the 'Child plans' and 'Parent plans' sections. Select plans in the following subsections:
   - Child plans — Plans whose builds you want triggered as a result of successful completion of the plan you are configuring.
   - Parent plans — Plans whose successful completion will trigger a build of the plan you are configuring. Any other plans in your Bamboo system that specify build dependencies, with the plan you are configuring, will have their check boxes selected.
5. Click **Save**.
Automatic dependency management with Maven 2

Automatic Dependency Management is a feature for users who use Maven 2 and wish for their Parent and Child dependencies to be setup 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.
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.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Override Project File</td>
<td>Optional. The location relative to the working directory or sub-working directory where the project file (pom.xml) is located.</td>
</tr>
<tr>
<td>Working Sub Directory</td>
<td>Optional. The sub directory from which the Task should look for the project file (pom.xml)</td>
</tr>
<tr>
<td>Alternate location of settings.xml</td>
<td>Optional. Specify an alternate settings.xml to be used if the Task needs to resolve dependencies from specific Maven repositories.</td>
</tr>
<tr>
<td>Path to Maven local repository</td>
<td>Optional. Specify a full path to a local Maven repository for the Task to use to resolve dependencies.</td>
</tr>
</tbody>
</table>

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 uses a build strategy based on source code updates.

Notes

Build dependencies work together with the build strategies 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 configure a build dependency 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, you can do this by specifying Manual as the build strategy for your plan. See Triggering 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 Triggering a plan build manually.

## Dependency blocking strategies

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.

⚠️ Dependency blocking only works when the plan uses a build strategy based on source code updates (i.e. the Polling the Repository for changes or Repository triggers the build when changes are committed build strategies). This feature will not work when a plan uses a build strategy based on a schedule or triggered via a parent build (when there are multiple parent plan builds in progress).

The three dependency blocking strategies are explained below:

- **Do not block** — If a plan build with this dependency blocking strategy is triggered by a source code update, the plan build will run whenever it is triggered by a source code update, regardless of whether or not it has parent plan build dependencies.

- **Block build if parent builds are queued or in progress** — If a plan build with this dependency blocking strategy is triggered by a source code update, Bamboo will check whether its parent plans’ jobs are building or are waiting in the build queue.
  - If so, the plan’s build will be blocked.
  - If not, the plan’s build will run.

- **Block build if parent plans have unbuilt changes** — If a plan build with this dependency blocking strategy is triggered by a code update, Bamboo will check whether its parent plans are building, queued to build or have changes.
  - If so, the plan build will be blocked.
  - If not, Bamboo will check if any of the parent plan’s builds have changes. If there are parent plan builds with changes, those plans will be triggered and your plan’s build will be blocked. If there are no parent plan builds with changes, your plan’s build will run.

These dependence blocking strategies are illustrated in the flowchart below:
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'.

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.

Related topics

For more information on working with build results, see the following topics:

- Viewing a build result
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.

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.
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.

Screenshot: Build Result Summary

- The Summary tab (shown in the screenshot above) displays a snapshot of the build result. You can also click the Comment button to add a comment.
  - The ✔️ icon represents a successful build.
  - The 🔄 icon represents a build that was not completed. For example, it may have been stopped manually.
  - The 🚫 icon represents a failed build. If a build has failed, you can run the entire build again or rerun just the failed Stage.
- Click the Tests tab to view the build's test results.
- Click the Changes tab to view the code changes that triggered this build (if applicable).
- Click the Artifacts tab to view any artifacts relating to this build.
- Click the Logs tab to view a complete build log.
- Click the Metadata tab to view any metadata that relates to this build result.
- Depending on how your Bamboo administrator has configured the system, the following additional tabs may be available:
  - Click the Issues tab to view any JIRA issues that relate to this build result (if applicable).
  - Click the Clover tab to view the Clover code-coverage that relate to this build result (if applicable).

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.

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.

Before you begin:

- You must be logged in to Bamboo before you can comment on 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.

Screenshot: Adding a comment to a build result

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

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 Configuring 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
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'.

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.

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.
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.

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.

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 the side-by-side diff view to analyze the change and then open the correct files in your IDE.

To view the code changes that triggered a particular build result:

1. Navigate to the build results for the plan, as described in Viewing a build result, and click the desired build result.
2. 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:

1. Go to the build result. See Viewing a build result for instructions.
2. 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
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.

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 Log tab.
   - Click View for the desired log.
   - Click Download to download a text file of the log.

Screenshot: Build Log
Viewing the metadata for a build result

If your source-code repository provides metadata (i.e. key-value properties that are used to describe your build) for your build results, Bamboo will display it.

**Related pages:**

- Working with build results

**To view the metadata for a build result:**

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 Metadata tab.

Screenshot: Metadata for a Build Result
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 job, as described in Configuring jobs and tasks.
2. Click the Tests tab.
3. Click the sub-tabs to filter the rest statistics (see screenshots below).
   - To view a test's history, click the test name.

*Screenshot: Test statistics for a job*
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.
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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build results</td>
<td>All build results data (including artifacts and build logs) are deleted.</td>
</tr>
<tr>
<td>Artifacts</td>
<td>Only user-defined artifacts are deleted from the build results.</td>
</tr>
<tr>
<td>Build logs</td>
<td>Only build logs are deleted from the build results.</td>
</tr>
<tr>
<td>Expiry period</td>
<td>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.</td>
</tr>
<tr>
<td>Minimum builds to keep</td>
<td>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.</td>
</tr>
<tr>
<td>Labels to keep</td>
<td>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.</td>
</tr>
</tbody>
</table>

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
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 ‘999999999 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

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 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

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:

1. Navigate to the build results for the Plan/Job, as described in Viewing a build result, and click the desired build result.
2. Click the Tests tab.
3. Click the name of the test in which you are interested. The test's latest result will be displayed.
4. Click View test case across builds. The 'Test History' will be displayed, as shown below.

Screenshot: Test History

![Test History Screenshot]

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.

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 Triggering a plan build manually.

On this page:
- Stopping an active plan build
- Stopping an active job build

Related pages:
- Triggering 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).

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
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.

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 reports**

## 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](#)          |

### 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.
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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>All plans</td>
<td>Wallboard &gt; All Plans</td>
<td>Alternatively, use the following URL in your browser, replacing 'bambooserver' with the real name of your Bamboo server: <a href="http://bambooserver:8080/bamboo/telemetry.action">http://bambooserver:8080/bamboo/telemetry.action</a></td>
</tr>
<tr>
<td>Favourite plans</td>
<td>Wallboard &gt; Favourite Plans</td>
<td>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: <a href="http://bambooserver:8080/bamboo/telemetry.action?filter=favourites">http://bambooserver:8080/bamboo/telemetry.action?filter=favourites</a></td>
</tr>
<tr>
<td>Filtered plans</td>
<td>Wallboard &gt; Filtered Plans</td>
<td>You need to have set up a plan filter first. See Using the Bamboo Dashboard.</td>
</tr>
</tbody>
</table>

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.
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:

- 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.

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

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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>Select the event type you want to be notified about. Refer to the list of events (<a href="#">below</a>) for details.</td>
</tr>
<tr>
<td>Recipient Type</td>
<td><strong>User</strong> — Enter the username of the appropriate Bamboo user, or click the icon to select from a list of users.</td>
</tr>
<tr>
<td></td>
<td><strong>Hipchat</strong> — Enter the Hipchat API Token and Room Name.</td>
</tr>
<tr>
<td></td>
<td><strong>Group</strong> — Enter the name of the appropriate Bamboo group(s).</td>
</tr>
<tr>
<td>Email Address</td>
<td>You can use email to send notifications to a person who is not a Bamboo user. Type the appropriate email address. Note that:</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td>IM Address</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>• If you specify a broadcast address (e.g. <code>project-x@broadcast.chat.mycompany.com</code>), Bamboo will not know the context of related IM responses.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td>Committers</td>
<td>The Bamboo users who have committed code to a particular build since build was last checked out by Bamboo.</td>
</tr>
<tr>
<td>Watchers</td>
<td>The Bamboo users who have marked this plan as one of their favourites.</td>
</tr>
</tbody>
</table>

4. Click Add, then configure further notifications if required.
5. Click Save when you have finished.

*Screenshot: Plan build notifications*
Notification events

<table>
<thead>
<tr>
<th>Plan Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All Builds Completed</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Failed Builds And First Successful</strong></td>
<td>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 and the previous plan build failed).  This notification is generally suitable for the majority of plans.</td>
</tr>
<tr>
<td><strong>Change of Build Status</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>After X Failed Builds</strong></td>
<td>This notification allows you to specify the <strong>Number Of Failures</strong> (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.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Job Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Jobs Completed</td>
</tr>
</tbody>
</table>

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.

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 and the previous job build failed).

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.

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:**

1. Navigate to the configuration for the desired Plan, as described on Editing a plan's configuration.
2. Click the **Notifications** tab.
3. 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:

1. Configure Bamboo to send SMTP email (see below).
2. 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:**

1. Click **Administration** in the top navigation bar.
2. Click **Mail Server** in the left navigation column (under 'Communication'). This will display the ‘Mail Server Details’ page (see screenshot below).

3. Edit the mail server settings as necessary:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>A display-name for the email address e.g. ‘SMTP Server’</td>
</tr>
<tr>
<td><strong>From Address</strong></td>
<td>The email address from which Bamboo notifications will be sent.</td>
</tr>
</tbody>
</table>
| **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*
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:
3. Ensure that the files mail-X.X.jar and activation-X.X.jar exist only in the apache-tomcat-xx 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**.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>The address of your IM server (for example, 'chat.atlassian.com').</td>
</tr>
<tr>
<td>Port</td>
<td>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).</td>
</tr>
<tr>
<td>Username</td>
<td>The login name of the IM account from which Bamboo notifications will be sent.</td>
</tr>
<tr>
<td>Change password</td>
<td>Select this if you have specified a username different from the currently logged-in user.</td>
</tr>
<tr>
<td>Resource</td>
<td>An identifying name for the connection if multiple clients use the same jabber account.</td>
</tr>
<tr>
<td>Requires a TLS/SSL connection</td>
<td>Select this if your IM server uses SSL.</td>
</tr>
<tr>
<td>Force legacy SSL</td>
<td></td>
</tr>
<tr>
<td>Test Recipient Address</td>
<td>You can test this configuration by entering an address and clicking <strong>Test</strong>.</td>
</tr>
</tbody>
</table>

4. Click **Save**.

*Screenshot: Instant Messaging server details*
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](http://code.google.com/apis/talk/open_communications.html)

To configure Bamboo to use Google Talk for Instant Messaging:

1. Click Administration in the top menu bar.
2. Click IM Server in the left navigation panel (under 'Communication').
3. Click Edit. Modify the settings as required.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
</table>

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
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. atlassianbamboo@gmail.com NOT atlassianbamboo.
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](http://freemarker.org/).

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.

**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](http://docs.atlassian.com/bamboo/en/content.html), 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](http://freemarker.org/) below for tips on updating templates.
4. Save your changes to the template. Your updated template will be used in the next notification that the template applies to. You do not have to restart your Bamboo server.
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.

<table>
<thead>
<tr>
<th>System Property</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamboo.notifications.logLinesToInclude</td>
<td>Specifies the number of log lines to include in email notifications that display log information.</td>
<td>100</td>
</tr>
</tbody>
</table>

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.

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.
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
Documentation for Bamboo 4.0

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
Provide test error details

```groovy
[#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
```

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
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.

<table>
<thead>
<tr>
<th>RSS feed scope</th>
<th>Options</th>
<th>Set up</th>
</tr>
</thead>
<tbody>
<tr>
<td>All plans</td>
<td>• All build results</td>
<td>1. Go to the Dashboard’s All Plans tab.</td>
</tr>
<tr>
<td></td>
<td>• Failed build results</td>
<td>2. Locate the RSS icon at the bottom of the screen.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Right-click either all builds or all failed builds, and copy its URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Paste the URL into your RSS reader.</td>
</tr>
<tr>
<td>A specific plan</td>
<td>• All build results</td>
<td>1. Go to the plan.</td>
</tr>
<tr>
<td></td>
<td>• Failed build results</td>
<td>2. Locate the RSS icon at the bottom of the screen.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Right-click either all builds or all failed builds, and copy its URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Paste the URL into your RSS reader.</td>
</tr>
</tbody>
</table>
### Build results with a particular label

1. Go to the [Dashboard](#).
2. Go to any plan that has a label (this may involve trial and error).
3. Click on any label, near the top of the screen.
4. Click [All Labels](#).
5. Click the label of interest.
6. Locate the RSS icon at the bottom of the screen.
7. Right-click [Feed for builds labelled](#) and copy its URL.
8. Paste the URL into your RSS reader.

### Reporting

You are able to get reports about various kinds of activity in Bamboo:

<table>
<thead>
<tr>
<th>Summary statistics for all users</th>
<th>A list of summary build statistics for all Bamboo users, showing such things as the number of builds triggered, broken and fixed.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See <a href="#">Viewing build statistics for all users</a>.</td>
</tr>
<tr>
<td>Build results for an author</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>See <a href="#">Viewing build results for an author</a>.</td>
</tr>
<tr>
<td>Comparison charts for authors</td>
<td>Create comparison charts of build activity for selected authors.</td>
</tr>
<tr>
<td></td>
<td>See <a href="#">Generating reports on selected authors</a>.</td>
</tr>
<tr>
<td>Comparison charts for plans</td>
<td>Create comparison charts of build results for selected plans.</td>
</tr>
<tr>
<td></td>
<td>See <a href="#">Generating reports across multiple plans</a>.</td>
</tr>
<tr>
<td>Clover code-coverage reports</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See <a href="#">Viewing the Clover code-coverage for a job</a>.</td>
</tr>
<tr>
<td></td>
<td>See <a href="#">Viewing the Clover code-coverage for a build</a>.</td>
</tr>
</tbody>
</table>

### 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.
Viewing build results for an author

An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user.

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Triggered</th>
<th>Failed</th>
<th>% Failed</th>
<th>Broken</th>
<th>Fixed</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adrian Decco</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Adrian Hempel</td>
<td>589</td>
<td>156</td>
<td>26%</td>
<td>60</td>
<td>56</td>
<td>-4</td>
</tr>
<tr>
<td>Agnes Ro</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alex Wei</td>
<td>1</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Andrew Lynch</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anna Butfield</td>
<td>3</td>
<td>2</td>
<td>67%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Arun Bhatia</td>
<td>37</td>
<td>16</td>
<td>43%</td>
<td>5</td>
<td>2</td>
<td>-3</td>
</tr>
<tr>
<td>Belinda Teh</td>
<td>396</td>
<td>131</td>
<td>33%</td>
<td>29</td>
<td>21</td>
<td>-8</td>
</tr>
<tr>
<td>Ben Woskow</td>
<td>162</td>
<td>91</td>
<td>50%</td>
<td>17</td>
<td>7</td>
<td>-10</td>
</tr>
</tbody>
</table>

Related pages:
- Viewing build statistics for all users
Generating reports on selected authors

An author is any person who contributes to a build by checking-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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>Choose from the available reports, described below. Additional reports may have been configured by your Bamboo administrator.</td>
</tr>
<tr>
<td>Authors</td>
<td>Choose the authors on whom you want to report. Use the &lt;Ctrl&gt; key to select multiple authors.</td>
</tr>
<tr>
<td>Group By</td>
<td>Choose the time scale for the horizontal axis.</td>
</tr>
</tbody>
</table>

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**

![Build Activity Chart]

**Number of build failures**

![Number of Build Failures Chart]

**Number of builds broken**

![Number of Builds Broken Chart]
Number of builds fixed

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>Choose from the available reports, shown below. Additional reports may have been configured by your Bamboo administrator.</td>
</tr>
<tr>
<td>Build plans</td>
<td>Choose the plans on which you want to report. You can use the <code>&lt;Ctrl&gt;</code> key to select multiple plans.</td>
</tr>
<tr>
<td>Group By</td>
<td>Choose the time scale for the horizontal axis.</td>
</tr>
<tr>
<td>Date Filter</td>
<td>Choose the time period on which to report. Use Select Range to set a custom range.</td>
</tr>
</tbody>
</table>

3. Click Submit.

On this page:

- Generating plan reports
- Plan report types
Plan report types

Some of the standard plan report types are illustrated below.

**Build activity**

How many builds are triggered in a given time period? This indicates the level of activity for the plan.

**Build duration**
Build Duration

The report shows how long your build takes over time. Is it getting slower or faster?

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.

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.

<table>
<thead>
<tr>
<th>Chart</th>
<th>Data Table</th>
<th>Builds</th>
</tr>
</thead>
</table>

- **Duration**
  - 8h 20m to 2d 20m

**Graph:**
- **Blue Line:** Bamboo Main - Extras
- **Red Line:** Bamboo Main - CI Tests
- **Green Line:** Bamboo Main - Stable Extras
- **Yellow Line:** Bamboo Main - Stable CI Tests

**Dates:**
- 11-Feb to 15-Feb

### 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.

<table>
<thead>
<tr>
<th>Chart</th>
<th>Data Table</th>
<th>Builds</th>
</tr>
</thead>
</table>

- **# Tests**
  - 0 to 300

**Graph:**
- **Blue Line:** OpenID - Default
- **Red Line:** Sitemesh - Unit Tests
- **Green Line:** Sitemesh - Resin 3.0.19
- **Yellow Line:** Sitemesh - Tomcat 5.0.30
- **Blue Line:** Sitemesh - Tomcat 5.5.17
- **Blue Line:** Apache Felix - Default

**Dates:**
- Jan-2007 to Jul-2009

### Number of build failures

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
Number of Build Failures
How many builds are being broken? A high value indicates a relatively unstable plan that tends to be broken often.

Clover lines of code

Clover Lines of Code
Provides an indication of the size of the code base for the build.

Clover code coverage
Viewing the Clover code-coverage for a job

If you use Atlassian’s Clover and your job specifies a Clover directory (see Enabling the Clover plugin), you will be able to view the Clover coverage summary for the job.

**Related pages:**
- Enabling the Clover plugin
- Viewing the Clover code-coverage for a build
- Generating reports across multiple plans

To view the Clover coverage summary for a job:

1. Navigate to the desired job, as described on Configuring jobs.
2. Click the Clover tab, to see graphs of the coverage history and lines of code history.

**Screenshot: Clover Coverage Summary for a job**
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 plugin).

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 plugin
- Generating reports across multiple plans

To view Clover code-coverage for a build result:

1. Navigate to the build results for the plan, as described in Viewing a build result, and click the desired build result.
2. Click the Clover tab.

*Screenshot: Clover code-coverage for a build result*
Integrating Bamboo with Atlassian applications

You can integrate Bamboo with the following Atlassian applications:

**When Bamboo is integrated with JIRA**, you can:

- run a Bamboo build when releasing a JIRA version
- 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.

**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
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.

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 see which issues are being actively coded, which builds have run for an issue, find the build that fixed the issue, download your distribution and much more.

Configuring Bamboo and JIRA to work together simply requires you to set up an application link (two-way) between JIRA and Bamboo.

<table>
<thead>
<tr>
<th>Application</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo</td>
<td>Version 3.1 or later</td>
<td>If you are using an earlier version of Bamboo, please refer to these instructions in the corresponding older version of the Bamboo documentation.</td>
</tr>
</tbody>
</table>
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.

**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.
- 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 JIRA-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.

Linking JIRA issues to a build

If your organisation uses the 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 the build. If you have specified an issue in your build comments, labels or commit messages (note, you must specify the issue key in upper-case), then these issue links are either created automatically by Bamboo. You can also manually add new issue links to your build, or edit or remove any issue links.

On this page:

- Editing issue links for your build
- Manually linking issues to a build

Related pages:

- Viewing linked JIRA issues
- Integrating Bamboo with JIRA

Editing issue links for your build

Issue links that have been created automatically or manually, can be edited or removed from the build.

To edit an existing JIRA issue link for a build result:

1. Go to the plan in Bamboo.
2. Click the Issues tab in the 'Plan Summary'.

All of the JIRA issues linked to your build are listed, grouped by 'Fixed Issues' and 'Related Issues'. By default, all issue links that are automatically created by Bamboo are created as 'Related Issues'.

- To change a 'Related Issue' to a 'Fixed Issue', click the up arrow icon.
- To change a 'Fixed Issue' to a 'Related Issue', click the down arrow icon.
- To remove an issue link from the build (the issue will not be removed from JIRA), click the trash icon.
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 the **Issues** tab in the 'Plan Summary'. All of the JIRA issues linked to your build will be listed.
3. Click the **Add linked issue** link.
4. Choose an issue type from the **Type of Issue Link** list:
   - **Fixed** means that this issue is fixed by this build, e.g. a bug.
   - **Related** means that this issue is related to this build, but not fixed by it, e.g. a documentation task related to changes from the build.
5. Enter the JIRA issue key of the issue you want to link to this build. Please note, you must specify the issue key in upper-case, e.g. 'JIRA-1234'.
6. Click **Save**.

**Screenshot: Adding new JIRA issue links to a build**

<table>
<thead>
<tr>
<th>Plan Summary</th>
<th>Tests</th>
<th>Changes</th>
<th>Artifacts</th>
<th>Tracking</th>
<th>Logs</th>
<th>Metadata</th>
<th>Issues</th>
</tr>
</thead>
</table>

**Add Linked JIRA Issue**

JIRA issues can be manually linked to this build as either:

- 'Fixed' - fixed by this build
- 'Related' - linked to the build but not fixed by it

**Type of Issue Link**: Related

**JIRA Issue**: BAM-10234

Please enter the JIRA issue key (for example PROJ-1323)

**Save**  **Cancel**

Viewing linked JIRA issues

If your organisation uses Atlassian's [JIRA](http://jira.atlassian.com) and your administrator has integrated Bamboo with JIRA, you will be able to view the JIRA issues that have been linked to a build. These issue links are either created automatically by Bamboo or can be added manually. This provides an easy way to jump to relevant issues to see details about what the code is intended to achieve.

The JIRA issues linked to a build can be viewed on the build result pages. If you have specified an issue in your build comments, labels or commit messages, then they will be automatically linked to your build and displayed. Issues linked manually are also displayed.

**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
- 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 the issue in JIRA.

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 desired build result.

   - Plan Summary tab — the ‘JIRA Issues’ section displays two of the issues linked to the build.
   - Issues tab — displays all of the JIRA issues linked to the build. Related issues are linked to the build but are not fixed by it.

Screenshot: JIRA issues for a build result — Plan Summary tab
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. Using Confluence and Bamboo together allows you to embed Bamboo gadgets in Confluence pages.

Configuring Bamboo and Confluence to work together simply requires you to set up an application link (two-way) between Confluence and Bamboo.
Before you begin

Version Requirements

<table>
<thead>
<tr>
<th>Application</th>
<th>Version Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo</td>
<td>Version 3.2 or later</td>
</tr>
<tr>
<td>Confluence</td>
<td>Version 3.5.9 or later</td>
</tr>
</tbody>
</table>

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 FishEye

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.

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 other applications
- Specifying the source repository

### To integrate Bamboo with FishEye:

1. Navigate to the 'Source Repositories' tab for the plan.
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]

<table>
<thead>
<tr>
<th>FishEye</th>
<th>FishEye URL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This is the URL of your FishEye instance e.g <a href="http://svn.atlassian.com/fisheye/">http://svn.atlassian.com/fisheye/</a></td>
</tr>
<tr>
<td>Repository Name</td>
<td></td>
</tr>
<tr>
<td>The name of the repository in FishEye e.g myProject</td>
<td></td>
</tr>
<tr>
<td>Repository Path</td>
<td></td>
</tr>
<tr>
<td>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</td>
<td></td>
</tr>
</tbody>
</table>

### 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.

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 name (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.
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

Associating your author name with your user profile

An author is any person who contributes to a build by checking-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.
Before you begin:

- If your Bamboo user profile has not yet been associated with your author name, then:
  - your 'My Bamboo' tab will not contain any data about your recent builds.
  - your 'Author' information will not include a User Details tab.

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.

Related pages:
- Managing your user profile

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

Administering plans
Configuring global plan settings.

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.

Plugins
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

Administering 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. These plan administration tasks are:

- Moving plans to a different project — organising plans in projects.
- Modifying multiple plans in bulk — making changes, such as adding a notification, to all plans at once.
- Monitoring job builds — configuring plan timeout events.
- Configuring concurrent builds — building the same plan on more than one agent at the same time.

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
To move a plan to a different project:

1. Click **Administration** in the menu bar.
2. Click **Move Plans** (under ‘Plans’) in the left-hand panel.
3. 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**.
4. Select one or more plans to move.
5. 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

Warning: 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 plan 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**: New Project
- **Project Name**: A New Project
- **Project Key**: NEWPROJ

Select: All, None

**Artifact Sharing Dogfooding**
- Artifact sharing
- CI Tests
- Maven Sharing CI Tests
- Maven Sharing CI Tests with Maven 3

**Bamboo Main**
- CI Tests
- Extras
- Stable CI Tests
- Stable Extras

**UI**
- Default
- MC Test
- WebDriver Tests
- WebDriver Tests JDK 1.6

Move

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

<table>
<thead>
<tr>
<th>Bulk Action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add new notification</td>
<td>See Configuring notifications for a plan and its jobs for further details.</td>
</tr>
<tr>
<td>Remove all notifications</td>
<td>See Configuring notifications for a plan and its jobs for further details.</td>
</tr>
<tr>
<td>Disable Plan</td>
<td>See Disabling or deleting a plan for further details.</td>
</tr>
<tr>
<td>Enable Plan</td>
<td></td>
</tr>
<tr>
<td>Run manual build</td>
<td>You have the option to disable dependencies when running the manual builds for the selected plans.</td>
</tr>
<tr>
<td>Update CVS module</td>
<td>See CVS documentation for further details.</td>
</tr>
<tr>
<td>Update CVS root and credentials</td>
<td>See CVS documentation for further details.</td>
</tr>
<tr>
<td>Update SVN credentials</td>
<td>See Subversion documentation for further details.</td>
</tr>
<tr>
<td>Update SVN repository URL</td>
<td>See Subversion documentation for further details.</td>
</tr>
<tr>
<td>Update web repository</td>
<td>See the Subversion, CVS or Perforce documentation for further details.</td>
</tr>
</tbody>
</table>
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)

Monitoring job builds

The following Bamboo features can help you monitor your running job builds:

- Configuring the hanging build event
- Configuring the build queue timeout event
- Disabling build monitoring

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
- Changing how often Bamboo checks for hung builds
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 menu 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

Changing how often Bamboo checks for hung builds
By default, Bamboo will check whether a hanging build event has been thrown every 60 seconds. You can change this 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.

**On this page:**

- Configuring the build queue timeout event
- Disabling the build queue timeout event
- Changing how often Bamboo checks for build queue timeouts

**Related pages:**

- Configuring notifications for a plan and its jobs
- Disabling build monitoring

**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.

**To edit the build queue timeout event settings:**

1. Click Administration in the menu 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_
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.

Changing how often Bamboo checks 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 menu 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
### 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 menu 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**.

### 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](#)
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.

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Date</td>
<td>Tuesday, 15 Feb 2011</td>
</tr>
<tr>
<td>System Time</td>
<td>15:48:22</td>
</tr>
<tr>
<td>Up Time</td>
<td>1 day, 31 minutes, 14 seconds (since Mon Feb 14 15:18:08 EST 2011)</td>
</tr>
<tr>
<td>Username</td>
<td>panda</td>
</tr>
<tr>
<td>User Timezone</td>
<td>Australia/Sydney</td>
</tr>
<tr>
<td>User Locale</td>
<td>English (United States)</td>
</tr>
<tr>
<td>System Encoding</td>
<td>MacRoman</td>
</tr>
<tr>
<td>Operating System</td>
<td>Mac OS X 10.6.5</td>
</tr>
<tr>
<td>Operating System Architecture</td>
<td>x86_64</td>
</tr>
<tr>
<td>Available Processors</td>
<td>8</td>
</tr>
<tr>
<td>Application Server Container</td>
<td>Apache Tomcat6.0.18</td>
</tr>
</tbody>
</table>

## Java / JVM Information

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Version</td>
<td>1.6.0_17</td>
</tr>
<tr>
<td>Java Vendor</td>
<td>Apple Inc.</td>
</tr>
<tr>
<td>JVM Spec. Version</td>
<td>1.0</td>
</tr>
<tr>
<td>JVM Spec. Vendor</td>
<td>Sun Microsystems Inc</td>
</tr>
<tr>
<td>JVM Version</td>
<td>14.3-b01-101</td>
</tr>
<tr>
<td>JVM Vendor</td>
<td>Apple Inc.</td>
</tr>
<tr>
<td>JVM Name</td>
<td>Java HotSpot(TM) 64-Bit Server VM</td>
</tr>
<tr>
<td>JRE Version</td>
<td>1.6.0_17-b04-243</td>
</tr>
<tr>
<td>JRE Name</td>
<td>Java(TM) SE Runtime Environment</td>
</tr>
</tbody>
</table>

## Network

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>panda.sydney.atlassian.com</td>
</tr>
<tr>
<td>IP Address</td>
<td>172.20.6.108</td>
</tr>
</tbody>
</table>

## Memory Statistics

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Memory</td>
<td>292 MB</td>
</tr>
<tr>
<td>Free Memory</td>
<td>63 MB</td>
</tr>
<tr>
<td>Used Memory</td>
<td>229 MB</td>
</tr>
</tbody>
</table>

## Bamboo Version Information

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>3.0</td>
</tr>
<tr>
<td>Build Number</td>
<td>2301</td>
</tr>
<tr>
<td>Build Date</td>
<td>14/02/11</td>
</tr>
</tbody>
</table>
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.

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.

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.

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:
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. The build logs are located in the `<Bamboo-Home>/xml-data/builds/` sub-directories. The build logs can be downloaded as an artifact (see Viewing a build's artifacts).

  **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
  - Locating important directories and files
  - Viewing a build's artifacts

- **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. The location will generally be either the root `<Bamboo-Install>` directory or the directory you started Bamboo from. In case of a Tomcat webapp deployment, the logs are piped out to catalina.out file.

    Please note, this log is different to the `bamboo.log` found in the `<Bamboo-Install>/logs` directory, which is the log written by the Java Service wrapper.

  - **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.

  - **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.

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 that the level after it:

\[\text{DEBUG} > \text{INFO} > \text{WARN} > \text{ERROR} > \text{FATAL}\]

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 menu 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
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
```

1. 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 menu 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

You can access Bamboo's data from an external program by using Bamboo's REST-style remote API. The remote API is disabled by default. Bamboo will return an error if people try to use the remote API when it is disabled.

Please note, the Bamboo remote API described in **Bamboo Remote API - Deprecated** has been deprecated in
favour of the new Bamboo REST API. You only need to enable the Accept remote API calls? option (as described below), if you want to allow access to the deprecated remote API. Access to the new REST API is enabled by default.

To enable the remote API:

1. Click Administration in the top menu bar.
2. Click General Configuration (under ‘System’) in the left navigation panel.
3. Select Accept remote API calls?.
4. Click Save. Bamboo will now accept remote calls. You do not have to restart the Bamboo server.

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)

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 */
```

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
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.

**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](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](image)

**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**
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.

To enable (or disable) Gravatar support:

1. Click Administration in the top menu 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

Viewing an agent

To view an agent, including the agent properties, capabilities and the plans that an agent can build, see Viewing an agent.

Viewing the status of all agents

To view the status of all your agents, see Monitoring agent status.
Viewing an agent’s capabilities

To find out what capabilities an agent already has, please see Viewing an agent’s capabilities.

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.

About agents and capabilities

A Bamboo agent is a service that provides capabilities to 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 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:

- an executable (e.g. Maven)
- a JDK
- 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’).
- a Version Control System client application (e.g. Git)

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?
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 Bamboo's agents

A Bamboo agent is a service that provides capabilities to 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 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.
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.
   - 'X' is the number of online agents that are currently building Bamboo builds and 'Y' is the total number of available online agents.

   **Screenshot: Accessing the List of All Bamboo Agents**

4. A list of all agents in your Bamboo system will be displayed (see screenshot below).

   **Screenshot: View Agents**

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:
     
     **Screenshot: Accessing the Agent Name in the List of Builds**

     ![Building Tab](image)

     **Building**

     | Building | Status |
     |----------|--------|
     | JIRA Functional Tests - HEAD - Enterprise JDK 1.8 building on Core 2 | Building for 32 mins. 10 mins remaining |
     | JIRA - MAVEN 2 - Standalone Func Tests building on Elastic Agent on i-514f5d1a | Building for 35 mins. 25 mins remaining |

   - Click **X of Y online agents are building** in the 'Building' section of the page to view the list of all Bamboo Agents.
     
     **i**'X' is the number of online agents that are currently building Bamboo builds and 'Y' is the total number of available online agents.

     ![Building Tab](image)

     **Building**

     | Building | Status |
     |----------|--------|
     | JIRA Functional Tests - HEAD - Enterprise JDK 1.8 building on Core 2 | Building for 32 mins. 10 mins remaining |
     | JIRA - MAVEN 2 - Standalone Func Tests building on Elastic Agent on i-514f5d1a | Building for 35 mins. 25 mins remaining |

   - Click the name of an agent in the 'Local Agents' section of this page (see screenshot below).

   **i** You can view agents listed in the 'Remote Agents' or 'Elastic Images' sections of this page by clicking their names too. However, you will require Bamboo administrator permissions to access these details. Upon clicking one of these agent's names, you may be prompted to log in to Bamboo.

   **Screenshot: Choosing an Agent on the List of All Bamboo Agents Page**
4. The details of the selected agent will be displayed. If you have not changed the default view, the 'Recent Builds' view will be shown. If not, click the 'Recent Builds' tab.

Screenshot: View Agent
You can click on the 'Executable Plans' tab to view the plans that this agent is capable of building. 
Screenshot: View Agent - Executable Plans

You can also click on the 'Capabilities' tab to view the capabilities of this agent. 
Screenshot: View Agent - capabilities
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 as shown in the following screenshot:
   "Screenshot: Accessing the Agent Name in the List of Builds"
   - Hover your mouse pointer over the ‘X of Y online agents are building’ link and in the resulting popup, click on the agent’s name as shown in the following screenshot:
   "X' is the number of online agents that are currently building Bamboo builds and "Y" is the total number of available online agents.
4. The details of the selected agent will be displayed in the Bamboo administration console, where you can further configure this agent and its capabilities. By default, the 'Capabilities' view will be shown. If not, click the Capabilities tab.
You can click on the **Executable Plans** tab to view the plans that this agent is capable of building.

*Screenshot: View Agent - Executable Plans*
You can also click on the **System Properties** tab to view the system properties of this agent.  
*Screenshot: View Agent - System Properties*
Configuring agents

A Bamboo agent is a service that provides capabilities to 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 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.

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:
  • an executable (e.g. Maven)
  • a JDK
  • 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’).
  • a Version Control System client application (e.g. Git)

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).

Creating a local agent
A Bamboo agent is a service that provides capabilities to 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 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

Add

Cancel

Editing an agent's details

A Bamboo agent is a service that provides capabilities to 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 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.

For more information, see:

- Configuring agents
- Agents and capabilities
- Configuring a job's requirements

On this page:

- Editing an agent's name or description
- Editing an agent's capabilities

Related pages:

- Configuring capabilities
- Configuring an agent-specific executable capability
- Configuring an agent-specific JDK capability
- Configuring an agent-specific 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 an agent.
2. Click Edit Details.
3. Update the details for the agent.
4. Click Save.

Editing an agent's capabilities

To edit an agent's capabilities, see:

- Configuring an agent-specific executable capability
- Configuring an agent-specific JDK capability
- Configuring an agent-specific custom capability

Screenshot: Editing the details of an agent

![Screenshot: Editing the details of an agent](image)

Edit Details - Test Agent

Update the details of this agent, then click Update.

**Information**

<table>
<thead>
<tr>
<th>Name</th>
<th>Test Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Agent for testing purposes</td>
</tr>
</tbody>
</table>

[Update] [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.
Viewing an agent

A Bamboo agent is a service that provides capabilities to 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 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 an agent:

1. Click Administration in the menu 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.

Viewing an agent’s capabilities

A capability is a feature of an agent. A capability can be:

- an executable (e.g. Maven)
- a JDK
- 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’).
- a Version Control System client application (e.g. Git)

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).

To view an agent’s capabilities:

1. Click Administration in the top navigation bar.
2. Click **Agents** in the left navigation panel.
3. Click the name of the agent whose capabilities you wish to view.
4. If necessary, click the **Capabilities** tab to show a list of all 'Agent-Specific Capabilities' and 'Shared Capabilities' (see **screenshot below**). The capabilities in each of these sections are grouped into the following subsections:
   
   These subsections will only be shown if you have at least one of that particular type of capability defined in Bamboo.
   
   - Custom — [custom capabilities](#)
   - Executable — [executable capabilities](#)
   - JDK — [JDK capabilities](#)
   - Perforce, Mercurial, Git — [VCS capability](#)

   **To define a new capability, see** [Configuring capabilities](#).

**Screenshot: An agent's capabilities**
Notes

- How 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.

Viewing the jobs that an agent can build

To view the plans that an agent can build:

1. Click Administration in the top navigation bar.
2. Click Agents in the left navigation panel to display the 'Agents' screen.
3. Click the name of the agent of interest.
4. If necessary, click the **Executable Plans** tab to show a list of jobs which this agent is capable of building (see screenshot below).

**Screenshot: Viewing the executable plans for an agent**

![Agents Default Agent (local)](image)

<table>
<thead>
<tr>
<th>Plan</th>
<th>Last built</th>
</tr>
</thead>
<tbody>
<tr>
<td>DellTest - DellTestChain - Test1</td>
<td>DELTEST-DELTESTCHAIN-TEST1-65, 2 months ago (Last successful build DELTEST-DELTESTCHAIN-TEST1-54, 2 months ago)</td>
</tr>
<tr>
<td>ImportMaven2 - Maven Quick Start Archetype 1.0 - Default Job</td>
<td>IMPORTMAVEN2-MAVENQUICKSTARTARCHETYPE10-JOB1-10, 3 months ago (Never built successfully)</td>
</tr>
<tr>
<td>Logtest - logtest - Default Job</td>
<td>LOGTEST-LOGTEST-JOB1-50, 1 month ago (Last successful build LOGTEST-LOGTEST-JOB1-55, 1 month ago)</td>
</tr>
<tr>
<td>Project1 - Maven Quick Start Archetype 1.02 - Default Job</td>
<td>PROJECT1-MAVENQUICKSTARTARCHETYPE10-JOB1-93, 2 months ago (Never built successfully)</td>
</tr>
<tr>
<td>PROJECT - SUCCESS - Default Job</td>
<td>PROJECT-SUCCESS-JOB1-1, 2 months ago</td>
</tr>
<tr>
<td>PROJECT - FAIL - Default Job</td>
<td>PROJECT-FAIL-JOB1-1, 2 months ago</td>
</tr>
</tbody>
</table>

**Notes**

- Determining which plans an agent can build — 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.

- To see which agents are capable of building particular jobs, see **Determining which agents can build which jobs**.

**Determining which agents can build which jobs**

An agent can only build a job if the **capabilities** of the agent match the **requirements** of that job. Read more on **Configuring a job's requirements** and **Configuring capabilities**.

The 'Agents and Plans Matrix' page displays the agents that are capable of building every job currently set up in your Bamboo system, including disabled jobs. Every shared agent, remote agent and elastic image is listed.
against each job with either a tick ✓ or a cross ✗ to indicate whether or not the agent is not capable of building the job.

To see the jobs that a specific agent can build, refer to Viewing the jobs that an agent can build.

Related pages:
- Configuring a job’s requirements
- Configuring capabilities

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.

Agents and Plans Matrix

The matrix below shows which enabled, active Bamboo agents can execute which jobs. Each column represents an agent while each row represents a job. If you have enabled Elastic Bamboo, your elastic image configurations will also be displayed below. Elastic agents will derive their capabilities from the elastic image. Hover your mouse over any ✗ icons to see which job requirements are not being met by the relevant agent.

<table>
<thead>
<tr>
<th></th>
<th>1 Local Agent</th>
<th>2 bamboo-agent-PC: sydney.atlassian.com</th>
<th>✗ Default</th>
<th>✗ Maven 2.1 image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Default Job part of Bamboo - Acceptance Test</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>2 Default Job part of Bamboo - Acceptance Test JDK 1.6</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3 Func Test Clean part of Bamboo - CI Tests</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4 Func Test Dependencies part of Bamboo - CI Tests</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>5 Func Test Misc part of Bamboo - CI Tests</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

1. Maven 2.1 exist

Viewing an agent’s system properties

To view the system properties for an agent:

1. Navigate to the desired agent, as described on Viewing an agent.
2. Click System Properties tab to view the agent’s system properties.

Screenshot: Viewing an agent’s system properties
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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Available to execute builds.</td>
</tr>
<tr>
<td>Building</td>
<td>Currently executing a build.</td>
</tr>
<tr>
<td>Cancelling</td>
<td>Currently cancelling a Job build</td>
</tr>
<tr>
<td>Disabled</td>
<td>Not available to execute builds (see Disabling or deleting an agent).</td>
</tr>
<tr>
<td>Disabled - Building</td>
<td>Currently executing a build, but disabled so cannot execute further builds.</td>
</tr>
<tr>
<td>Disabled - Cancelling</td>
<td>Currently cancelling a build, and disabled so cannot execute further builds.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Agent</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Agent</td>
<td>Building - SANDBOX-RFIN-JOB1-22</td>
<td>View</td>
</tr>
<tr>
<td>Reporting agent</td>
<td>Idle (Disabled)</td>
<td>View</td>
</tr>
</tbody>
</table>

**Add Local Agent**

### Remote Agents

Remote agents run on computers other than the Bamboo server.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic Agent on i-01bcc86d</td>
<td>Idle</td>
<td>View</td>
</tr>
<tr>
<td>Elastic Agent on i-03bcc86f</td>
<td>Idle</td>
<td>View</td>
</tr>
<tr>
<td>Elastic Agent on i-1fbc873</td>
<td>Building - BT9-FUNC-JOB1-7815</td>
<td>View</td>
</tr>
</tbody>
</table>

There are currently 3 remote agents online (3 elastic). Start **elastic agents** here. A maximum of 25 agents are supported by your license.

---

### Configuring capabilities

A **capability** is a feature of an **agent**. A capability can be:

- an executable (e.g. Maven)
- a JDK

---

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
• 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').
• a Version Control System client application (e.g. Git)

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:
• Configuring a new executable
• Configuring a new JDK
• Configuring a new custom capability
• Configuring 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.

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 an agent.
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.

Screenshot: Viewing the capabilities of an agent
Configuring a new executable

About executables

An executable is a program external to Bamboo used to automate processes. 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.

At least one executable was automatically configured when you installed Bamboo. You can configure more by defining capabilities for the new executables (i.e. executable capabilities). Bamboo supports the following types of executables:

- Ant
- Maven
- Grails
- NAnt
- devenv.com
The executables listed above are supported out-of-the-box. If you need to use a different executable, 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).

Configuring a new executable

You can define a new executable capability for:

- a specific local agent — see Configuring an agent-specific executable capability
- all local agents — see Configuring a shared executable capability
- a specific remote agent — see Configuring an agent-specific executable capability
- all remote agents — see Configuring a shared executable capability

If an agent has its own specific executable capability, the value will override the value of a shared executable capability of the same name (if one exists).

Once you have configured a new executable capability in your Bamboo system, its label (e.g. 'Ant') will appear in the Executable list when you configure the executable for a task (see Configuring tasks). The executable you select will be used every time the task is executed 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.

Notes

- Pre-configured executables — The executable that was automatically configured 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.
- 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 a program external to Bamboo used to automate processes. 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 menu bar.
2. Click Executables in the left navigation panel.
3. Click a specific executable's tab on the left 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 Configuring an agent-specific 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.
- **Edit the configuration of a Job which relies on/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_
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 an agent-specific executable capability**

Once you have configured a new executable capability in your Bamboo system, its label (e.g. 'Ant') will appear in the Executable list when you configure the executable for a task (see Configuring tasks). The executable you select will be used every time the task is executed 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. 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).

**Related pages:**
- Configuring tasks
- Configuring a shared executable capability

To configure a new agent-specific executable capability:

1. Navigate to the desired agent, as described on Viewing an agent.
2. In the 'Agent-Specific Capabilities' section of the Capabilities tab, click Add Capability. The 'Add Capability' page is displayed (see screenshot below).
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 Executable list whenever a task's executable is configured.
6. In the Path field, type the appropriate path. 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.

**Screenshot: Add Capability — Executable**

![Add Capability - Local Agent](image)

**Configuring a shared executable capability**

Once you have configured a new executable capability in your Bamboo system, its label (e.g. 'Ant') will appear in the Executable list when you configure the executable for a task (see Configuring tasks). The executable you select will be used every time the task is executed 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. 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).
Configuring a new local server executable capability

Before you begin:

- Shared remote executable capabilities are **not shared** with elastic agents.
- 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 configure a new local server executable capability:

1. Click **Administration** in the top menu 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 **screen shot 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' dropdown 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**.

Configuring a new shared remote executable capability

To configure a new shared remote executable capability:

1. Click **Administration** in the top menu 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 > Builder** in the 'Add Capability' section at the end of the page (see **screenshot below**).
5. Select the appropriate type of executable from the **Type** list.
6. In the **Executable Label** field, type a name/label for the Builder.
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**.

**Screenshot: Add Shared Capability — Executable**
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:

```
$BAMBOO_HOME/.m2/AGENT-${agendid}/repository
```

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.

**Related pages:**
- Configuring a shared executable capability

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:
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 Executable](attachment:image.png)

**Elastic Image Configurations with capability**

3 elastic image configurations have this capability.

<table>
<thead>
<tr>
<th>Elastic Image Configuration</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>KBTEST</td>
<td>/opt/maven-2.0</td>
</tr>
<tr>
<td>Default</td>
<td>/opt/maven-2.0</td>
</tr>
<tr>
<td>Maven 2.1 Image</td>
<td>/opt/maven-2.0</td>
</tr>
</tbody>
</table>

**Jobs with requirement**

28 jobs rely on this capability.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifact Sharing Dogfooding</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>Artifact Sharing Dogfooding</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>Artifact Sharing Dogfooding</td>
<td>exists</td>
<td></td>
</tr>
</tbody>
</table>

**Capability Configuration**

| Local repository isolation | Yes |

[See More]
Configuring a new JDK

At least one JDK was automatically configured when you installed Bamboo. You can configure more by defining JDK capabilities.

You can define a new JDK capability for:

- a specific local agent — see Configuring an agent-specific JDK capability
- all local agents — see Configuring a shared JDK capability
- a specific remote agent — see Configuring an agent-specific JDK capability
- all remote agents — see Configuring a shared JDK capability

Note that if an agent has its own specific JDK capability, the value will override the value of a shared JDK capability of the same name (if one exists).

Once you have configured 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.

---

1 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.

Configuring an agent-specific JDK capability

Once you have configured 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. 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).

Related pages:
- Configuring a shared JDK capability

To configure a new agent-specific JDK capability:

1. Navigate to the desired agent, as described on Viewing an agent.
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.

Screenshot: Add Capability — JDK

Add Capability - Local JDK

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

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 have set up redundant JDK capabilities, you can view the list of JDK capabilities set up in Bamboo and delete any unwanted JDK capabilities.

Configuring a shared JDK capability

Once you have configured 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. 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).

On this page:

- Configuring a new local server JDK capability
- Configuring a new shared remote JDK capability
- Notes

Related pages:

- Configuring an agent-specific JDK capability

Configuring a new local server JDK capability

Before you begin:

- Shared remote JDK capabilities are not shared with elastic agents.

To configure a new local server JDK capability:

1. Click Administration in the top menu 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 (see screenshot below).
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.

Configuring a new shared remote JDK capability

To configure a new shared remote JDK capability:

1. Click Administration in the top menu 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.

Screenshot: ‘Add Capability — JDK’

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 have set up redundant JDK capabilities, you can view the list of JDK capabilities set up in Bamboo and delete any unwanted JDK capabilities.

### 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:

- [Configuring a new JDK](#)

### 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 [Configuring an Agent-specific 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](#).
   - Edit the configuration of a Job which 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**

---

*Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.*
Configuring 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 to be set so that Bamboo can checkout 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 or Git submodules)
- **Mercurial**
- **Perforce**

**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.

<table>
<thead>
<tr>
<th>Capability type</th>
<th>Example paths</th>
</tr>
</thead>
</table>
| 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 configure a new version control capability:

1. Navigate to the desired agent, as described in Viewing an agent.
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.

If you install a new agent on a machine that has Git already installed, the agent will find the Git client automatically.

### Configuring a new custom 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 '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 configure a new custom capability for:

<table>
<thead>
<tr>
<th>a specific local agent</th>
<th>see Configuring an agent-specific custom capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>a specific remote agent</td>
<td></td>
</tr>
<tr>
<td>all local agents</td>
<td>see Configuring a shared custom capability</td>
</tr>
<tr>
<td>all remote agents</td>
<td></td>
</tr>
</tbody>
</table>

Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

### Configuring an agent-specific custom 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 'operating.system=WindowsXP' for the appropriate agent(s), and specify it as a requirement for this job. (See Configuring a job's requirements.) 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).

### Related pages:

- Configuring a shared custom capability

To configure a new agent-specific custom capability:

1. Navigate to the desired agent, as described in Viewing an agent.
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.

Screenshot: Adding an agent-specific custom capability

![Add Capability - Local Agent](image)

**Configuring a shared custom 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 'operating.system=WindowsXP' for the appropriate agent(s), and specify it as a requirement for this job. (See Configuring a job's requirements.) **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).

**On this page:**
- Configuring a new local server custom capability
- Configuring a new shared remote custom capability

**Related pages:**
- Configuring an agent-specific custom capability

**Configuring a new local server custom capability**

Before you begin:
- Shared remote custom capabilities are **not shared** with elastic agents.

To configure 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.

**Configuring a new shared remote custom capability**

To configure 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**.

**Screenshot: Adding a shared custom capability**

![Add Capability](image)

**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 agents and capabilities
- 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 an agent**.
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 menu 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 agents and capabilities
- Configuring capabilities

Renaming a local server capability

To rename a local server capability:

1. Click Administration in the menu 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 menu 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
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 provides capabilities to 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 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.

For more information, see:

- Configuring agents
- Agents and capabilities
- Configuring a job's requirements

On this page:

- Disabling remote agent support
- Re-enabling remote agent support

Disabling 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.

Before you begin:

- Note, remote agent support must be enabled to use Elastic Bamboo. Disabling remote agent support will disable Elastic Bamboo.
To disable remote agent support:

1. Click the Administration link in the top navigation bar.
2. Click the Agents link in the left navigation column to display the Agents screen.
3. Click the Disable Remote Agent Support link (see screenshot below).

Screenshot: Disabling remote agent support

---

To re-enable remote agent support:

1. Click the Administration link in the top navigation bar.
2. Click the Agents link in the left navigation column. This displays the Agents screen with a message indicating that remote agent support is disabled.
3. Click the Enable Remote Agent Support link (see screenshot below).

Screenshot: Re-enabling remote agent support

---

Working with Elastic Bamboo

Elastic Bamboo is a feature in Bamboo that 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.

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 Elastic Bamboo** — managing your elastic image, instances and agents.
- **Elastic Bamboo Security** — setting up secure communication between Bamboo and the EC2.

---

About Elastic Bamboo
Elastic Bamboo is a feature in Bamboo that 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.

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 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 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. |
### 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.

### 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 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](https://www.atlassian.com/software/jira) 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](https://aws.amazon.com/registration) before you can enable Elastic Bamboo.

Full details on Amazon EC2 pricing is available on the [Amazon EC2 pricing page](https://aws.amazon.com/ec2/pricing/). 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](https://docs.atlassian.com/elastic-bamboo/docs/configuring-elastic-images/) 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](https://aws.amazon.com/management/). 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](https://aws.amazon.com/blogs/compute/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 is a feature in Bamboo that allows you to use computing resources from the Amazon Elastic Compute Cloud.
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.

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 document 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.
• 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 is a feature in Bamboo that 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 menu 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.
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 (see Notes below).

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.

<table>
<thead>
<tr>
<th>Maximum Number of Elastic Instances</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatically shut down elastic instance when elastic agent process ends</td>
<td>Controls whether your elastic instances will automatically shut down after the elastic agent processes running on them terminate.</td>
</tr>
<tr>
<td><strong>Shutdown Delay</strong> — controls how long an elastic instance will wait before shutting down, after its elastic agent process terminates.</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Enable support for spot instances</th>
<th>Select this checkbox to enable support for spot instances.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fallback to a regular instance after</td>
<td>The time (in minutes) after which Elastic Bamboo will fall back to using a regular instance, if a spot instance has not become available.</td>
</tr>
<tr>
<td>Your current bid levels (per hour)</td>
<td>Fill out this table with your bids. The bids are categorised by EC2 instance type and operating system.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Upload AWS account identifiers to new elastic instances</th>
<th>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).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key files location</td>
<td>Choose how private key and certificate will be provided.</td>
</tr>
<tr>
<td>Account Private Key File</td>
<td>You must specify the location of this file to use the Amazon EBS with Elastic Bamboo.</td>
</tr>
<tr>
<td>Account Certificate File</td>
<td>You must specify the location of this file to use the Amazon EBS with Elastic Bamboo.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Elastic Instance Management</th>
<th>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.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Balances build queue clearance rates with elastic instance usage costs.</td>
</tr>
<tr>
<td>Aggressive</td>
<td>Favours higher build queue clearance rates but with higher elastic instance usage costs.</td>
</tr>
<tr>
<td>Passive</td>
<td>Favours lower instance usage costs but with lower build queue clearance rates.</td>
</tr>
<tr>
<td>Custom</td>
<td>Choose your own settings, as described below.</td>
</tr>
<tr>
<td>Disabled</td>
<td>Disables Bamboo's automatic elastic instance management feature.</td>
</tr>
</tbody>
</table>
### 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.

> *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.*

<table>
<thead>
<tr>
<th>Allowed non-Bamboo instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>The maximum number of elastic instances allowed on your AWS account that are not controlled by this Bamboo instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Number of Instances to Start at Once</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Builds in Queue Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>The total number of builds in a queue. When this and all other thresholds have been reached, new elastic instances will be started.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Elastic Builds in Queue Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Average Queue Time Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

### Generating your AWS Private Key File and Certificate File

The Amazon Web Services (AWS) private key file and certificate file work together to allow Elastic Bamboo to secure access AWS. It is different to the security mechanism provided by the AWS Secret Access Key and is 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 first time you use Elastic Bamboo, Bamboo will automatically generate the private key file and certificate file for you. 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*  
*Screenshot above: Downloading the generated AWS private key file and certificate file*

**Notes**

Please take note of the following important information regarding your AWS private key file and certificate file:

- 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 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 in this document).


4. Access your elastic instance via SSH (see Accessing an elastic instance for instructions).

5. Follow the steps below to create an EBS volume and attach it to the elastic instance (step 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.
   
   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.
   
   The device can not unmount if any terminals are currently in the mounted volume.
   

**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.
1. Navigate to the Bamboo administration menu, i.e. click the ‘Administration’ link in the top navigation bar of the Bamboo application.
2. Click the ‘Image Configurations’ link in the left navigation column under the ‘Elastic Bamboo’ sub-header. The ‘Manage Elastic Image Configurations’ page will display.
3. Click the ‘Edit’ link in the ‘Operations’ column of 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).

4. Check the ‘Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances’ checkbox. The ‘EBS Snapshot ID’ field will display.
5. Enter the Snapshot ID of your EBS snapshot in the ‘EBS Snapshot ID’ field.
6. Click the ‘Save’ button. 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:
   - **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.

  ⚠️ *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](https://confluence.atlassian.com/pages_password_protected_page.action?pageId=110985) 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](https://confluence.atlassian.com/pages_password_protected_page.action?pageId=110985) 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](https://confluence.atlassian.com/pages_password_protected_page.action?pageId=110985). 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](https://confluence.atlassian.com/pages_password_protected_page.action?pageId=110985)

---

**Uploading Maven 2 repository data**

You can upload [Maven 2 repository data](https://confluence.atlassian.com/pages_password_protected_page.action?pageId=110985) 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:

```bash
#!/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

yum install -y vnc-server
xorg-x11-server-Xvfb xterm
xorg-x11-server-utils \
twm xorg-x11-fonts-* yum install
compat-libstdc++-33

if [ ! -d /home/bamboo/.vnc ]; then
mkdir /home/bamboo/.vnc
fi

cp /mnt/bamboo-ebs/bin/vncpasswd
/home/bamboo/.vnc/passwd
cp /mnt/bamboo-ebs/bin/vncxstartup
/home/bamboo/.vnc/xstartup

chown -R bamboo.bamboo /home/bamboo/.vnc
chmod 600 /home/bamboo/.vnc/passwd
chmod u+x /home/bamboo/.vnc/xstartup

rm -rf /opt/firefox

zcat
/mnt/bamboo-ebs/bin/firefox-2.0.0.20.tar.gz
z | tar -xvf - -C /opt

grep -q LD_LIBRARY_PATH
/home/bamboo/.bashrc
if [ "$?” == "1" ]; then
echo "export LD_LIBRARY_PATH=/opt/firefox"
>> /home/bamboo/.bashrc
fi
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.

### 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 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 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 also create your own custom elastic images.

An image is similar to an agent, hence 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 (via the elastic agent created from the image), similar to how you view the Jobs that an agent is capable of viewing. 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 menu 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>The name of the image.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMI ID</td>
<td>The Amazon Machine Image identifier that uniquely identifies the image.</td>
</tr>
<tr>
<td>EBS Snapshot ID</td>
<td>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.</td>
</tr>
<tr>
<td>Instance Type</td>
<td>The instance type of new instances started from this image. Read more about Amazon instance types.</td>
</tr>
<tr>
<td>Availability Zone Preference</td>
<td>New instances started from this image will be run in the Amazon availability zone named here.</td>
</tr>
<tr>
<td>Active Instances</td>
<td>The number of currently active instances that were started from this image.</td>
</tr>
</tbody>
</table>
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-0ab54563
- **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

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, Test builds=true’). For an agent to be able to build a job, both the ‘Key’ and ‘Value’ must match the job’s requirements.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamboo.functionalTest</td>
<td>true</td>
<td>View</td>
</tr>
</tbody>
</table>

**Builder**

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

<table>
<thead>
<tr>
<th>Builder Label</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant (Ant)</td>
<td>/opt/apache-ant-1.7.1</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2 (Maven 2.x)</td>
<td>/opt/maven-2.0</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.1 (Maven 2.x)</td>
<td>/opt/maven-2.1</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.2 (Maven 2.x)</td>
<td>/opt/maven-2.2</td>
<td>View</td>
</tr>
</tbody>
</table>

**JDK**

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

<table>
<thead>
<tr>
<th>JDK Label</th>
<th>Java Home</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDK</td>
<td>/opt/jdk-5</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.5</td>
<td>/opt/jdk-5</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.6</td>
<td>/opt/jdk-6</td>
<td>View</td>
</tr>
</tbody>
</table>

**Mercurial**

The path to the Mercurial executable (e.g. ‘C:\Program Files (x86)\Mercurial\hg.exe’ or ‘/usr/local/bin/hg’)

<table>
<thead>
<tr>
<th>Executable</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercurial</td>
<td>/usr/bin/hg</td>
<td>View</td>
</tr>
</tbody>
</table>
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 also can 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 a different set of inherited capabilities than 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 menu bar in Bamboo.
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’:

<table>
<thead>
<tr>
<th>Name</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A description for your image, which is used in Bamboo only.</td>
</tr>
<tr>
<td><strong>AMI ID</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Instance Type</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Availability Zone</strong></td>
<td>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 &quot;Default (chosen by EC2)&quot; to allow Amazon to select the best zone for your instance. Read more about Amazon EC2 availability zones.</td>
</tr>
<tr>
<td><strong>Product</strong></td>
<td>The EC2 product name.</td>
</tr>
</tbody>
</table>

**Screenshot: Manage your Elastic Image Configurations**

![Create Elastic Image Configuration](image)

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 menu bar in Bamboo.
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’).
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 operating 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 also **create your own custom elastic images**.

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**. 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.
- These instructions are written for **Unix/Linux operating systems**.
- 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). However, we recommend that you follow the steps below unless you are an experienced AWS user.

---

**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

Before you begin, you need to 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. You can get the list of available AMIs by executing the following command:

```
ec2-describe-images -a
```

⚠️ Please check whether you want to launch **32-bit** or **64-bit** instances from your custom elastic image before selecting an existing AMI. Elastic Bamboo creates "High-CPU Medium" instances (32-bit) by default.

We recommend the following existing Linux/UNIX AMIs for customisation (in order of preference):

- **Atlassian’s AMI** — this AMI is the [default image](https://docs.atlassian.com/software/amazon-linux-ami/) provided by Atlassian. It is an Amazon Linux image updated and prepared for Bamboo, i.e. you will not have to install any Bamboo prerequisites.
- **Amazon’s AMI** — this is the base CentOS-based image provided by Amazon. It does not have any
Bamboo prerequisites installed. Typically, you will be better off using the Atlassian AMI.

- **Canonical’s AMI** — this is an Ubuntu image provided by Canonical. It does not have any Bamboo prerequisites installed. You can find AMI ids for Ubuntu images on the Canonical’s site.

The names of Atlassian's and Amazon's AMIs (and hence, their IDs) may change with each release of Bamboo, including both major and minor (or point) releases. To quickly access Atlassian's AMI ID for 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, the procedure is more complicated:

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 running
ami-e55bbd8c 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:
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

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 First, you need to create a `bamboo` user on your instance by running the following command:

```
useradd -m bamboo
```

5.3.2 Then, install Bamboo Agent binaries as described below. In this case we're using image version 1.8, you can find out which version you need by following the steps in section "Selecting an Existing AMI". Bamboo 3.4 is a special case, you should use image version 1.8 with it.
imageVer=1.8
wget https://maven.atlassian.com/content/repositories/atlassian-public/com/atlassian/bamboo/atlassian-bamboo-elastic-image/${imageVer}/atlassian-bamboo-elastic-image-${imageVer}.zip
mkdir -p /opt/bamboo-elastic-agent
unzip atlassian-bamboo-elastic-image-${imageVer}.zip -d /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 up the Bamboo agent automatically when the instance is started. You can do this by appending the `rc.local` file to the one that already exists on your instance, by running the following command on your instance in the `mnt` directory:

    cat /opt/bamboo-elastic-agent/etc/rc.local >> /etc/rc.d/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. 3.4.4).

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 bamboo-elastic-agent - bamboo`

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*
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:

   ```bash
   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:

   ```bash
   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:

   ```bash
   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.

To quickly access Atlassian's AMI ID for 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 ID of Atlassian's AMI is listed in the table row with the name 'Default Image'.
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**’.

   Now, follow the instructions from section ”Creating an image of your Customised Instance” to create an AMI.

   **Upgrading the agent for your 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 .

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 Elastic Bamboo document.

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 menu 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'.

<table>
<thead>
<tr>
<th>Current status</th>
<th>The status of the elastic instance. Values include, 'Pending' (instance starting up), 'Running' and 'Shutting down'.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public DNS</td>
<td>The public DNS address of the elastic instance. The IP address of the elastic instance is displayed here.</td>
</tr>
<tr>
<td>Start Time</td>
<td>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.</td>
</tr>
<tr>
<td>Elastic Agent</td>
<td>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.</td>
</tr>
<tr>
<td>Current Availability Zone</td>
<td>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.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Attached Volumes</td>
<td>The IDs of the attached EBS volumes, if you have configured your elastic instances to use EBS.</td>
</tr>
<tr>
<td>Configuration</td>
<td>The name of the elastic image configuration that was used to create this elastic instance. Click the name to configure the elastic image.</td>
</tr>
<tr>
<td>AMI ID</td>
<td>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).</td>
</tr>
<tr>
<td>EBS Snapshot ID</td>
<td>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.</td>
</tr>
<tr>
<td>Bambu 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: bamboo.agent.elastic.ebsVolumeSupervisionIntervalInSeconds</td>
<td></td>
</tr>
<tr>
<td>Instance Type</td>
<td>The instance type of your instance.</td>
</tr>
<tr>
<td>SSH Access</td>
<td>Please see Accessing an elastic instance for information on using this function.</td>
</tr>
<tr>
<td>Accessing Logs</td>
<td>Please see Accessing an elastic instance for information on using this function.</td>
</tr>
</tbody>
</table>

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-0ab54563
- **EBS Snapshot ID**: snap-58204e00
- **Instance Type**: High-CPU Medium

**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.

```bash
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.

```bash
scp -i elasticbamboo.pk root@ec2-184-72-81-22.compute-1.amazonaws.com:/home/bamboo/bamboo-elastic-agent.log .
```

**Amazon EC2 Console**
You can manage your EC2 instances using the Amazon EC2 Console. Once you've 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.
To start an elastic instance:

1. Click Administration in the top menu 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.
   
   ![Note: 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 |

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.
Managing your elastic instance schedules

To manage your elastic instance schedules:

1. Click Administration in the top menu bar.
2. Click Instance Schedule in the left navigation panel (under 'Elastic Bamboo').
3. Do any of the following:

<table>
<thead>
<tr>
<th>Task</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a new schedule</td>
<td>Click Add Elastic Instance Schedule to create a schedule from new.</td>
</tr>
<tr>
<td></td>
<td>Click Copy to use an existing schedule as a template.</td>
</tr>
<tr>
<td></td>
<td>See the Adding a New Elastic Instance Schedule section below for further instructions.</td>
</tr>
<tr>
<td>Edit an existing schedule</td>
<td>Click Edit for an existing schedule. You can also Delete existing schedules.</td>
</tr>
<tr>
<td>Enable existing schedules</td>
<td>Click Enable for a particular schedule, or click Enable All.</td>
</tr>
<tr>
<td>Disable existing schedules</td>
<td>Click Disable for a particular schedule, or click Disable All.</td>
</tr>
</tbody>
</table>

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

Adding a new elastic instance schedule

1. Click Administration in the top menu 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.

<table>
<thead>
<tr>
<th>Enabled</th>
<th>Clear if you do not want this schedule to be enabled when you create it.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger On</td>
<td>Choose when this schedule should start:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Next Bamboo startup</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>A cron schedule</strong> — edit <strong>Schedule</strong> as required. For information on constructing cron expressions, see <a href="#">this FAQ</a>.</td>
</tr>
<tr>
<td>On Trigger Bamboo Should</td>
<td>Choose the action Bamboo should perform:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Stop all elastic instances</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Adjust number of active instances</strong></td>
</tr>
<tr>
<td>Image Config</td>
<td>Choose which image the elastic instances should be started from. The elastic agents running on the instances will inherit the capabilities from the image.</td>
</tr>
<tr>
<td>Active Instances</td>
<td>Choose the logical operator and specify a value for the number of active instances.</td>
</tr>
</tbody>
</table>

**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**

- **KBTEST**

**Active Instances**

- **exactly**
  - Number of instances that Bamboo will attempt to adjust to.
```

Created by Atlassian in 2012. Licensed under a [Creative Commons Attribution 2.5 Australia License](#).
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 menu 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 menu 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. However, if you have orphaned your elastic instances from your Bamboo server (e.g. restarted your Bamboo server without shutting down your elastic instances), you may need to shut your elastic instances down directly in the Amazon Web Services (AWS) console.

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 menu 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**
Viewing your elastic agent usage history

When you shut down an elastic instance, the agent process for that instance is killed. As such, the elastic agent will not appear in an offline status, but will be removed altogether from your available 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 menu 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 for the elastic agent will display (see screenshot). This page 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**

---

**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, 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 menu 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. Adding a new capability to an image is very similar to adding capabilities to non-elastic agents. Please see the following pages for further information:
   - Configuring a new executable
   - Configuring a new JDK
   - Configuring a new version control capability
   - Configuring a new custom capability
5. You can also edit, rename and delete a capability from an elastic image, similar to how you would edit, rename and delete a capability from a non-elastic agent. 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

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 job's requirements.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamboo.functionalTest</td>
<td>true</td>
<td>View</td>
</tr>
</tbody>
</table>

#### Builder

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

<table>
<thead>
<tr>
<th>Builder Label</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant (Ant)</td>
<td>/opt/apache-ant-1.7.1</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2 (Maven 2.x)</td>
<td>/opt/maven-2.0</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.1 (Maven 2.x)</td>
<td>/opt/maven-2.1</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.2 (Maven 2.x)</td>
<td>/opt/maven-2.2</td>
<td>View</td>
</tr>
</tbody>
</table>

#### JDK

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

<table>
<thead>
<tr>
<th>JDK Label</th>
<th>Java Home</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDK</td>
<td>/opt/jdk-5</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.5</td>
<td>/opt/jdk-5</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.6</td>
<td>/opt/jdk-5</td>
<td>View</td>
</tr>
</tbody>
</table>

#### Mercurial

The path to the Mercurial executable (e.g. `C:\Program Files (x86)\Mercurial\hg.exe` or `/usr/local/bin/hg`)

<table>
<thead>
<tr>
<th>Executable</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercurial</td>
<td>/usr/bin/hg</td>
<td>View</td>
</tr>
</tbody>
</table>

### Add Capability

<table>
<thead>
<tr>
<th>Capability Type</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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](#).

<table>
<thead>
<tr>
<th>Instance ID</th>
<th>Agent Name</th>
<th>Status</th>
<th>Up Time</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-2776e24e</td>
<td>Elastic Agent on i-2776e24e</td>
<td>Running</td>
<td>5 minutes</td>
<td>View</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Instance ID</th>
<th>Agent Name</th>
<th>Status</th>
<th>Up Time</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-2776e24e</td>
<td>Elastic Agent on i-2776e24e</td>
<td>Online (Disabled)</td>
<td>6 minutes</td>
<td>Enable</td>
</tr>
</tbody>
</table>

### 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 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 fashion 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 that job build to this 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 menu 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
About users and authors

An author is any person who contributes to a build by checking-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
- Connecting to external user directories

Notes

An author is any person who contributes to a build by checking-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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Username cannot be changed after the user is created.</td>
</tr>
<tr>
<td>Password</td>
<td>The user can easily change their password later.</td>
</tr>
<tr>
<td>Email</td>
<td>The address to which notifications are sent.</td>
</tr>
<tr>
<td>Instant Messaging Address</td>
<td>If no IM address is specified, Bamboo will not be able to recognise the user's context when interacting using IM.</td>
</tr>
<tr>
<td>Source Repository Aliases</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

**Screenshot: Deleting a Bamboo user**
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@fooobar.
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](#)).

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.

```sql
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';

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:
## 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.

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:

<table>
<thead>
<tr>
<th>External User Management Setup</th>
<th>Read-only External User Management? check-box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo integrated with — Crowd using the Crowd database (i.e. Internal Directories)</td>
<td>Unchecked</td>
</tr>
<tr>
<td>Bamboo integrated with — Crowd connected to a read-only LDAP</td>
<td>Checked</td>
</tr>
</tbody>
</table>
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:

   ```xml
   <!--<authenticator
   class="com.atlassian.bamboo.user.authentication.BambooAuthenticator"/>-->
   ```

4. Add a new authenticator, by adding the following tag:

   ```xml
   <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 versions prior to 1.2.2, it is not possible to perform XML backups of your Bamboo instance when
integrated with LDAP. In Bamboo version 1.2.2 and later, exports are possible, but user data will not be included in the export.

⚠️ 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

Before you begin

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 on your LDAP** — If you have users on your LDAP 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 on your LDAP** — If you have groups on your LDAP that are also on Bamboo, this may cause unpredictable behaviour when you attempt to integrate your LDAP server with Bamboo.

Step 1 — Backup your data
We strongly recommend that you backup your data before attempting LDAP integration.

Step 2 — 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.

   ```xml
   <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>
   ...
   
   Please ensure that the following line is also active in your `atlassian-user.xml` (it should be there by default):

   ```xml
   <hibernate name="Hibernate Repository" key="hibernateRepository" description="Hibernate Repository" />
   ```

Step 3 — 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>(&amp; (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 4 — 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.L
dapCtxFactory</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 onto 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:
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](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.

On this page:

- Disabling the Caching of Users
- Enabling the Caching of Users
- Configuring the LDAP Caches
- Notes

Disabling 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 disable the caching of users:

1. Click the 'Administration' link in the top navigation bar.
2. Edit the file .../webapp/WEB-INF/classes/atlassian-user.xml
3. Set the property cache="false" on your LDAP repository, as shown in the example below:

   ```xml
   <ldap key="myLdapRepository" name="LDAP Repository@hecate.atlassian.com" cache="false">
     <host>hecate.atlassian.com</host>
     <port>389</port>
   </ldap>
   ```

Enabling the Caching of Users

By default, caching is activated for your LDAP users. If you need to enable caching, follow the instructions below:

To enable the caching of users:

1. Edit the file .../webapp/WEB-INF/classes/atlassian-user.xml
2. Set the property cache="true" on your LDAP repository as shown in the example below:
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:

- Disabling the Caching of Users
- Enabling the Caching of Users
- Configuring the LDAP Caches
  - Configuring Caches for Users
    - LDAPUserManagerReadOnly.*.users
    - LDAPUserManagerReadOnly.*.users_ro
    - LDAPUserManagerReadOnly.*.repository
  - Configuring Caches for User Groups
    - LDAPGroupManagerReadOnly.*.groups
    - LDAPGroupManagerReadOnly.*.groups_hasMembership* and LDAPGroupManagerReadOnly.*.groups_getGroupsForUser
    - LDAPGroupManagerReadOnly.*.repositories

- Notes
  - Related Topics

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 atlas...
ssian-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.

```xml
<cache
    name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.users"
    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.

```xml
<cache
    name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.users_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.

```xml
<cache
    name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.repository"
    maxElementsInMemory="500"
    eternal="false"
    timeToIdleSeconds="300"
    timeToLiveSeconds="300"
/>
```

**Configuring Caches for User 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.
LDAPGroupManagerReadOnly.*.groups_hasMembership and *LDAPGroupManagerReadOnly.*.groups_getGroupsForUser The LDAPGroupManagerReadOnly.*.groups_hasMembership and LDAPGroupManagerReadOnly.*.groups_getGroupsForUser 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.

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.

Notes

Related Topics

Integrating Bamboo with LDAP
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`.

On this page:
- Using Paddle
- Parameters
- Sample output
- Notes

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-Install>/webapp/WEB-INF/classes/` directory.
3. Run `java -jar paddle-x.x.jar` (where x.x is the version of Paddle you downloaded).

Parameters

Paddle currently supports the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td><code>java -jar paddle-x.x.jar debug</code></td>
<td>Prints DEBUG messages to the console as well as paddle.log.</td>
</tr>
<tr>
<td>limit</td>
<td><code>java -jar paddle-x.x.jar limit=100</code></td>
<td>Sets the limit on the number of results returned by user and group queries. Defaults to 10.</td>
</tr>
</tbody>
</table>

Sample output
This is an example of a successful run:

```
#########################################
#########################################
#######################################
LDAP Support Tool version 1.1
Connection to LDAP/Active Directory Server
```

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
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$MALTSHOVEL$MS
SQLSERVER
Members:

(1)
CN=S-1-5-18,CN=ForeignSecurityPrincipals,D
C=ad,DC=atlassian,DC=com

Group:
CN=SQLServer2005MSSQLUser$MALTSHOVEL$MSSQL
SERVER
Members:

(1)
CN=S-1-5-18,CN=ForeignSecurityPrincipals,D
C=ad,DC=atlassian,DC=com

Group:
CN=SQLServer2005MSFTEUser$MALTSHOVEL$MSSQL
SERVER
Members:

(1)
CN=S-1-5-18,CN=ForeignSecurityPrincipals,D
C=ad,DC=atlassian,DC=com

Group:
CN=SQLServer2005MSOLAPUser$MALTSHOVEL$MSSQL
SERVER
Members:

(1)
CN=S-1-5-18,CN=ForeignSecurityPrincipals,D
C=ad,DC=atlassian,DC=com

Group:
CN=SQLServer2005NotificationServicesUser$M
ALTSHOVEL
No members in this group.

Found more than 10 results.

Notes

Related Topics

Integrating Bamboo with LDAP

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.

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
Deleting a group

Note that the **bamboo-admin** group cannot be deleted.

**Related pages:**
- [Managing groups](#)

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.

**Screenshot: Deleting a group**

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**.

**Related pages:**
- Managing groups

**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**.

**Screenshot: Adding users to a group**

**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 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 menu bar.
2. Click **Bulk Edit Plan Permissions** in the left navigation panel.
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**.

<table>
<thead>
<tr>
<th>User category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logged in Users</td>
<td>Users who are logged in to Bamboo.</td>
</tr>
<tr>
<td>Anonymous Users</td>
<td>Users who are not logged in to Bamboo.</td>
</tr>
<tr>
<td>User</td>
<td>A user already created in the Bamboo system.</td>
</tr>
<tr>
<td></td>
<td>To edit plan permissions for an existing user:</td>
</tr>
<tr>
<td></td>
<td>1. In the Grant permission to list, select <strong>User</strong>.</td>
</tr>
<tr>
<td></td>
<td>2. Type the username into the box, or click the icon to select from a list.</td>
</tr>
<tr>
<td></td>
<td>3. Click <strong>Add</strong>. The user will be added to the list on the screen, and you can then select permissions for them.</td>
</tr>
<tr>
<td>Group</td>
<td>A group already created in the Bamboo system.</td>
</tr>
<tr>
<td></td>
<td>To edit plan permissions for an existing group:</td>
</tr>
<tr>
<td></td>
<td>1. In the Grant permission to list, select <strong>Group</strong>.</td>
</tr>
<tr>
<td></td>
<td>2. Type the group name into the box, or click the icon to select from a list.</td>
</tr>
<tr>
<td></td>
<td>3. Click <strong>Add</strong>. The group will be added to the list on the screen, and you can then select permissions for the group.</td>
</tr>
</tbody>
</table>

*Screenshot: Bulk Edit Plan Permissions Wizard*
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.

To change global permissions:

1. Click Administration in the menu 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.
Related pages:
- Managing permissions
- Configuring a plan's permissions.

<table>
<thead>
<tr>
<th>User category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logged in Users</td>
<td>Users who are logged in to Bamboo.</td>
</tr>
<tr>
<td>Anonymous Users</td>
<td>Users who are not logged in to Bamboo.</td>
</tr>
<tr>
<td>User</td>
<td>A user already created in the Bamboo system.</td>
</tr>
<tr>
<td></td>
<td>To edit plan permissions for an existing user:</td>
</tr>
<tr>
<td></td>
<td>1. In the Grant permission to list, select User.</td>
</tr>
<tr>
<td></td>
<td>2. Type the username into the box, or click the icon to select from a list.</td>
</tr>
<tr>
<td></td>
<td>3. Click Add. The user will be added to the list on the screen, and you can then select permissions for them.</td>
</tr>
<tr>
<td>Group</td>
<td>A group already created in the Bamboo system.</td>
</tr>
<tr>
<td></td>
<td>To edit plan permissions for an existing group:</td>
</tr>
<tr>
<td></td>
<td>1. In the Grant permission to list, select Group.</td>
</tr>
<tr>
<td></td>
<td>2. Type the group name into the box, or click the icon to select from a list.</td>
</tr>
<tr>
<td></td>
<td>3. Click Add. The group will be added to the list on the screen, and you can then select permissions for the group.</td>
</tr>
</tbody>
</table>

You can grant the following global permissions:

<table>
<thead>
<tr>
<th>Global permission</th>
<th>Description</th>
<th>Can be granted to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Permission to view the Bamboo system. 🔄 The ability to view build plans and build results is subject to individual plan permissions.</td>
<td>- a particular user&lt;br&gt;- a particular group&lt;br&gt;- all logged-in users&lt;br&gt;- anonymous users</td>
</tr>
<tr>
<td>Create Plan</td>
<td>Permission to create new build plans.</td>
<td>- a particular user&lt;br&gt;- a particular group&lt;br&gt;- all logged-in users</td>
</tr>
</tbody>
</table>
### Admin

Permission to:
- access the Bamboo Administration menu
- create plans
- delete plans

The 'Admin' global permission also includes all plan permissions, for every plan.

⚠️ The 'Restricted Admin' global permission used in JIRA Studio is equivalent to this permission but has the following restrictions:

- Local agents cannot be created because JIRA Studio only uses Elastic Agents.
- Users and groups are managed by JIRA Studio.

### 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.

<table>
<thead>
<tr>
<th>Users</th>
<th>Access</th>
<th>Create Plan</th>
<th>Admin</th>
</tr>
</thead>
<tbody>
<tr>
<td>luke</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Groups</th>
<th>Access</th>
<th>Create Plan</th>
<th>Admin</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamboo-admin</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other</th>
<th>Access</th>
<th>Create Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>All logged in users</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Anonymous users</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

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.

**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.

---

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
3. Select the **Access** check box for ‘Anonymous users’.
4. Click **Save**.

⚠ 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**

![Global Permissions screenshot]

### 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 [JIRA Studio](#) 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 JIRA Studio only uses [elastic agents](#).
- Users and groups are managed by JIRA Studio.

### 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.

**Related pages:**

- [Global security and permission properties](#)
- [Using Captcha for failed logins](#)

**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.

**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 users’ 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**: 3

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.

**Related pages:**

- [Global security and permission properties](#)

### 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**.

### 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**

Plugins

A plugin is an add-on to the core Bamboo code that extends the Bamboo functionality. Some plugins are shipped with Bamboo and others are available for you to install yourself.

Examples of Bamboo plugins are:

- JIRA plugin
- NAnt Builder plugin
- Clover plugin

You can develop additional plugins (see the Bamboo Plugin Guide) or download existing plugins from the Atlassian...
an Plugin Exchange, and install them into your Bamboo system.

Managing your plugins

- Installing a plugin
- Upgrading your existing plugins
- Checking plugin compatibility for Bamboo upgrades
- Configuring plugins
- Disabling or enabling a plugin
- Viewing the plugin audit log
- Viewing your installed plugins
- Uninstalling a plugin
- Plugin blacklist

About the Universal Plugin Manager (UPM)

The Universal Plugin Manager (UPM) provides you with a powerful and user-friendly interface to manage your plugins. The Universal Plugin Manager itself is a plugin, which contains a number of modules that are implementations of the Atlassian REST plugin module type. It allows you to perform common plugin tasks, such as:

- Enabling/disabling plugins and their plugin modules.
- Installing new plugins.
- Configuring advanced plugin options.
- Finding out-of-date plugins and updating them.
- Checking the compatibility of your installed plugins against newer versions of the application.

The Universal Plugin Manager also interfaces with the Atlassian Plugin Exchange, so you can browse the wide range of plugins available for your application from within your application. You can install any of these plugins with a single click, or upload your own plugins using the Universal Plugin Manager as well.

Screenshot: The Universal Plugin Manager in Bamboo
Notes

- **Plugin Safety.** Plugins are very powerful. They can change the behaviour of almost any part of the Bamboo server. This makes it very important that you trust a plugin before you install it. Always be aware of where and who a plugin comes from.
- **Troubleshooting.** Having problems with the Universal Plugin Manager? Try the [Universal Plugin Manager FAQ](#). (The link will direct you to the Universal Plugin Manager documentation. Use the back button on your browser to return the Bamboo documentation).

Installing a plugin

This page describes how to install a plugin in Bamboo. You can use plugins to customise and extend the functionality of Bamboo.

A number of plugins are available from the [Atlassian Plugin Exchange](#). You can also create your own as described in the [Bamboo Plugin Guide](#).

**On this page:**

- Adding a plugin from the Atlassian Plugin Exchange
- Uploading your own plugin
- Notes

**Related pages:**

- Plugins

**Adding a plugin from the Atlassian Plugin Exchange**
Go to the Plugin Manager in Bamboo:
1. Log in as a user with the 'Admin' global permission.
2. Click Administration in the top navigation bar.
3. Click Plugin Manager, in the left-hand panel, under 'Plugins'.

To find a plugin in the Atlassian Plugin Exchange and add it to Bamboo:
1. Click the 'Install' tab in the UPM. You will see a list of featured plugins.
2. Search for your plugin as follows:
   - Enter some keywords that describe the plugin in the 'Search the Plugin Exchange' search box and press 'Enter'.
   - Alternatively, browse to the desired plugin in the list. You can choose 'Featured', 'Popular', 'Supported' (by Atlassian) or 'All available' from the 'Plugins to show' dropdown to see a different list of plugins.
3. Click the 'Install' button for the desired plugin to add it to your application. A confirmation message and the plugin details will appear when the plugin is installed successfully.
   Note: You may need to restart your application for your change to take effect. The Universal Plugin Manager will inform you if this is the case.
   Note: Not all plugins can be automatically installed. Some required manual installation. These plugins will have a 'Download' button instead of an 'Install' button. In these cases, you should read and follow the plugin's installation instructions.

Now, restart Bamboo with the plugin installed.

Uploading your own plugin

Go to the Plugin Manager in Bamboo:
1. Log in as a user with the 'Admin' global permission.
2. Click Administration in the top navigation bar.
3. Click Plugin Manager, in the left-hand panel, under 'Plugins'.

To upload your own plugin to Bamboo:
1. Click the 'Install' tab in the UPM. You will see a list of featured plugins.
2. Click the 'Upload Plugin' link. The 'Upload Plugin' window will appear.
3. Specify the location of your plugin:
   - If the plugin you want to install is on your computer, use the 'Browse' dialogue to choose the plugin JAR file.
   - If you want to install a plugin from a remote location, enter the URL of the plugin JAR file in the 'From this URL' text box.
4. Click the 'Upload' button to upload and enable your plugin. A confirmation message will appear when the plugin is successfully installed.
   Note: You may need to restart your application for your change to take effect. The Universal Plugin Manager will inform you if this is the case.

Now, restart Bamboo with the plugin installed.

Screenshot: Uploading a new plugin
Notes

- In Bamboo, you can install and uninstall both version 1 and version 2 plugins using the Universal Plugin Manager. You will see an Install or an Uninstall button.
- Some entries that you find listed in the Universal Plugin Manager are not actually plugins. These entries will show a Download button that allows you to download the application to your desktop and run it.

Upgrading your existing plugins

Plugins are often developed separately from Bamboo. You may wish to upgrade your plugins to more recent versions to allow them to work with your Bamboo version or simply to take advantage of new features in a plugin version. The Universal Plugin Manager (UPM) provides you with a list of plugins that have available upgrades and allows you to upgrade each plugin individually or in bulk.

On this page:

- Upgrading a plugin
- Upgrading all your plugins
- Notes

On this page:

- Plugins

Upgrading a plugin

Go to the Plugin Manager in Bamboo:
1. Log in as a user with the ‘Admin’ global permission.
2. Click Administration in the top navigation bar.
3. Click Plugin Manager, in the left-hand panel, under ‘Plugins’.

To upgrade a plugin in Bamboo:
1. Click the ‘Upgrade’ tab. The plugin upgrades page will appear.
   - If there is a later version of a plugin that you have already installed, this page will show the latest compatible version of the plugin.
   - You can click the plugin name to expand the row and see more information about the plugin.
   - You can filter your list by entering keywords in the ‘Filter plugins’ text box.
2. Click the ‘Upgrade Now’ button next to the relevant plugin to update it to the plugin version shown.
Upgrading all your plugins

Go to the Plugin Manager in Bamboo:
1. Log in as a user with the 'Admin' global permission.
2. Click Administration in the top navigation bar.
3. Click Plugin Manager, in the left-hand panel, under 'Plugins'.

To upgrade all available plugins in Bamboo:
1. Click the 'Upgrade' tab. The plugin upgrades page will appear.
   • If there is a later version of a plugin that you have already installed, this page will show the latest compatible version of the plugin.
   • You can click the plugin name to expand the row and see more information about the plugin.
   • You can filter your list by entering keywords in the 'Filter plugins' text box.
2. Click the 'Upgrade All' button to update every plugin to the plugin versions shown.
   Note: Some plugins cannot be installed via the Universal Plugin Manager. You must install these plugins manually. These plugins will not be upgraded automatically.

Notes
• If you are considering upgrading Bamboo, you can use the Universal Plugin Manager to check the compatibility of your plugins with your desired Bamboo version. See Checking plugin compatibility for Bamboo upgrades.

Checking plugin compatibility for Bamboo upgrades
You can use the plugin manager's 'Upgrade Check' to verify that your plugins will still work after a Bamboo upgrade.

For example, if you are thinking of upgrading from Bamboo 3.1 to Bamboo 3.2, the upgrade check can tell you the following:

• Installed plugins that are compatible with Bamboo 3.1 and Bamboo 3.2.
• Installed plugins that are not compatible with Bamboo 3.2 but will be compatible with Bamboo 3.2 if you upgrade them.
• Installed plugins that are not compatible with Bamboo 3.2, even if you upgrade them to their latest version.

Related pages:
• Plugins
• Configuring plugins

Go to the Plugin Manager in Bamboo:
1. Log in as a user with the 'Admin' global permission.
2. Click Administration in the top navigation bar.
3. Click Plugin Manager, in the left-hand panel, under 'Plugins'.

To check the compatibility of your plugins against different Bamboo versions:
1. Click the 'Upgrade Check' tab.
2. In the 'Check compatibility for' dropdown menu, select the version of your application to check the plugins against.
3. Click the 'Check' button.
4. The page display any of your installed plugins that are not compatible with the selected application version. The compatibility checker will also check the compatibility of the latest available version of each plugin (if not already upgraded). You can click on the name of any of the plugins to view more information about the plugin.
The plugins are grouped into sections under the following headings:

- **Incompatible** – The installed versions of these plugins are not compatible with the selected application version. There are currently no plugin upgrades available that are compatible with the selected application version.

- **Compatible, if upgraded** – The installed versions of these plugins are not compatible with the selected application version. However, the plugins will be compatible if you upgrade them. There are buttons allowing you to upgrade these plugins.

- **Compatible if both <the application> and the plugin are upgraded** – The installed versions of these plugins are not compatible with the selected application version. There is a plugin compatible with the newer application version, but it is not compatible with the application version you are currently running. You must upgrade the application and then upgrade the plugin. There are buttons allowing you to disable these plugins before proceeding with the upgrade.

- **Compatible** – The currently installed versions of these plugins are compatible with the selected application version.

- **Unknown** – These plugins may or may not be compatible with the selected application version. If a plugin is not registered with the Atlassian Plugin Exchange, the Universal Plugin Manager cannot check its compatibility with different application versions.

### Configuring plugins

A number of Bamboo plugins offer advanced configuration options. If you have one of these plugins installed in your application, you can view and update these configuration options via the Universal Plugin Manager (UPM).

If you would like to disable or enable a plugin, please refer to [Disabling or enabling a plugin](#).

<table>
<thead>
<tr>
<th>On this page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring a plugin</td>
</tr>
<tr>
<td>Configuration notes for specific plugins</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Related pages:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins</td>
</tr>
<tr>
<td>Checking plugin compatibility for Bamboo upgrades</td>
</tr>
</tbody>
</table>

#### Configuring a plugin

**Go to the Plugin Manager in Bamboo:**

1. Log in as a user with the 'Admin' global permission.
2. Click **Administration** in the top navigation bar.
3. Click **Plugin Manager**, in the left-hand panel, under 'Plugins'.

**To configure a plugin in Bamboo:**

1. Click the 'Manage Existing' tab.
2. Locate the plugin that you want to configure and click its title. The plugin details section will expand.
3. Click the 'Configure' link for that plugin.
   - The link will be disabled if the plugin is disabled.
   - If there is no 'Configure' link, then there are no advanced configuration options available for that plugin.
4. The advanced configuration options for the plugin will appear. Update the configuration settings as desired and save your changes.

**Note:** The advanced configuration screens are provided by the plugin. If you encounter any problems after you click the 'Configure' link, the plugin is responsible for the issue, not the Universal Plugin Manager.

#### Configuration notes for specific plugins
Please refer to the following pages about configuring specific plugins:

- [Enabling the Clover plugin](#)

**Enabling the Clover plugin**
This page contains instructions on enabling and configuring the Atlassian Clover plugin for a job in Bamboo.

To enable the Clover plugin on a job:
1. Navigate to the desired job, as described on Configuring jobs.
2. Choose Actions > Configure Job.
3. Click Miscellaneous.
4. Select Use Clover to collect Code Coverage for this build and complete the following settings:

**Related pages:**
- [Using Bamboo with Clover](#)
- [Viewing the Clover code-coverage for a build](#)
- [Viewing the Clover code-coverage for a job](#)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Automatically integrate Clover into this build</strong></td>
<td>You will need to provide a Clover license (evaluation licenses are available), unless this has been configured globally in the Administration panel (Administration &gt; Plugins &gt; Clover Plugin).</td>
</tr>
<tr>
<td><strong>Generate a Clover Historical Report</strong></td>
<td>Displays the current coverage results compared with previous Clover code coverage reports.</td>
</tr>
<tr>
<td><strong>Generate a JSON report</strong></td>
<td>Provides the Clover results in a format ready for embedding into applications or external report views.</td>
</tr>
<tr>
<td><strong>Use plan-defined Clover license key</strong></td>
<td>Override the global Clover license for this particular plan.</td>
</tr>
<tr>
<td><strong>Clover is already integrated into this build</strong></td>
<td>Use this option when you already have Clover-for-Ant or Clover-for-Maven configured to generate a report.</td>
</tr>
<tr>
<td><strong>Clover XML Location</strong></td>
<td>Specify the location where Bamboo will look for the XML report file from Clover. Please specify the file path relative to your plan’s root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. do not specify an absolute path.</td>
</tr>
</tbody>
</table>

Screenshot: Enabling Clover for a job
Disabling or enabling a plugin

The Universal Plugin Manager (UPM) allows you to disable a plugin in Bamboo without permanently removing it. You can also enable any plugins that have been previously disabled. If you want to add or remove a plugin from your Bamboo site, please refer to Installing a plugin or Uninstalling a plugin respectively.

You can also disable all user installed plugins in Bamboo, by enabling safe mode. This may help you to diagnose a plugin-related problem more easily.

Disabling a plugin

Go to the Plugin Manager in Bamboo:
1. Log in as a user with the 'Admin' global permission.
2. Click Administration in the top navigation bar.
3. Click Plugin Manager, in the left-hand panel, under 'Plugins'.

To disable a plugin in Bamboo:

1. Click the ‘Manage Existing’ tab. You will see a list of the plugins installed in your application. Enabled plugins will have this icon: 
2. Locate the plugin that you want to disable and click the title to expand the plugin details section.
3. Click the ‘Disable’ button.
4. Once a plugin has been disabled, you may need to restart your application for your change to take effect. If so, you will see a message for the plugin, 'Disabled, requires restart'. Once the plugin is fully disabled, you will see an 'Enable' link for the plugin.

Enabling a plugin

Go to the Plugin Manager in Bamboo:
1. Log in as a user with the 'Admin' global permission.
2. Click Administration in the top navigation bar.
3. Click **Plugin Manager**, in the left-hand panel, under 'Plugins'.

**To enable a plugin in Bamboo:**

1. Click the 'Manage Existing' tab. You will see a list of the plugins installed in your application. Disabled plugins will have this icon: 
2. Locate the plugin that you want to enable and click the title to expand the plugin details section.
3. Click the 'Enable' button.
4. Once a plugin has been enabled, you *may* need to restart your application for your change to take effect. If so, you will see a message for the plugin, 'Enabled, requires restart'. Once the plugin is fully disabled, you will see a 'Disable' link for the plugin.

**Disabling or enabling all user installed plugins (Safe Mode)**

Running your application in safe mode disables all user installed plugins at once. By "user installed plugins", we mean plugins that were not shipped with Bamboo but were installed later via the UPM.

All plugins that were disabled when you entered safe mode will be re-enabled when you exit safe mode.

**Go to the Plugin Manager in Bamboo:**

1. Log in as a user with the 'Admin' global permission.
2. Click **Administration** in the top navigation bar.
3. Click **Plugin Manager**, in the left-hand panel, under 'Plugins'.

**To enable safe mode in Bamboo:**

1. Click the 'Manage Existing' tab. You will see a list of the plugins installed in your application.
2. Click the 'Enable Safe Mode' button.
3. Click the 'Continue' button in the confirmation window. All user installed plugins will be disabled and your application will now be running in 'Safe Mode'.
4. You can now make changes to your installed plugins as desired. For example, you may want to enable/disable specific plugins or plugin modules.
5. Exit safe mode by clicking one of the links in the Safe Mode banner:
   - Click 'Exit Safe Mode and restore the previous configuration' to restore your plugin configuration to its state before you entered Safe Mode.
   - Click 'Exit Safe Mode and keep the current configuration' to keep all changes made to your plugin configuration during Safe Mode.

**Viewing the plugin audit log**

The Universal Plugin Manager (UPM) keeps a log of all plugin activity for your Bamboo site. Such activities may be adding plugins, enabling plugins, and so on. You can adjust the period of time for which log entries are kept.

On this page:

- Viewing the plugin audit log
- Configuring the plugin audit log

Related pages:

- Plugins

**Viewing the plugin audit log**

**Go to the Plugin Manager in Bamboo:**

1. Log in as a user with the 'Admin' global permission.
2. Click **Administration** in the top navigation bar.
3. Click Plugin Manager, in the left-hand panel, under 'Plugins'.

To view the plugin audit log:

1. Click the 'Audit Log' tab. The plugin audit log will appear, showing the 25 most recent entries.
2. Use the arrows if you want to view older entries.
3. Click the orange RSS icon if you want to receive the audit log activity in an RSS feed.

Configuring the plugin audit log

Go to the Plugin Manager in Bamboo:
1. Log in as a user with the 'Admin' global permission.
2. Click Administration in the top navigation bar.
3. Click Plugin Manager, in the left-hand panel, under 'Plugins'.

To configure the length of time for which log entries are kept:

1. Click the 'Audit Log' tab. The plugin audit log will appear.
2. Click the 'Configure purge policy' link.
3. In the 'Purge audit log after' field, specify the number of days for which you wish to keep the logs.
4. Click the 'Confirm' button.

Viewing your installed plugins

Using the Universal Plugin Manager (UPM) you can see a list of plugins installed on your Bamboo site. These include plugins that are bundled with Bamboo as well as any plugins that you have installed. Both enabled and disabled plugins are displayed.

On this page:
- Viewing your installed plugins
- Viewing a plugin's details
- Notes

Related pages:
- Plugins

Viewing your installed plugins

To view your installed plugins:

1. Click the Manage Existing tab. You will see a list of the plugins installed in your application.
   - The plugins are grouped into 'User-installed Plugins' and 'System Plugins'.
   - You can filter your list by entering keywords in the 'Filter visible plugins' text box.
   - The list of 'System Plugins' will be hidden by default. Click the Show System Plugins link to see them.

   - Enabled plugins will have this icon: ✨ Disabled plugins will have this icon: ✭
   - Click the name of a plugin to view the plugin's details.
   - Click Enable Safe Mode to run your application in safe mode. This mode disables all user installed plugins.

Note that you can also see a list of the plugins that are enabled on your Bamboo server by clicking Administration and then System Information (under ‘System’).

Viewing a plugin’s details
You can view the details for a plugin by clicking the name of a plugin in the list of installed plugins. The summary contains a short description of the plugin as well as links to plugin operations and related information.

- **Plugin key** – A unique key that identifies each plugin in the system.
- **Developer** — The name of the plugin developer and a link to the developer’s home page, if provided by the plugin developer.
- **Plugin version** – The version of the plugin currently installed.
- **Manage plugin modules** — Click this link to display the plugin modules below the plugin summary. A module is a component of the plugin. This link will only appear if the plugin has modules. If you want to enable or disable a plugin module, hover your mouse over the module and click the ‘Enable’/’Disable’ button for that module.
- **Configure** – Click this link to display the configuration settings for the plugin. This link is disabled if the plugin is disabled. Please note that not all plugins have settings that can be configured through the Universal Plugin Manager.
- **Disable** – Click this button to disable the plugin in your application. This button will only appear if the plugin is enabled.
- **Enable** – Click this button to enable the plugin in your application. This button will only appear if the plugin is disabled.
- **Uninstall** – Click this button to uninstall the plugin from your application. This button will only appear for user installed plugins.

**Notes**

- **What is the difference between a 'system plugin' and a 'user installed plugin'?**
  - System plugins are shipped with the application. These plugins are essential for the functioning of the system. Although you can disable some of them, you should not do so unless instructed by an Atlassian support engineer. *Note:* Not every system plugin can be disabled. You cannot uninstall any system plugins.
  - User installed plugins are those which have been installed in the application after it was set up. You can install a plugin either by uploading a JAR file or via the Universal Plugin Manager. You can uninstall these plugins.

**Uninstalling a plugin**

If you wish to completely remove a plugin from Bamboo, you can uninstall it via the Universal Plugin Manager (UPM). If you only want to temporarily remove it, you may wish to disable the plugin instead.

**Go to the Plugin Manager in Bamboo:**
1. Log in as a user with the 'Admin' global permission.
2. Click **Administration** in the top navigation bar.
3. Click **Plugin Manager**, in the left-hand panel, under 'Plugins'.

**To uninstall a plugin from Bamboo:**

1. Click the ‘Manage Existing’ tab. You will see a list of the plugins installed in your application.
2. Click the name of the plugin that you wish to uninstall. The plugin details will appear.
3. Click the ‘Uninstall’ button. The information summary will display an 'Uninstalling' message and the plugin will be uninstalled from your application.

**Related Topics**

- [Plugins](#)

**Plugin blacklist**

Outdated plugins may break certain functionality in Bamboo. If Bamboo detects the presence of a non-working plugin 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 plugin 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 Git Plugin, please remove the plugin from your Bamboo installation, and manually reconfigure each plan that was using it to use the Bamboo Git Plugin that is distributed with Bamboo.

**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

**Locating important directories and files**

The information on this page describes how to find important Bamboo directories and files.

<table>
<thead>
<tr>
<th>On this page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Bamboo server installation directory</td>
</tr>
<tr>
<td>- Bamboo server home directory</td>
</tr>
<tr>
<td>- Bamboo agent home directory</td>
</tr>
</tbody>
</table>

**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.)

<p>| webapp/WEB-INF/classes/bamboo-init.properties | This file tells Bamboo where to find the <em>Bamboo home directory</em>. 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. |</p>
<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conf/wrapper.conf</td>
<td>This file is used to configure Bamboo on startup, when using the Java Service wrapper under Linux or Windows.</td>
</tr>
<tr>
<td>scripts/</td>
<td>This directory contains operational scripts, including scripts for CVS and SVN triggers.</td>
</tr>
<tr>
<td>wrapper/*</td>
<td>This directory contains the necessary files to start Bamboo using the Java Service wrapper (see the Mac and Linux installation guides).</td>
</tr>
<tr>
<td>logs/*</td>
<td>This directory contains logs 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.)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>webapp/</td>
<td>This directory contains all the Bamboo server application files.</td>
</tr>
<tr>
<td>webapp/WEB-INF/lib/</td>
<td>This directory is used when deploying Bamboo plugins. It also contains other libraries required by Bamboo.</td>
</tr>
<tr>
<td></td>
<td>This is Bamboo’s logging configuration file.</td>
</tr>
<tr>
<td></td>
<td>This is the configuration file for Jetty, the application server that is bundled with Bamboo distribution.</td>
</tr>
</tbody>
</table>

* 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.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>artifacts/PLAN_KEY/shared/build-BUILD_NUMBER/</td>
<td>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 BUILD_NUMBER will always have a minimum of 5 digits, having the number completed with zeros when necessary. For instance, for build &quot;42&quot; the number will be &quot;00042&quot;.</td>
</tr>
<tr>
<td>bamboo.cfg.xml</td>
<td>This is Bamboo’s core configuration file. It includes the configuration information for connecting to Bamboo’s database.</td>
</tr>
<tr>
<td>Directory</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>xml-data/</td>
<td>This directory contains all files relating to source repositories and build results.</td>
</tr>
<tr>
<td>xml-data/build-dir/JOB_KEY</td>
<td>This is known as the Working Directory. 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.</td>
</tr>
<tr>
<td>xml-data/builds/</td>
<td>This is known as the Build Directory. 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.</td>
</tr>
<tr>
<td>xml-data/builds/JOB_KEY/results</td>
<td>Contains the build results for all the builds belonging to the JOB_KEY plan. Each build result is an individual XML file. Do not edit these files or the corresponding information in the database may become corrupt.</td>
</tr>
<tr>
<td>xml-data/builds/JOB_KEY/download-data</td>
<td>Contains the logs for each build belonging to the JOB_KEY plan.</td>
</tr>
<tr>
<td>xml-data/configuration/</td>
<td>This directory 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.</td>
</tr>
<tr>
<td>database/</td>
<td>This directory contains Bamboo's embedded HSQL database. The database contains plan configurations and some build results data.</td>
</tr>
<tr>
<td>index/</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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. By default, this directory is located under the xml-data directory in the **Bamboo home directory**.

**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 -

   ```xml
   ....
   <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**
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.

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
8. At Step 3 of the Setup Wizard (see screenshot below), select Import existing data and specify the export file created in Step 1 above.
9. Wait while Bamboo imports your data. (You will not need to complete any more steps of the Setup Wizard.)
10. When the data import has finished, restart your new instance Bamboo.
11. Re-index your Bamboo data as described in Optimising or re-indexing data.
12. 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).

**To re-index or optimise Bamboo's build results data:**

1. Click **Administration** in the top navigation bar.
2. Click **Indexing** in the left navigation column, under 'System'.
3. 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:**

1. Click **Administration** in the top navigation bar.
2. Click **Scheduled Backups** in the left navigation column (under 'System').
3. Click **Edit** to modify the schedule settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable scheduled backups</td>
<td>This check box must be cleared for automatic backups to be performed.</td>
</tr>
<tr>
<td><strong>Backup Artifacts</strong></td>
<td>Select if you want to include <a href="#">build artifacts</a> in your scheduled backups.</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **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 `bamboo.paths.set.allowed` system property to do this.  
*Note that:*  
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. |
| **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. |

4. 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:**

1. Click **Administration** in the top navigation bar.  
2. 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.  
3. Click **Edit** to edit the current 'Scheduled Backup Details'.  
4. Select the **Disable scheduled backups** check box.  
5. Click **Save**.

_Screenshot: Scheduled backups_
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 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.
- 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.
To export data for backup:

1. Click **Administration** in the top navigation bar.
2. Click **Export** in the left navigation column.
3. Specify the absolute **Export Directory Path** to which Bamboo will export its data. For example, 'C:\Documents and Settings\<me>\bamboo-home\my-backups' for a Windows-based operating system. You would typically use forward-slashes (without drive letter specification) on UNIX-based operating systems.
4. Edit the default **File Name** of the file to which Bamboo will export, if necessary.
5. Select the **Export Artifacts** check box if you want to export your build artifacts.
6. Click the **Export**.

**Screenshot: Exporting data for backup**

---

**Importing data from backup**

The instructions on this page describe how to import data from a Bamboo backup.

Before you begin:

- 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.
- Bamboo will be unavailable until the import process is complete, which may take some time.
- 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 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:

1. Click **Administration** in the top navigation bar.
2. Click the **Import** in the left navigation column (under 'System').
3. Type the absolute **File Path** from which Bamboo is to import data. For example, "/opt/bamboo/bamboo-home/export.zip" on UNIX-based operating systems.
4. Select the **Backup data** check-box. (This is highly recommended.)
5. Specify the absolute **Backup Directory Path** to which Bamboo will backup its data, if required. This must be different from the **File Path** specified above. For example, "C:\Documents and Settings\<me>\bamboo-home\my-backups" for a Windows-based operating system.
6. Type the **File Name** of the file to which Bamboo will export its data.
7. Click **Import**.
8. After the import is complete,
   - check the paths of your builders and JDK.
   - **index your data**.

**Screenshot: Importing data from a backup**
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

---

Documentation for Bamboo 4.0

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
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 menu bar.
2. Click Build Expiry (under ‘Plans’) in the left-hand panel.
3. Click Enable.
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.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build results</td>
<td>All build results data (including artifacts and build logs) are deleted.</td>
</tr>
<tr>
<td>Artifacts</td>
<td>Only user-defined artifacts are deleted from the build results.</td>
</tr>
<tr>
<td>Build logs</td>
<td>Only build logs are deleted from the build results.</td>
</tr>
<tr>
<td>Expiry period</td>
<td>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.</td>
</tr>
<tr>
<td>Minimum builds to keep</td>
<td>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.</td>
</tr>
<tr>
<td>Labels to keep</td>
<td>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.</td>
</tr>
</tbody>
</table>

6. Click Save.

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 menu 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
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

No content found for label(s) security-resources.

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

<table>
<thead>
<tr>
<th>Online Remote Agents</th>
<th>Offline Remote Agents</th>
<th>Agent Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Below is a list of IP addresses and corresponding unique IDs. You may approve agents pending approval or revoke approval from previously approved agents.

<table>
<thead>
<tr>
<th>Agent IP</th>
<th>Agent Unique ID (UUID)</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.20.7.206</td>
<td>d549e666-faeo-4787-8b17-3773e49cf3de</td>
<td>Waiting</td>
<td>Approve Access</td>
</tr>
</tbody>
</table>

### 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).

Please be warned that 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.

To mitigate some of these security risks, Elastic Bamboo incorporates an SSL tunneling implementation to provide a secure communication channel between your Bamboo server and the EC2. This tunneling implementation encrypts traffic between the Bamboo server and elastic agents using SSL, which means that you do not need to compromise your firewall by allowing inbound connections - all connections will be initiated from the Bamboo Server to the EC2 instance.

SSH tunnelling 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.

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
- Setting up your 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').

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.

Setting up your VCS for Elastic Bamboo

We recommend that you take the following steps to ensure that your VCS is set up securely for Elastic Bamboo:

1. Make your VCS accessible to the public internet
2. Use VCS authentication and access control
3. Use encrypted connections to VCS

1. Make your VCS accessible to the public internet

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**

As you have made your VCS available to the public internet, we highly recommend that you secure access to your VCS by enabling the authentication and access control features on your VCS. The instructions for doing this vary from VCS to 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). Again, the instructions for doing this vary from VCS to VCS. Please consult the documentation for your VCS for details.

**Bamboo cookies**

Bamboo uses [Seraph](https://seraph.net), 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](https://en.wikipedia.org/wiki/HTTP_cookie).

---

**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
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:

<table>
<thead>
<tr>
<th>Cookie</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJS.conglomerate.cookie</td>
<td>Track which general tabs are open and closed</td>
</tr>
<tr>
<td>BAMBOO-AGENT-FILTER</td>
<td>Date range to show the builds for agents</td>
</tr>
<tr>
<td>BAMBOO-BUILD-FILTER</td>
<td>Date range to show the builds</td>
</tr>
<tr>
<td>BAMBOO-LOG-REFRESH</td>
<td>Log refresh interval in seconds</td>
</tr>
<tr>
<td>BAMBOO-MAX-DISPLAY-LINES</td>
<td>Maximum # of lines to show on the live logs page</td>
</tr>
<tr>
<td>atlassian.bamboo.dashboard.tab.selected</td>
<td>Which tab is selected on the dashboard</td>
</tr>
<tr>
<td>bamboo.author.view</td>
<td>Which tab is selected on the Authors tab</td>
</tr>
<tr>
<td>bamboo.build.groupby.type</td>
<td>Which time group-by period is used in the reports</td>
</tr>
<tr>
<td>bamboo.dash.display.toggle</td>
<td>The ids of the projects that are expanded on the dashboard</td>
</tr>
</tbody>
</table>

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 adresses, 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

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.

Please note that you cannot set up client certificates in Bamboo due to limitations with Active
MQ. Thus, while encryption works both ways between the server and client in this
recommended configuration, authentication is only one-way. That is, the clients (i.e. agents)
can authenticate the server, but the server will not be able to authenticate the clients (i.e.
agents).

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 atruststore (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.
     
     ```
     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.
     
     ```
     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,

```bash
java -jar bamboo-agent-2.0-SNAPSHOT.jar <agentserverURL>
```

including the following command line parameters,

```bash
-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.

```bash
http://192.168.3.235:8085/agentServer/
```

For example,

```bash
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.

---

### Installing and upgrading Bamboo

#### Release notes

- [Bamboo release notes](#)
- [Bamboo release summary](#)

No content found for label(s) bambooreleasenotes.

#### Security advisories

- [Bamboo security advisories](#)

No content found for label(s) securityadvisory.

#### 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

No content found for label(s) bambooupgradeguide.

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.

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 to 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
3. Installation and setup

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 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.
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. Click the tab for your operating system, click the 'Show all' link to display the WAR file and select the Bamboo WAR file to download.

2. Deploy onto your application server by using either of the following methods:
   - Place the WAR file directly into the webapps folder of Tomcat. Tomcat will perform all the necessary extractions when it starts.
   - Extract the WAR file to your chosen directory in the webapps folder. 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.
- **Solaris** users will need to use GNU tar to handle the long filenames.

ℹ️ By default, the WAR file will extract to a folder called Bamboo-<version>. The name of the directory in the webapps folder will form the URL required to access Bamboo (e.g. Tomcat/webapps/bamboo-1.0/ will become http://host:port/bamboo-1.0/). You may wish to change the directory name for a more concise access URL.

2. Set the Bamboo Home

Set your Bamboo Home Directory. You can do this in one of three ways:

- Set the bamboo.home property in the file /WEB-INF/classes/bamboo-init.properties to your chosen Bamboo home directory.
- Pass the Bamboo home directory to the application server as a java opt. (eg. -Dbamboo.home=C:/bamboo/bamboo-home).
- 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 /WEB-INF/classes/bamboo-init.properties file:
bamboo.jms.broker.uri=tcp://localhost:5466

- 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. Set 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](https://docs.oracle.com/javase/7/docs/technotes/guides/vm/perf.html).

In Tomcat, you can set the above Java OPTs as follows:

**Windows:**

1. Find the setenv.bat file.
2. Assign the desired properties to the JAVA_OPTS variable:

```
set JAVA_OPTS=-server -XX:MaxPermSize=256m
-Dbamboo.home=/opt/bamboo/bamboohome -Xmx512m -Djava.awt.headless=true
%JAVA_OPTS%
```

**Linux-based systems:**

1. Find the setenv.sh file
2. Assign the desired properties to the JAVA_OPTS variable:

```
JAVA_OPTS="-server -XX:MaxPermSize=256m
-Dbamboo.home=/opt/bamboo/bamboohome -Xmx512m -Djava.awt.headless=true
$JAVA_OPTS"
export JAVA_OPTS
```

5. 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:
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](https://confluence.atlassian.com/).

### 6. Configure Bamboo

1. Shut down and then restart your application server.
3. Configure Bamboo via the Setup Wizard which will display. Read [Running the Setup Wizard](https://confluence.atlassian.com/) 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](https://confluence.atlassian.com).

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:

   ```xml
   <Manager useHttpOnly="true"/>
   ```

   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 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.

⚠️ **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 — via a startup script or via 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.

Alternatively, you can start Bamboo via 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 will need to fire 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 via the Setup Wizard which will display. 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.

⚠️ 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/](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.

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:

      - **Bamboo installation 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

**Warning:** 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 via the Setup Wizard which will display. 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:

License Key:

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.

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.

Step 2. General Configuration

⚠️ 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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>See <a href="#">Specifying Bamboo’s Title</a></td>
</tr>
<tr>
<td>Base URL</td>
<td>See <a href="#">Specifying Bamboo’s URL</a></td>
</tr>
<tr>
<td>Configuration Directory</td>
<td>The location for Bamboo configuration files.</td>
</tr>
<tr>
<td>Build Data Directory</td>
<td>The location for Bamboo project data files.</td>
</tr>
<tr>
<td>Build Working Directory</td>
<td>The location of project files checked out from source control.</td>
</tr>
<tr>
<td><strong>Broker URL</strong></td>
<td>Only visible if you are permitted remote agents under your Bamboo license.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>- 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.</td>
</tr>
<tr>
<td></td>
<td>- If port number <code>54663</code> is already in use, specify a different port number.</td>
</tr>
</tbody>
</table>

**Screenshot: General Configuration**

**General Configuration**

Please specify the following settings for the Bamboo Server.

**What is the title of this Bamboo instance?**

Name:  
Bamboo instance name.

**What is the server’s address?**

**Base URL:**  
http://testuser/8080/bamboo  
This is the base URL of the installation of Bamboo. All links created (for example, for email etc.) will be prefixed by this URL. For example, “http://testuser/8080/bamboo”.

**System Paths and Directories:**

**Configuration Directory:**  
Users/testuser/dwdir/8080/bamboo-trunk-testenv/bamboo-home/8080/xml-data/configuration  
The configuration directory in which your settings for Bamboo will be stored. For example, “/Users/testuser/dwdir/8080/bamboo-trunk-testenv/bamboo-home/8080/xml-data/configuration”.

**Build Data Directory:**  
Users/testuser/dwdir/8080/bamboo-trunk-testenv/bamboo-home/8080/xml-data/builds  
The build directory where the build data such as results and artifacts are stored. For example, “/Users/testuser/dwdir/8080/bamboo-trunk-testenv/bamboo-home/8080/xml-data/builds”.

**Build Working Directory:**  
Users/testuser/dwdir/8080/bamboo-trunk-testenv/bamboo-home/8080/xml-data/build-dir  
The location of project files checked out from source control. For example, “/Users/testuser/dwdir/8080/bamboo-trunk-testenv/bamboo-home/8080/xml-data/build-dir”.

**Remote Agent Communication**

**Broker URL:**  
http://testuser.sydney.atlassian.com:54663/?t=awt&FatFormat=master&duration=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.

**Step 3. Choose a Database Configuration**
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

---

Step 4. Database Configuration

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

---

Step 5. Starting Data
This step applies to the 'Custom Installation' method only.

Screenshot: Starting Data

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

**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**
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

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
```
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 JBDC

On the 'Database Configuration' page of the Setup Wizard, ensure that **Direct JDBC connection** has been selected and make the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Driver Class Name</strong></td>
<td>Type <code>org.postgresql.Driver</code> (if different from the default).</td>
</tr>
<tr>
<td><strong>Driver Class Name</strong></td>
<td>Type the URL where Bamboo will access your database (if different from the default). For details about syntax, please refer to the <a href="#">Postgres JDBC driver documentation</a>.</td>
</tr>
<tr>
<td><strong>User Name</strong></td>
<td>Type the username that Bamboo will use to access your database.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Type the password (if required) that Bamboo will use to access your database.</td>
</tr>
<tr>
<td><strong>Overwrite existing data</strong></td>
<td>Select if you wish Bamboo to overwrite any tables that already exist in the database.</td>
</tr>
</tbody>
</table>

*Screenshot 1: Setup JDBC Connection (PostgreSQL)*
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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNDI name</td>
<td>Type the JNDI name of your datasource, as configured in your application server.</td>
</tr>
<tr>
<td></td>
<td>If java:comp/env/jdbc/DataSourceName does not work, try jdbc/DataSourceName (and vice versa).</td>
</tr>
<tr>
<td>Overwrite existing data</td>
<td>Select if you wish Bamboo to overwrite any tables that already exist in the database.</td>
</tr>
</tbody>
</table>

Screenshot 2: Setup Datasource Connection
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:
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
- If the MySQL database and Bamboo servers are on the same physical computer, you can use `localhost` and **not set a password** by omitting `IDENTIFIED BY 'password'` from the 2nd MySQL statement above (if you trust the security **within** this computer).

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 — via JDBC or via a datasource. JDBC is generally simpler and is the recommended method.

**Connecting via JDBC**

To connect Bamboo to a MySQL database, via 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):

<table>
<thead>
<tr>
<th><strong>Driver Class Name</strong></th>
<th><strong>Type</strong> com.mysql.jdbc.Driver (if different from the default).</th>
</tr>
</thead>
</table>
### Database URL

Type the URL where Bamboo will access your database (if different from the default). Your URL must include the `autoReconnect=true` flag.

- If you intend to use non-Latin characters in Bamboo, ensure that your URL includes the `useUnicode=true` and `characterEncoding=utf8` flags.
- If your MySQL database server is configured to use a storage engine other than InnoDB by default, ensure that your URL includes the `sessionVariables=storage_engine=InnoDB` flag.

If you include all of these flags, your **Database URL** should look similar to:

```
jdbc:mysql://localhost/bamboo?autoReconnect=true&useUnicode=true&characterEncoding=utf8&sessionVariables=storage_engine=InnoDB
```

*If the `autoReconnect=true` 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.

<table>
<thead>
<tr>
<th>User Name</th>
<th>Type the username that Bamboo will use to access your database. This is <code>bamboouser</code> defined in section 1 (above).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>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.</td>
</tr>
</tbody>
</table>

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)*
Connecting via a datasource

To connect Bamboo to a MySQL database, via 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:
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.

On this page:
- Configuring Oracle
- Connecting using JDBC
- Connecting using a datasource

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 Bamboo. You do not have to download and install the driver.

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).
ember 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.

```sql
CREATE USER bamboo-user IDENTIFIED BY password;
```

3. Ensure that the user has the following permissions:

```sql
GRANT CONNECT, RESOURCE, CREATE TABLE TO bamboo-user;
```

### Connecting using JBDC

To connect Bamboo to an 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.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Class Name</td>
<td>Type: <code>oracle.jdbc.driver.OracleDriver</code></td>
</tr>
<tr>
<td>Database URL</td>
<td>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.</td>
</tr>
<tr>
<td>Username</td>
<td>Type the username that Bamboo will use to access your database.</td>
</tr>
<tr>
<td>Password</td>
<td>Type the password that Bamboo will use to access your database.</td>
</tr>
</tbody>
</table>

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)**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Class Name</td>
<td><code>oracle.jdbc.driver.OracleDriver</code></td>
</tr>
<tr>
<td>Database URL</td>
<td><code>jdbc:oracle:thin:@localhost:1521:SID</code></td>
</tr>
<tr>
<td>Username</td>
<td>The username to access the database</td>
</tr>
<tr>
<td>Password</td>
<td>(optional) Enter the password if the database configuration requires it</td>
</tr>
<tr>
<td><strong>Overwrite existing data</strong></td>
<td>If you wish Bamboo to overwrite any existing tables that may exist in the database.</td>
</tr>
</tbody>
</table>

### Connecting using a datasource

---

*Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.*
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 'Set Up Database 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. 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. Click Continue.

**Screenshot Setup Datasource Connection**

![Screenshot Setup Datasource Connection](image)

## 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

---

Created by Atlassian in 2012. Licensed under a [Creative Commons Attribution 2.5 Australia License](#).
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**

⚠ 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:

  ```sql
  ALTER DATABASE <database name>
  SET READ_COMMITTED_SNAPSHOT ON
  WITH ROLLBACK IMMEDIATE;
  ```

  To verify the changes, use this query which should result in '1':

  ```sql
  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.

- If you are planning to support Unicode in Bamboo please [enable unicode settings for SQL Server](https://confluence.atlassian.com/software-release-notes/1094606)

  *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):

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Class Name</td>
<td>Type net.sourceforge.jtds.jdbc.Driver (if different from the default)</td>
</tr>
<tr>
<td>Database URL</td>
<td>The URL where Bamboo will access your database, e.g. jdbc:jtds:sqlserver://localhost:1433/&lt;database&gt;. If you are connecting to a Named Instance, you will need to append <em>instance=mssqlnamehere</em> to the connection string, where <em>mssqlnamehere</em> is the name of your named instance. For more details about syntax, please refer to the Microsoft SQL Server documentation.</td>
</tr>
<tr>
<td>Username</td>
<td>The username that Bamboo will use to access your database.</td>
</tr>
<tr>
<td>Password</td>
<td>The password that Bamboo will use to access your database.</td>
</tr>
</tbody>
</table>

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

**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.


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

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 CONF-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 JBDC
  - 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

---

**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:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Class Name</td>
<td>The classname of your JDBC driver (consult your JDBC driver documentation for details).</td>
</tr>
<tr>
<td>Database URL</td>
<td>The URL where Bamboo will access your database (consult your JDBC driver documentation for details).</td>
</tr>
</tbody>
</table>
### User Name
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 not supported by Atlassian. 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.

<table>
<thead>
<tr>
<th>Database</th>
<th>Dialect</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>net.sf.hibernate.dialect.DB2Dialect</td>
</tr>
<tr>
<td>DB2 AS/400</td>
<td>net.sf.hibernate.dialect.DB2400Dialect</td>
</tr>
<tr>
<td>DB2 OS390</td>
<td>net.sf.hibernate.dialect.DB2390Dialect</td>
</tr>
<tr>
<td>Oracle 9/10g</td>
<td>net.sf.hibernate.dialect.Oracle9Dialect</td>
</tr>
<tr>
<td>Oracle (other versions)</td>
<td>net.sf.hibernate.dialect.OracleDialect</td>
</tr>
<tr>
<td>Sybase</td>
<td>net.sf.hibernate.dialect.SybaseDialect</td>
</tr>
<tr>
<td>Sybase Anywhere</td>
<td>net.sf.hibernate.dialect.SybaseAnywhereDialect</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>net.sf.hibernate.dialect.SQLServerDialect</td>
</tr>
<tr>
<td>SAP DB</td>
<td>net.sf.hibernate.dialect.SAPDBDialect</td>
</tr>
<tr>
<td>Informix</td>
<td>net.sf.hibernate.dialect.InformixDialect</td>
</tr>
<tr>
<td>Ingres</td>
<td>net.sf.hibernate.dialect.IngresDialect</td>
</tr>
<tr>
<td>Progress</td>
<td>net.sf.hibernate.dialect.ProgressDialect</td>
</tr>
<tr>
<td>Mckoi SQL</td>
<td>net.sf.hibernate.dialect.MckoiDialect</td>
</tr>
<tr>
<td>Interbase</td>
<td>net.sf.hibernate.dialect.InterbaseDialect</td>
</tr>
<tr>
<td>Pointbase</td>
<td>net.sf.hibernate.dialect.PointbaseDialect</td>
</tr>
</tbody>
</table>
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.

Bamboo remote agent installation guide

This page describes how to install the Bamboo Remote Agent manually.

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.
- 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.
- 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.
1. Click \texttt{Administration} in the menu bar.

2. Click \texttt{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.

3. Click \texttt{Install Remote Agent}. The following screen will be displayed:

   \hspace{1cm}
   \textbf{Installing a Remote Agent}
   \hspace{1cm}

   To install a remote agent, please follow these instructions:

   1. Ensure that you have Java Runtime Environment 5.0 or later installed on the agent machine.
   2. Download the remote agent JAR file to a directory on the agent machine.

   \hspace{1cm}
   \textbf{Running a Remote Agent}
   \hspace{1cm}

   Once installed, you can run the remote agent by executing the following command line from the directory containing the remote agent JAR file:

   \begin{verbatim}
   java -jar atlassian-bamboo-agent-installer-3.0-SNAPSHOT.jar
   http://bamboo-host-server:8085/agentServer/
   \end{verbatim}

   This will start a service wrapper for your agent, which will automatically restart in case of failure. You may also add extra system properties like \texttt{-Dbamboo.home=...} to customise the agent's home location. For more information, see our Bamboo Remote Agent Installation Guide.

   \hspace{1cm}
   \textbf{Running the agents without the service wrapper}
   \hspace{1cm}

   For customers wanting to run the Bamboo agent without the service wrapper, the direct agent JAR is available at \texttt{bamboo-agent-3.0-SNAPSHOT.jar}. For more information about using the JAR directly, please consult our documentation.

5. Click \texttt{DOWNLOAD Remote Agent JAR} and save the JAR file to the directory 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.

\section*{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:

\begin{verbatim}
java -jar atlassian-bamboo-agent-installer-2.2-SNAPS HOT.jar
http://bamboo-host-server:8085/agentServer/
\end{verbatim}

\begin{itemize}
\item \textbf{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.}
\end{itemize}

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.
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:

```
```

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:

```
```

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
   ```

2. Click the 'the direct agent JAR is available at bamboo-agent-2.2.2.jar.' link and save 'classic' agent jar.

3. 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...
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-SNAPS
HOT.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.

(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-SNAPS
HOT.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 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 menubar.
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.

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:

```
```

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:
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:
  
  ```sh
  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 [Sun documentation](https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardSecurityConfig.html).

**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](#).

**Hardware sizing considerations**

> Also see the [Bamboo System Requirements](#).

**On this page:**

- [General information](#)
- [Database connection pool size](#)
- [Configuring the connection pool limit](#)
- [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 amount of database connections available to Bamboo is the the lower of two values: your DBMS connection limit and the configuration of connection pool on Bamboo size. 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 the default limit. Note: having too many connections available to Bamboo carries no performance penalty as long as your DBMS can handle the load.

Configuring the connection pool limit

Bamboo's connection limit can be modified by altering the following value in your bamboo.cfg.xml file:

```
<property
  name="hibernate.c3p0.max_size">25</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:

\[(\text{Concurrent users})/5 + (\text{Busy remote agents})/5 + (\text{Local agents})\times1.1 + (\text{Amount of concurrent change detections})\]

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 release notes
Latest Release

✅ Bamboo 4.0.1 has been released. Read the Bamboo 4.0.1 Release Notes and Upgrade Guide.

Don't have Bamboo 4.0? 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

No content found for label(s) bamboorelesenotes.

All Beta Releases

No content found for label(s) bamboobetarelesenotes.

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 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

Bamboo 1.0 — 20 February 2007
- All Plans tab
- More in release notes

Bamboo 4.0 Release Notes
27 March 2012
Atlassian is proud to present Bamboo 4.0 with automated build branches and branch merging, quarantined tests and many more improvements.

Upgrading to Bamboo 4.0 is free for all customers with active Bamboo software maintenance.

 Highlights of this release:

- Automatically Build Branches
- Automatically Merge Branches using Gatekeeper and Branch Updater
- Manage Build Failures by Quarantining Intermittent Tests
- Fresh New User Experience
- Plus Over 100 Fixes and Improvements

Thank you for your feedback:

⭐ 55 new features and improvements implemented
⭐ 106 votes fulfilled

Please keep logging your votes and issues. They help us decide what needs doing, and are much appreciated!
Upgrading to Bamboo 4.0

You can download Bamboo from the [Atlassian website](https://atlassian.com). If upgrading from a previous version, please read the [Bamboo 4.0 Upgrade Guide](https://confluence.atlassian.com/display/.bamboo/Bamboo+4.0+Upgrade+Guide).

1

Automatically Build Branches

Git and Mercurial have made creating and merging branches extremely fast and easy, allowing developers to work on features, bug fixes and other improvements without conflicts. In the past, most continuous integration systems were configured to only build the 'master' or mainline branch, not the branch where the developer is actually working. This left the developer and their team uncertain if the branch changes actually worked or not.

Now, with Plan Branches, Bamboo can build branches without having to duplicate the build configuration for each branch. You can have Bamboo watch Git and Mercurial repositories and create plan branches automatically when a developer creates a branch. This allows feedback on changes without the hassle of manually configuring a new build for every branch.

More...

2

Automatically Merge Branches using Gatekeeper and Branch Updater

Remove integration uncertainty in your build when developing on Git and Mercurial branches. With our new support for automatic merging, Bamboo provides two methods to automatically test and merge your code on every change to ensure that your branch will integrate perfectly when it's time to merge your changes.

Using the Gatekeeper model, Bamboo automatically merges work on a feature branch back into trunk and pushes it if the build passes.
This is great for teams who are working on bug fixes on a separate branch or small features that can be included in the project as soon as they are completed.

The **Branch Updater** model, Bamboo automatically merges work from another specified branch into the current branch. This works fabulously for situations where a branch needs to be kept up to date with changes from master or another branch.

More...

### Manage Build Failures by Quarantining Intermittent Tests

Fans of "Freakonomics" know about the **Broken Window Theory**: breakages that are left visible tend to invite further destruction and neglect.

The world of software is no different. Builds with flaky or ever-failing tests tend to fall into severe decay because breakages become the norm. That's why teams adopt the discipline of either fixing the problem right away, or pulling the test out. But if the test was worth writing, it's worth keeping tabs on, and Bamboo's new Quarantine feature makes that easy.

Simply hit the **Quarantine** button next to a failing test and let Bamboo do the dirty work. Not only that, but you'll see the count of Quarantined tests in every build result as a reminder to reincorporate them into your build. Gone are the days of commenting out test code or manually updating test suite configuration files.

More...

### Fresh New User Experience

Springtime means spring cleaning, so we've freshened up the UI in Bamboo 4.0. Don't want to see the **Plan Navigator** all the time? Toggle it hidden or shown to keep content-packed pages free of clutter and easy to read.

On the **Plan Summary** pages, you'll notice that the Plan Statistics panel has undergone a facelift. And just under that, you'll find a list of all branches for the plan, complete with build status indicators!
History buffs and stats junkies alike will rejoice in the new bar on the plan detail and build history configuration pages that shows the status of the last 10 builds. Hover your mouse over any indicator to see the build number what triggered the build and whether any tests failed.

Plus Over 100 Fixes and Improvements

The top 10 issues by votes are shown below. Please refer to our public JIRA site to see the full list of fixes and improvements in this release of Bamboo.

<table>
<thead>
<tr>
<th>JIRA Issues (10 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>![ ]</td>
</tr>
<tr>
<td>![ ]</td>
</tr>
<tr>
<td>![ ]</td>
</tr>
<tr>
<td>![ ]</td>
</tr>
<tr>
<td>![ ]</td>
</tr>
<tr>
<td>![ ]</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>BAM-1940</td>
</tr>
<tr>
<td>BAM-1146</td>
</tr>
<tr>
<td>BAM-10677</td>
</tr>
<tr>
<td>BAM-10888</td>
</tr>
</tbody>
</table>

The Bamboo 4.0 Team

Development

Core Team

Brydie McCoy
Jason Berry
Marek Went
Krystian Brazulewicz
Przemek Bruski
Marcin Gardias
Piotr Stefan Stefaniak
Nathan Wilson

Team Lead

Mark Chaimungkalanont

Project Manager

Anton Mazkovoi

Support

Ajay Sridhar
Armen Khachatryan
Bamboo 4.0.1 Release Notes

13 April 2012

The Atlassian Bamboo team is proud to announce the release of Bamboo 4.0.1.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 4.0.1 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 4 yet?

Take a look at all the new features in the Bamboo 4.0 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 4.0 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 4.0.1 are shown below.

<table>
<thead>
<tr>
<th>JIRA Issues (12 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
</tbody>
</table>

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Reporter</th>
<th>Assignee</th>
<th>Status</th>
<th>Fixed By</th>
<th>Fixed Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-113 99</td>
<td>Repository Null Error when attempting to delete repo</td>
<td>Unassigned</td>
<td>Gretchen Jones</td>
<td>Resolved</td>
<td>Apr 09, 2012</td>
<td></td>
</tr>
<tr>
<td>BAM-113 96</td>
<td>Links in Bamboo application pointing to documentation pages are broken</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Armen Khachaturyan [Atlassian]</td>
<td>Resolved</td>
<td>Apr 06, 2012</td>
<td></td>
</tr>
<tr>
<td>BAM-113 88</td>
<td>Switching from Mercurial to Git Repository throws exception</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Resolved</td>
<td>Apr 04, 2012</td>
<td></td>
</tr>
<tr>
<td>BAM-113 87</td>
<td>OAuth token expires every hour</td>
<td>Unassigned</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Resolved</td>
<td>Apr 04, 2012</td>
<td></td>
</tr>
<tr>
<td>BAM-113 77</td>
<td>Attempting to remove a repository results in a null error</td>
<td>Unassigned</td>
<td>Jens Schumacher [Atlassian]</td>
<td>Resolved</td>
<td>Apr 02, 2012</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Issue ID</td>
<td>Description</td>
<td>Assignee(s)</td>
<td>Resolution</td>
<td>Fixed 1</td>
<td>Fixed 2</td>
</tr>
<tr>
<td>-----</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------------------------</td>
<td>------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>3</td>
<td>BAM-11354</td>
<td>Fail to authenticate BAM-11354</td>
<td>Unassigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11340</td>
<td>Git SubModules flag not propagated to build</td>
<td>Brydie McCoy [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11328</td>
<td>Bamboo should be able to work with mercurial subrepositories, even if they use relative paths.</td>
<td>Piotr Stefan Stefaniak [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11324</td>
<td>Changing repository url turns off the &quot;automatic branch detection&quot;.</td>
<td>Brydie McCoy [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11311</td>
<td>Changing repository url turns off the &quot;automatic branch detection&quot;.</td>
<td>Piotr Stefan Stefaniak [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11324</td>
<td>Changing repository url turns off the &quot;automatic branch detection&quot;.</td>
<td>Piotr Stefan Stefaniak [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11311</td>
<td>Changing repository url turns off the &quot;automatic branch detection&quot;.</td>
<td>Piotr Stefan Stefaniak [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11311</td>
<td>Changing repository url turns off the &quot;automatic branch detection&quot;.</td>
<td>Piotr Stefan Stefaniak [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11311</td>
<td>Changing repository url turns off the &quot;automatic branch detection&quot;.</td>
<td>Piotr Stefan Stefaniak [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11311</td>
<td>Changing repository url turns off the &quot;automatic branch detection&quot;.</td>
<td>Piotr Stefan Stefaniak [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAM-11311</td>
<td>Changing repository url turns off the &quot;automatic branch detection&quot;.</td>
<td>Piotr Stefan Stefaniak [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Documentation for Bamboo 4.0

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
Bamboo 4.0 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 4.0 from a previous version of Bamboo. For details on the Bamboo 4.0 release, see the Bamboo 4.0 Release Notes.

Please follow the Bamboo 4.0-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
- Upgrading from Bamboo 3.4 to 4.0
- Upgrading from Bamboo prior to 3.4
- Developing for Bamboo 4.0
- Checking for Known Issues and Troubleshooting the Bamboo Upgrade

Upgrade Notes

The following upgrade notes are specific to 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.

Upgrading from Bamboo 3.4 to 4.0

To upgrade to Bamboo 4.0, following the appropriate instructions below:

- 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.

Upgrading from Bamboo prior to 3.4

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 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 you custom EBS, see Updating EBSes created for Fedora to support Amazon Linux.

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
• 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 4.0. 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 will need to set aside time, as described in the Bamboo 2.0 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 will need to 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 4.0

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 4.0 guide, which outlines changes in Bamboo 4.0 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 3.4 Release Notes

14 December 2011

Atlassian is proud to present Bamboo 3.4 with improved Git support, stronger agent security, global repositories, enhanced email templates and many more improvements.

Upgrading to Bamboo 3.4 is free for all customers with active Bamboo software maintenance.

Highlights of this release:

• Git Submodule Support
• Shared Repositories
• Agent Security Improvements
• New Email Templates
• Elastic Bamboo support for Microsoft Windows®
• Plus over 100 fixes and improvements
Thank you for your feedback:

🌟 107 new features and improvements implemented
🌟 56 votes fulfilled

Please keep logging your votes and issues. They help us decide what needs doing, and are much appreciated!

### Upgrading to Bamboo 3.4

You can download Bamboo from the [Atlassian website](https://www.atlassian.com/software). If upgrading from a previous version, please read the [Bamboo 3.4 Upgrade Guide](https://confluence.atlassian.com/display/BA/Upgrading+to+Bamboo+3.4).

---

#### Git Submodule Support

By supporting native Git clients in addition to its embedded Git implementation, Bamboo now unlocks the full power of Git while still making it easy to get started. The new support for Git Submodules allows you to structure your projects the way you want, and makes it easy to build multimodule projects.

It is still simple to get started with Git. If you don’t have a native Git client installed on your agent, Bamboo will automatically fall back to its embedded Git implementation. To use the improved Git support, simply let Bamboo know where the Git executable is located.

---

#### Shared Repositories

In Bamboo 3.3 we made it easy to add multiple repositories, and now in 3.4 we make it easy to share them. Are you using the same repository in multiple plans and are tired of keeping them all in sync when the configuration of the repository changes? With Shared Repositories you can define repositories globally and share the configuration with as many plans as you want. When you update the configuration, the change will automatically be picked up by all plans that use the repository definition. More...
Agent Security Improvements

Bamboo 3.4 now provides a way to verify that remote agents are allowed to connect to the Bamboo server. Bamboo prevents unknown agents from connecting to the server. Agents now need to be manually approved by an administrator before they can communicate with the server in any way. Note that Elastic agents do not have to be approved. This improvement means that sensitive information on the Bamboo server is now much more secure. More...
New Email Templates

We've made the Bamboo email notification a whole lot easier on the eyes. The new design makes the email much easier to read and allows you to see all the important information about the build at a glance.

Elastic Bamboo support for Microsoft Windows®

If you've ever wanted to use Elastic Bamboo to test your application in Internet Explorer 9, or for testing .NET applications, Bamboo is able to help you do this quickly and cheaply in the Cloud using Amazon EC2. Start the new Windows 2008 64bit server Elastic Bamboo image with a single click, and get testing on Windows in minutes. More...

Plus over 100 fixes and improvements

The top 10 issues by votes are shown below. For the full list of fixes and improvements, please refer to our public JIRA site to see a full list of issues fixed in this release of Bamboo.
<table>
<thead>
<tr>
<th>Ticket</th>
<th>Description</th>
<th>Votes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-8106</td>
<td>Git Repository should support submodules</td>
<td>26</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5206</td>
<td>Elastic Bamboo support for Windows/.Net images</td>
<td>12</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-6361</td>
<td>Support for x86_64 using default ami</td>
<td>10</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1956</td>
<td>SVN:Externals changes are not picked up in sub-folders</td>
<td>5</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-10280</td>
<td>Upgrade Fedora version on EC2 Default Image</td>
<td>4</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1647</td>
<td>Cannot create export files larger than 4G</td>
<td>3</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-8743</td>
<td>GitHub support only seems to support personal and public organisation repos and not private organisation repos</td>
<td>3</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-6238</td>
<td>Bamboo does not connect to eJabberd 2.1.3</td>
<td>3</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-10015</td>
<td>Agent class server blacklists JIRA plugin needed for deserialisation of build message</td>
<td>2</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-9960</td>
<td>Upgradetask 2704 fails if there are marked for deletion plans in the database</td>
<td>1</td>
<td>Resolved</td>
</tr>
</tbody>
</table>

The Bamboo 3.4 Team
Development
Core Team
Bamboo 3.4.1 Release Notes

21 December 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.4.1.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.4.1 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.4 yet?

Take a look at all the new features in the Bamboo 3.4 Release Notes and see what you are missing out on!
Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the [Bamboo 3.4 Upgrade Guide](#).

Updates and Fixes in this Release

The issues addressed in Bamboo 3.4.1 are shown below. To view the list in JIRA, please refer to our main [JIRA site](#).

<table>
<thead>
<tr>
<th>JIRA Issues (7 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td><img src="#" alt="GitHub repository configuration doesn't persist credentials" /></td>
</tr>
</tbody>
</table>
Bamboo 3.4.2 Release Notes

28 December 2011

The Atlassian Bamboo team is happy to announce the release of **Bamboo 3.4.2**.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.4.2 is of course free to all customers with active Bamboo software maintenance.

**Don't have Bamboo 3.4 yet?**

Take a look at all the new features in the Bamboo 3.4 Release Notes and see what you are missing out on!

![Download latest version]

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 3.4 Upgrade Guide.

**Updates and Fixes in this Release**

The issues addressed in Bamboo 3.4.2 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (2 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
</tbody>
</table>

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
Bamboo 3.4.3 Release Notes

12 January 2012

The Atlassian Bamboo team is happy to announce the release of Bamboo 3.4.3.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.4.3 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.4 yet?

Take a look at all the new features in the Bamboo 3.4 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.4 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.4.3 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (12 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>--------</td>
</tr>
</tbody>
</table>

BAM-105 58

Error when deleting plan The DELETE statement conflicted with the REFERENCE constraint "FKEC405ED86EAFB613". The conflict occurred in database "testowa", table "dbo.BRS_CONSUMED_SUBSCRIPTION", column 'ARTIFACT_LINK_ID'.

Marcin Gardias [Atlassian]

Marcin Gardias [Atlassian]
<table>
<thead>
<tr>
<th></th>
<th>BAM-105</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Expand all/Collapse all is very slow on large dashboards</strong></td>
<td>Przemek Bruski [Atlassian]</td>
<td>Przemek Bruski [Atlassian]</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>
Bamboo 3.4.4 Release Notes

22 February 2012

The Atlassian Bamboo team is happy to announce the release of Bamboo 3.4.4.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.4.4 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.4 yet?

Take a look at all the new features in the Bamboo 3.4 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.4 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.4.4 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Authenticate to retrieve your issues</td>
</tr>
</tbody>
</table>

Bamboo 3.4 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 3.4 from a previous version of Bamboo. For details on the Bamboo 3.4 release, see the Bamboo 3.4 Release Notes.

Please follow the Bamboo 3.4-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.
Upgrade Notes

The following upgrade notes are specific to Bamboo 3.4.

- EC2 Security - when using Elastic Bamboo it’s no longer necessary to open any inbound ports to your Bamboo Server instance. To increase security of your server, please remove any firewall exceptions you may have added.
- EC2 Security Groups - if you’ve customised the security groups for Bamboo by removing ingress rules, note that as of Bamboo 3.4, Bamboo will keep the security group synchronised with its default setting - i.e. you will not be able to remove SSH, RDP and Bamboo tunnel ingress rules from the security group definition. You will still be able to add new ingress rules and your existing rules will be kept.
- Improved Git support - to use the improved Git support, configure a Git capability on the agent. If you don’t have a native Git client installed on your agent, Bamboo will automatically fall back to its embedded Git implementation.
- Bamboo WAR instances installed in Tomcat - after running the upgrade process, make sure you will apply the recommendations from this KB article.

Upgrading from Bamboo 3.3 to 3.4

To upgrade to Bamboo 3.4, following the appropriate instructions below:

- 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.

Upgrading from Bamboo prior to 3.3

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 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 you 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 3.4. If you are using a version of Bamboo earlier than 2.6.3, we recommend that you upgrade to it 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 (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 3.4

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 3.4 guide, which outlines changes in Bamboo 3.4 that may affect Bamboo plugins compiled for Bamboo version 3.1.x or earlier.

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 3.3 Release Notes
11 October 2011

Atlassian is proud to present Bamboo 3.3 with support for Multiple Repositories, Reload-able Plugins, multiple source aliases and many more improvements.

Upgrading to Bamboo 3.3 is free for all customers with active Bamboo software maintenance.

Highlights of this release:

• Multiple Source Repositories
• Reload-able Plugins
• Source Repository User Aliases
• Automatic Agent Upgrades
• Fast, history-friendly tabbed navigation
• Commit Centric View
• Plus over 170 fixes and improvements

Thank you for your feedback:

⭐ 35 new features and improvements implemented
⭐ 218 votes fulfilled

Your votes and issues help us keep improving our products, and are much appreciated.

Please keep logging your votes and issues. They help us decide what needs doing!
Upgrading to Bamboo 3.3

You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 3.3 Upgrade Guide.

1

Multiple Source Repositories

With Bamboo 3.3 you can now monitor and checkout code from multiple repositories. Easily build large projects, which are often composed of smaller, self-contained modules, without using externals or submodules.

The new repository configuration interface makes it easy to administrate multiple repositories and specify which of the selected repositories should trigger the build.

2

Reload-able Plugins
Reloadable Plugins make it faster and easier than ever to install or update plugins in Bamboo. Bamboo's most common module types now support Atlassian Plugin Framework 2, which allows for installing and updating plugins without restarting your Bamboo server.

Install new Tasks on the fly without interrupting your builds. And for Tasks that are hugely complex and incompatible with earlier versions, Bamboo now offers a way to safely pause your server without breaking running builds.

Source Repository User Aliases

To ensure all code changes across different repositories are attributed to the right person, Bamboo 3.3 now supports multiple user aliases. If your username is "clarkkent" in one repository, and "superman" in another one, you can map both aliases to the same user in Bamboo. That's not only handy in the DVCS world, but will also be useful for Plans with multiple source repositories. To make sure the statistics are still accurate, we have also aggregated the author statistics into a user statistic.
Automatic Agent Upgrades

Upgrading your agents manually for a new Bamboo release can be painful, especially if you have dozens of agents. With Bamboo 3.3 you no longer have to worry about your agent upgrades. Bamboo will upgrade your agents automatically. We have also massively improved the performance with which new Plugins and Classes are transferred to the agent.

Fast, history-friendly tabbed navigation

Bamboo 3.3 now shows you all the important information about your build even faster. By loading tabbed content via AJAX (instead of full-page reloads) we significantly decreased the page-load times when browsing build results. Utilising HTML5’s browser history API we keep track of which tabs (and their URLs) you had selected, so your back/forward buttons work as expected.
Commit Centric View

It's great to know what builds are broken, but for a developer it's even more important to know whether his recent commit was the cause. Bamboo 3.3 now provides a commit centric view in Bamboo, showing builds relation to an individual commit and providing a way to assess a level of overall confidence in a commit. The "My Bamboo" page now has been completely revamped and is not only a whole lot more useful, but also a whole lot prettier.
Plus over 170 fixes and improvements

The top 10 issues by votes are shown below. For the full list of fixes and improvements, please refer to our public JIRA site to see a full list of issues fixed in this release of Bamboo.

<table>
<thead>
<tr>
<th>JIRA Issues (10 issues)</th>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Votes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BAM-955</td>
<td>Allow a plan to have multiple repositories</td>
<td>69</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BAM-1141</td>
<td>All multiple repository aliases</td>
<td>25</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BAM-1015</td>
<td>Allow One Plan To Be Based On Multiple CVS Modules</td>
<td>16</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BAM-1099</td>
<td>Multiple Source Repository Aliases for a user</td>
<td>14</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BAM-1786</td>
<td>Multiple source repository aliases for a</td>
<td>13</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>
The Bamboo 3.3 Team

Development

Core Team

Brydie McCoy
James Dumay
Jason Berry
Marek Went
Krystian Brazulewicz
Przemek Bruski
Marcin Gardias
Piotr Stefan Stefaniak
Ben Woskow
Slawek Ginter
Nathan Wilson

Team Lead

Mark Chaimungkalanont

Project Manager

Anton Mazkovoi

Support

Renan Battaglin
Ajay Sridhar
Bamboo 3.3.1 Release Notes

12 October 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.3.1.

We've fixed a critical bug in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.3.1 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.3 yet?

Take a look at all the new features in the Bamboo 3.3 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.3 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.3.1 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (1 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>**Type</td>
</tr>
<tr>
<td>------</td>
</tr>
</tbody>
</table>
Bamboo 3.3.2 Release Notes

13 October 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.3.2.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.3.2 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.3 yet?

Take a look at all the new features in the Bamboo 3.3 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.3 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.3.2 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (6 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>-------</td>
</tr>
</tbody>
</table>
Bamboo 3.3.3 Release Notes

14 November 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.3.3.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.3.3 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.3 yet?

Take a look at all the new features in the Bamboo 3.3 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.3 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.3.3 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (15 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>BAM-100</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>BAM-999</td>
</tr>
<tr>
<td>BAM-998</td>
</tr>
</tbody>
</table>
Bamboo 3.3 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 3.3 from a previous version of Bamboo. For details on the Bamboo 3.3 release, see the Bamboo 3.3 Release Notes.

Please follow the Bamboo 3.3-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
- Upgrading from Bamboo 3.2 to 3.3
- Upgrading from Bamboo prior to 3.1
- Developing for Bamboo 3.3
- Checking for Known Issues and Troubleshooting the Bamboo Upgrade

### Upgrade Notes

The following upgrade notes are specific to Bamboo 3.3.

#### Multiple Repositories

**Schema changes**

The changes to Bamboo to support the Multiple Repositories feature of Bamboo 3.3 require schema changes to the database. Bamboo will automatically migrate existing data to match these changes which may take some time. On our internal system with 1200 Plans with 250,000 build results took approximately 1.5 hours for the upgrade to complete on a 8 core 2ghz Xeon E5405 system with 768m of memory allocated to Bamboo.

#### Remote Agents

**Automatic Upgrade to new Remote Agent Bootstrap**

- Remote Agents installed from a prior version of Bamboo will automatically attempt to upgrade to the new Bamboo 3.3 Remote Agent using the Bamboo 3.3 Agent Installer.
- The upgrade process may fail if the binary location of the Elastic Agent is not writable or is not running within the provided wrapper. If the upgrade fails, simply reinstall the agent using the Bamboo 3.3 Agent Installer.

**Changes to the Agent Installer**

- Bamboo now includes all the binaries necessary to run a Remote Agent inside the Agent Installer package that can be downloaded from the Agents administration page to reduce startup times.
- When the server version changes (e.g. upgrading to a new major or minor release), Bamboo will fetch the new binaries from the Bamboo server and cache them on the Remote Agent file system.

### Upgrading from Bamboo 3.2 to 3.3

To upgrade to Bamboo 3.2, following the appropriate instructions below:

- Follow the instructions in the Bamboo generic upgrade guide.
- If you've customised Amazon EC2 images to work with Bamboo, see Creating a Custom Elastic Image.

### Upgrading from Bamboo prior to 3.1
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 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 you custom EBS, see Updating EBSes created for Fedora to support Amazon Linux.

Notes for upgrading from Bamboo 2.6.x

- You will need to upgrade to Bamboo 2.7.4 before upgrading to Bamboo 3.3. If you are not running Bamboo 2.6.3, we recommend that you upgrade to it 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.

Notes for upgrading from Bamboo 2.5 or earlier

- 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.

Notes for upgrading from a version of Bamboo prior to 2.0

- 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. Please read the Bamboo 2.0 Upgrade Guide for important upgrade instructions for upgrading from earlier versions of Bamboo.

Developing for Bamboo 3.3

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 3.3 guide, which outlines changes in Bamboo 3.3 that may affect Bamboo plugins compiled for Bamboo version 3.1.x or earlier.

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 3.3 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 3.2 Release Notes**

26 July 2011

Atlassian presents Bamboo 3.2 with release management, manual Stages and improved application linking.

Upgrading to Bamboo 3.2 is free for all customers with active Bamboo software maintenance. Highlights of this release:
• Release Management
• Manual Stages
• Rerunning a Failed Stage
• Plan Filters on the Dashboard and Wallboard
• User Management via JIRA
• Improved Application Linking
• Plus over 130 fixes and improvements

Thank you for your feedback:

⭐ 51 new features and improvements implemented
⭐ 150 votes fulfilled

Your votes and issues help us keep improving our products, and are much appreciated.

Please keep logging your votes and issues. They help us decide what needs doing!

Upgrading to Bamboo 3.2

You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 3.2 Upgrade Guide.

1

Release Management

Bamboo can be used for more than just continuous deployment. The entire release process can be automated by Bamboo with the appropriate setup. If you are using Atlassian's JIRA with Bamboo, you can now synchronise the release activities between these two applications.

When releasing a version in JIRA, you will have the option of starting a Bamboo build (e.g. a build that tests and deploys the artefacts for the version). If the build passes, the version will be automatically released in JIRA. If it fails, the version will not be released and you will have the option of running it again.

We've released a new version of the JIRA-Bamboo plugin for JIRA that provides this release management functionality. Get it from the Atlassian Plugin Exchange: JIRA-Bamboo Plugin v4.3
Manual Stages

The new manual Stages feature also helps you to manage release activities, such as testing, deployment and the release itself, by allowing you pause the execution of your Plan at manual Stages. For example, you might want to use a manual Stage for the deployment to give your QA team a chance to perform a few manual tests before your software goes into production.

Any Stage can be configured as a manual Stage. If you run a Plan with manual Stages, Bamboo will pause the build every time it reaches a manual Stage. The Plan build will only continue once a user has manually triggered the Stage.
Rerunning a Failed Stage

Sometimes, it's not your developers' fault. Builds can fail for all sorts of reasons that are not related to the code (e.g. infrastructure problems). If so, you may want to rerun the Stage that a Plan failed at, rather than start the Plan build from scratch again.

We have provided you with the ability to rerun failed Stages in this release. Any Jobs that failed in the Stage will run again and the exact same data will be used.
Plan Filters on the Dashboard and Wallboard

Is your dashboard or wallboard a confusing mess of Plans? If so, you'll be happy to know that we've implemented Plan filters for the dashboard and wallboard in this release. Simply label your Plans, then filter the Plans on your dashboard/wallboard by Plan label, as desired.
5

User Management via JIRA

You can now use the same set of users in Bamboo and JIRA, and manage your users and groups in JIRA. We've redesigned the user management configuration screen to make it easy to connect Bamboo to JIRA/Crowd for user management. If you use Bamboo with Crowd, you'll also notice a few improvements to the Bamboo-Crowd integration. We've bundled the Crowd 2.3 integration libraries with Bamboo 3.2. Try using Bamboo 3.2 with Crowd 2.3.1 and you will notice an improvement in performance, particularly if you have a large user base.

More...

6

Improved Application Linking

Bamboo 3.2 bundles the new Application Links plugin. If you want to link Bamboo to JIRA to take advantage of the new release management feature, you won't need to mess around in the administration consoles of both Bamboo and JIRA any more. The Application Links plugin lets you create two-way links to other applications with your choice of authentication methods, from the Bamboo administration console.

More...
Plus over 130 fixes and improvements

The top 10 issues by votes are shown below. For the full list of fixes and improvements, please refer to our public JIRA site to see a full list of issues fixed in this release of Bamboo.

<table>
<thead>
<tr>
<th>JIRA Issues (10 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>!</td>
</tr>
</tbody>
</table>

The Bamboo 3.2 Team
Bamboo 3.2.2 Release Notes

23 August 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.2.2.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.2.2 is of course free to all customers with active Bamboo software maintenance.
Don't have Bamboo 3.2 yet?

Take a look at all the new features in the [Bamboo 3.2 Release Notes](#) and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the [Bamboo 3.2.2 Upgrade Guide](#).

Updates and Fixes in this Release

The issues addressed in Bamboo 3.2.2 are shown below. To view the list in JIRA, please refer to our main [JIRA site](#).

<table>
<thead>
<tr>
<th>JIRA Issues (12 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>BAM-9786</td>
</tr>
<tr>
<td>BAM-9461</td>
</tr>
<tr>
<td>BAM-9381</td>
</tr>
<tr>
<td>BAM-9151</td>
</tr>
<tr>
<td>BAM-3097</td>
</tr>
</tbody>
</table>
Bamboo 3.2.2 Upgrade Guide

Upgrading from Bamboo 3.2 to 3.2.2

Please follow the Bamboo generic upgrade guide.

No additional upgrade tasks are required to upgrade from Bamboo 3.2 to 3.2.2.

Upgrading from Bamboo 3.1.4 or earlier

In addition to the above, please read the Bamboo 3.2 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo upgrade guides section.

Bamboo 3.2 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 3.2 from a previous version of Bamboo. For details on the Bamboo 3.2 release, see the Bamboo 3.2 Release Notes.

Please follow the Bamboo 3.2-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
- Upgrading from Bamboo 3.1 to 3.2
- Upgrading from Bamboo prior to 3.1
- Developing for Bamboo 3.2
- Checking for Known Issues and Troubleshooting the Bamboo Upgrade

Upgrade Notes

The following upgrade notes are specific to Bamboo 3.2.

Latest versions of Chrome, Firefox and Safari now supported

We have amended our browser support policy. The latest stable versions of Chrome, Firefox and Safari are now supported. The Supported platforms page now states this, as well as listing the versions of the browser that we test against.

crowd.properties, atlassian-user.xml and crowd-ehcache.xml moved

From Bamboo 3.2 onwards, the crowd.properties, atlassian-user.xml and crowd-ehcache.xml files can be found in $BAMBOO_HOME/xml-data/configuration.

Note, copies of these files will still exist in the old BAMBOO/webapp/WEB-INF/classes directory, however you can safely remove or ignore them after you upgrade to Bamboo 3.2.

Auto-Favourite Plugin removed from Bamboo
The auto-favourite plugin has been removed from Bamboo.

EC2-related changes

- Default EC2 images are now Amazon Linux-based, if you have your own EBS, see [Updating EBSes created for Fedora to support Amazon Linux](#) for upgrade tips,
- Logging in to your instances using root account is deprecated and will be removed in future versions. Instead, use ec2-user account - this user is also able to execute sudo without password,
- Several build-related tools delivered with the images have been upgraded:
  1. JDK 6 has been upgraded to 6u26,
  2. Apache Ant has been upgraded to 1.8.2,
  3. PHPUnit has been upgraded to 3.4.15,
  4. VCSes (SVN, Mercurial, git and CVS) have been updated to the latest version available with Amazon Linux,
  5. Additional Grails versions have been installed: 1.3.4 and 1.3.7,
  6. The image now has make and GCC (gcc and g++) installed.

Upgrading from Bamboo 3.1 to 3.2

To upgrade to Bamboo 3.2, following the appropriate instructions below:

- If you are using Bamboo with [Crowd](#), follow the instructions in [Upgrading Bamboo with Crowd to Bamboo 3.2](#).
- If you are using Bamboo only, follow the instructions in the [Bamboo generic upgrade guide](#).
- If you've been using Amazon EC2 images with your custom EBS, see [Updating EBSes created for Fedora to support Amazon Linux](#) for more details.

Upgrading from Bamboo prior to 3.1

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 2.6.x

- You will need to upgrade to Bamboo 2.7.4 before upgrading to Bamboo 3.2. If you are not running Bamboo 2.6.3, we recommend that you upgrade it to 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.

Notes for upgrading from Bamboo 2.5 or earlier

- 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.

Notes for upgrading from a version of Bamboo prior to 2.0

- 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. Please read the [Bamboo 2.0 Upgrade Guide](#) for important upgrade instructions for upgrading from earlier versions of Bamboo.

Developing for Bamboo 3.2

If you are a Bamboo plugin developer, please refer to our [Changes for Bamboo 3.2](#) guide, which outlines changes in Bamboo 3.2 that may affect Bamboo plugins compiled for Bamboo version 3.1.x or earlier.

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 3.2 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.

### Updating EBSes created for Fedora to support Amazon Linux

With Bamboo 3.2, we are shipping the new Amazon Linux-based EC2 images by default (along with the old ones to ease the transition period). The Fedora release used up to now was a pretty old distribution, so despite the fact that the distributions have common roots (Fedora: RedHat, AmazonLinux: CentOS/RedHat), some changes are needed.

The idea is to keep the most popular gotchas on this page to keep the transition as smooth as possible.

#### Ephemeral storage

Ephemeral storage used to be mounted directly on /mnt, which is not a mount point according to FHS. The new mount point for the primary ephemeral storage is /media/ephemeral0 on all instance types. In case you customised your Bamboo working directory location, you should move it there. Changing bamboo-agent.cfg.xml to the following should do the trick.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration>
  <buildWorkingDirectory>/media/ephemeral0/bamboo-working-dir</buildWorkingDirectory>
</configuration>
```

Grepping your EBS for bamboo-working-dir (or even /mnt) is a very good idea.

#### Using packages from 3rd party sources

If needed, you should use the packages prepared for CentOS 5.x, which is binary compatible with Amazon Linux. The recipe for doing this is given in the Selenium paragraph.

#### Oracle

There are two issues with Using Oracle on Amazon Linux: "double tap setup technique" and swap.

**Running Oracle Setup**

There's a weird bug plaguing yum-based distros that causes the Oracle setup to fail the first time it's run. The solution is simple:
oracleRpm=$LOC/oracle-xe-univ-10.2.0.1-1.0.i386.rpm

yum -y localinstall --nogpgcheck $oracleRpm || yum -y localinstall --nogpgcheck $oracleRpm

**Adding swap space**

Oracle Express Edition (possibly others) require at least 1GB of swap space, regardless of how much RAM you have. Amazon Linux has ~800 MB swap by default. The easiest way of adding more swap is:

```sh
swapFile=/media/ephemeral0/swapfile
dd if=/dev/zero of=${swapFile} bs=1M count=1K && mkswap -f ${swapFile} && chmod 600 ${swapFile} && swapon -p -2 ${swapFile}
```

You don't need to add 1GB, like in the example - 200MB should be enough.

**PostgreSQL**

No special instructions. In case you've been using 3rd party RPMs to install a recent version of PostgreSQL, make sure you switch to the one distributed with Amazon Linux.

**MySQL**

No special instructions. In case you've been using 3rd party RPMs to install a recent version of MySQL, make sure you switch to the one distributed with Amazon Linux.

**Selenium**

In case you've been using 3rd party RPMs to install a recent version of packages needed for Selenium usage, make sure you switch to the one distributed with Amazon Linux.

Note that Firefox is not distributed with Amazon Linux. To install it, you have to add Centos 5 repositories, which are binary compatible with our Amazon Linux version. Only the packages not distributed with Amazon Linux will be pulled from the CentOS repository.

The complete script used to setup Selenium is given below.

```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=$basearch&repo=os
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-$centosMajorVersion
enabled=0

[centos-update]
name=CentOS - Updates
mirrorlist=http://mirrorlist.centos.org/?release=${centosVersion}&arch=$basearch&repo=updates
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-$centosMajorVersion
enabled=0

EOF

yum -y --enablerepo=centos-* install firefox

yum -y install xorg-x11-server-Xvfb xterm
Upgrading Bamboo with Crowd to Bamboo 3.2

The instructions on this page are for customers who are currently using Bamboo with Crowd, and want to upgrade to Bamboo 3.2. These instructions complement the instructions on the Bamboo 3.2 Upgrade Guide.

In this upgrade process, you may need to upgrade your Crowd installation in addition to upgrading Bamboo.

On this page:
- Upgrade Crowd to Crowd 2.3.1
- Testing the Crowd Upgrade
- Upgrade Bamboo to Bamboo 3.2
  - (Recommended) Method 1. Perform an “inline” upgrade without exporting/importing.
  - Method 2. Import through the administration panel.
  - Method 3. Upgrade Bamboo and import via the setup wizard

Upgrade Crowd to Crowd 2.3.1

Bamboo 3.2 ships with Crowd 2.3 integration libraries. Before upgrading Bamboo to Bamboo 3.2, you must upgrade your Crowd instance to at least Crowd 2.3.1. This will result in better performance, particularly if you have a large user base. For instructions on how to upgrade Crowd, see the Crowd Upgrade Guide.

Testing the Crowd Upgrade

If you would like to test whether Crowd 2.3.1 will work properly with your existing Bamboo installation, do the following:

1. Replace your crowd-integration-client with version 2.3.1, i.e.

```bash
rm Bamboo-3.1.4/webapp/WEB-INF/lib/crowd-integration-client*.jar
cp atlassian-crowd-2.3.1/client/crowd-integration-client-2.3.1.jar Bamboo-3.1.4/webapp/WEB-INF/lib
rm Bamboo-3.1.4/webapp/WEB-INF/classes/crowd-encache.xml
cp atlassian-crowd-2.3.1/client/conf/crowd-encache.xml Bamboo-3.1.4/webapp/WEB-INF/classes
```
2. Copy `atlassian-user-crowd-provider-3.2.jar` from the Bamboo 3.2 distribution into your old Bamboo WEB-INF/lib directory to avoid "NoClassDef" exceptions due to missing CrowdUserManager class, i.e.

```shell
cp Bamboo-3.2/webapp/WEB-INF/lib/atlassian-user-crowd-provider-3.2.jar Bamboo-3.1.4/webapp/WEB-INF/lib
```

**Upgrade Bamboo to Bamboo 3.2**

After upgrading Crowd, you can upgrade Bamboo to Bamboo 3.2 using one of the following methods:

1. **(Recommended) Method 1. Perform an "inline" upgrade without exporting/importing.**

   This method is similar to a regular Bamboo upgrade. You will install Bamboo 3.2, copy the Crowd settings over from your old Bamboo instance and point your new instance at your old Bamboo home.
   
   1. Download Bamboo 3.2 from the Bamboo download centre and install it. Do not start it.
   2. Copy the Crowd settings from your old Bamboo instance to the new Bamboo 3.2 instance, i.e.
      - `rm Bamboo-3.2/webapp/WEB-INF/classes/atlassian-user.xml`
      - `cp Bamboo-3.1.4/webapp/WEB-INF/classes/atlassian-user.xml Bamboo-3.2/webapp/WEB-INF/classes`
      - `rm Bamboo-3.2/webapp/WEB-INF/classes/crowd.properties`
      - `cp Bamboo-3.1.4/webapp/WEB-INF/classes/crowd.properties Bamboo-3.2/webapp/WEB-INF/classes`
   3. Configure your new Bamboo instance to use the old `bamboo-home (in webapp/WEB-INF/classes/bamboo-init.properties)`.
   4. Start up Bamboo.
      
      You can remove `crowd.properties, atlassian-user.xml and crowd-encache.xml files from Bamboo-3.2/webapp/WEB-INF/classes` folder after this, if you wish (as per BAM-9318).

2. **Method 2. Import through the administration panel.**

   This method requires you to install Bamboo 3.2 (including running the setup wizard), update your atlassian-user.xml and crowd.properties files in Bamboo, then import the data from your old Bamboo instance.
   
   1. Back up your existing Bamboo installation. See Bamboo generic upgrade guide for instructions.
   2. Download and install your new Bamboo 3.2 instance, including completing the setup wizard.
   3. Connect your new Bamboo 3.2 instance to your Crowd instance by editing `Bamboo-3.2-home/xml-data-configuration/atlassian-user.xml`
      
      Please note this is not the same file as `Bamboo-3.2/webapp/WEB-INF/classes/atlassian-user.xml`
      
      Your file should look like this after editing:

      ```xml
      <atlassian-user>
      <repositories>
          <crowd name='Crowd Repository' key='crowd'/>
      </repositories>
      </atlassian-user>
      ```
      
   4. Edit `Bamboo-3.2-home/xml-data-configuration/crowd.properties` and update the credentials and URLs.
      
      Please note this is not the same file as `Bamboo-3.2/webapp/WEB-INF/classes/crowd.properties`
      
      5. Restart Bamboo.
6. Do the import from the administration panel, as described in Importing data from backup.
   
   You can remove crowd.properties, atlassian-user.xml and crowd-encache.xml files from Bamboo-3.2/webapp/WEB-INF/classes folder after this, if you wish (as per BAM-9318).

Method 3. Upgrade Bamboo and import via the setup wizard

This method is similar to option 2. You are required to install Bamboo 3.2, however data is imported from your old Bamboo instance during the setup wizard. The atlassian-user.xml and crowd.properties files are updated in Bamboo, after the import.

Please be aware, that Bamboo will return error messages stating that the import has failed in this method. However, this is expected behaviour and the resultant upgrade will still work correctly.

1. Back up your existing Bamboo installation. See Bamboo generic upgrade guide for instructions.
2. Download and install your new Bamboo 3.2 instance.
3. Run the setup wizard. When you are prompted to import data, import data from your old Bamboo installation. At the end of the import, Bamboo will state that there is no user with administration privileges and will ask you to contact Atlassian support. However, your plans and builds will be properly migrated with the exception that the user management is not working yet i.e. you won't be able to log in to Bamboo.
4. Shut down the Bamboo instance.
5. Connect your new Bamboo 3.2 instance to your Crowd instance by editing Bamboo-3.2-home/xml-data-configuration/atlassian-user.xml.

Please note this is not the same file as Bamboo-3.2/webapp/WEB-INF/classes/atlassian-user.xml.

Your file should look like this after editing:

```xml
<atlassian-user>
  <repositories>
    <crowd name='Crowd Repository' key='crowd'/>
  </repositories>
</atlassian-user>
```

6. Edit Bamboo-3.2-home/xml-data/configuration/crowd.properties and update the credentials and URLs.

Please note this is not the same file as Bamboo-3.2/webapp/WEB-INF/classes/crowd.properties

7. Start Bamboo

   You can remove crowd.properties, atlassian-user.xml and crowd-encache.xml files from Bamboo-3.2/webapp/WEB-INF/classes folder after this, if you wish (as per BAM-9318).

Congratulations! You should now be able to log into your upgraded Bamboo instance and have your users managed by Crowd.

Bamboo 3.1 Release Notes

10 May 2011

Atlassian presents Bamboo 3.1 with Tasks, parameterised builds and support for Bitbucket and GitHub.

Upgrading to Bamboo 3.1 is free for all customers with active Bamboo software maintenance.

Highlights of this release:

- **Tasks**
- **Plan Variables & Parameterised Builds**
• .Net Support
• Bitbucket Support
• GitHub Support
• New Plugin Manager
• Support for Amazon EC2 Spot Instances
• Gravatar Support
• Improved Windows process handling
• Plus over 150 fixes and improvements
• The Bamboo 3.1 Team

Thank you for your feedback:

🌟 47 new features and improvements implemented
🌟 114 votes fulfilled

Your votes and issues help us keep improving our products, and are much appreciated.

Please keep logging your votes and issues. They help us decide what needs doing!

Upgrading to Bamboo 3.1

You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 3.1 Upgrade Guide.

Tasks

Tasks in Bamboo 3.1 provide developers and build engineers with another tool to design more flexible builds. Configure a Job with Tasks to build your application, execute a script, upload files to another server, create your documentation and much more. Bamboo allows you to add as many Tasks as needed for a Job, with each Task providing detailed log messages during the build. Tasks are executed against the same working directory, allowing you to perform actions like changing version numbers or copying files before a subsequent task is executed.

We've also implemented Final Tasks as part of the Tasks feature. No matter what happens in your previous tasks, Final Tasks will always be executed at the end of the build. This gives you the opportunity to clean up after your build, shutting down processes or services that you may have started as part of your build.
Plan Variables & Parameterised Builds

Parameterised Builds allow you to customise parts of your Build when the Build is run manually or via a script. We've introduced Plan variables in this release to complement the existing global variables. This allows you to change version numbers on the fly or change certain options you use within your script or commands for particular builds.

More...
.Net Support

We've completely re-written the .Net plugin for Bamboo to add Tasks for building and testing .Net projects. All Tasks take advantage of our improved Windows process handling. The following Tasks are included in the plugin:

- Visual Studio – Build Visual Studio projects with devenv.exe. The Task also allows you to switch between different architectures (x86, AMD64, IA32, IA64).
- MSBuild – Run MSBuild as part of your build.
- NAnt – Execute NAnt targets to build your project.
- MSTest Runner – Run your MSTest configuration and display the MSTest results.
- MSTest Parser – Parse and display MSTest test results.
- MBUnit Parser – Parse and display MBUnit test results.
- NUnit Parser – Parse and display NUnit test results.

The plugin is open-source. Feel free to fork it on Bitbucket.

More...

Bitbucket Support

Bamboo now supports Bitbucket. If you use Bitbucket for your source code hosting, you can use Bamboo to build any source code maintained in repositories on Bitbucket.

More...
GitHub Support

We've extended our Git support to include GitHub. If you use the GitHub for your source code hosting, you can use Bamboo to build any source code maintained in repositories on GitHub.

Source Repository

<table>
<thead>
<tr>
<th>Source Control</th>
<th>GitHub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>jschumacher</td>
</tr>
<tr>
<td>Password</td>
<td>******</td>
</tr>
<tr>
<td>Repository</td>
<td>jschumacher/speakeasy-plugin</td>
</tr>
<tr>
<td>Branch</td>
<td>master</td>
</tr>
</tbody>
</table>

More...

New Plugin Manager

Managing plugins and performing Bamboo upgrades are now much easier with the brand new plugin manager. The Universal Plugin Manager (UPM) is already bundled with JIRA and Confluence, and has now been integrated into Bamboo. With the UPM you can:

- Perform a plugin compatibility check before upgrading Bamboo.
- Install new plugins from the Atlassian Plugin Exchange.
- Manage existing plugins.
- With just one click, upgrade all plugins that have updates available.
- View and track updates via the audit log.
Support for Amazon EC2 Spot Instances

Bamboo now supports Amazon EC2 Spot Instances. If you are using Elastic Bamboo to run builds in the Amazon Elastic Compute Cloud (EC2), you can now bid for and use EC Spot Instances. This allows you to run your builds at a much lower price, provided that your bid exceeds the current "spot price" (determined by EC2 customer demand).
Gravatar Support

Bamboo 3.1 adds a personal touch to your CI environment with the introduction of Gravatar support. If your users have signed up to the Gravatar service, Bamboo will attempt to retrieve their profile pictures and display them in Bamboo. You'll see these profile pictures displayed against activities for the user, like code changes or comments on build results.

Code Changes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>James Dumay</td>
<td>Create a popup for available variables</td>
<td>(150900)</td>
</tr>
<tr>
<td>Brydie McCoy</td>
<td>Fixing ui for legacy builder configuration</td>
<td>(150890)</td>
</tr>
</tbody>
</table>

Improved Windows process handling
In previous versions of Bamboo, processes started from .bat scripts or a number of different methods were not shut down properly. We have improved the Windows process handling in Bamboo to ensure that the underlying processes and their children are stopped correctly.

Plus over 150 fixes and improvements

The top 10 issues by votes are shown below. For the full list of fixes and improvements, please refer to our public JIRA site to see a full list of issues fixed in this release of Bamboo.

<table>
<thead>
<tr>
<th>JIRA Issues (10 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
<tr>
<td>📝</td>
</tr>
</tbody>
</table>
The Bamboo 3.1 Team

Development

Core Team

Brydie McCoy
James Dumay
Jason Berry
Marek Went
Krystian Brazulewicz
Przemek Bruski
Marcin Gardias
Piotr Stefan Stefaniak
Ben Woskow

Git

Slawek Ginter

Team Lead

Mark Chaimungkalanont

Project Manager

Anton Mazkovoi

Support

Renan Battaglin
Ajay Sridhar
Zed Yap

Gurleen Anand
Felipe Kraemer
Rene Verschoor
Dylan Hansen

Others

Product Management

Jens Schumacher
Helen Hung

Product Marketing

Giancarlo Lionetti

Technical Writing

Andrew Lui

Operations

James Fleming
Bamboo 3.1.1 Release Notes

24 May 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.1.1.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.1.1 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.1 yet?

Take a look at all the new features in the Bamboo 3.1 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.1 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.1.1 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (12 issues)</th>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grails will check for tests even if there are tests not configured.</td>
<td>James Dumay [Atlassian]</td>
<td>James Dumay [Atlassian]</td>
<td>⬤</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 12, 2011</td>
<td>May 24, 2011</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Size of the</td>
<td>Unassigned</td>
<td>Krystian Brazulew</td>
<td>⬤</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 20, 2011</td>
<td>May 23, 2011</td>
</tr>
<tr>
<td>ID</td>
<td>Title</td>
<td>Author(s)</td>
<td>Resolution</td>
<td>Status</td>
<td>Created</td>
<td>Resolved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------</td>
<td>------------</td>
<td>---------</td>
<td>-----------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-876</td>
<td>NAnt home was not autodetected correctly</td>
<td>James Dumay [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 11, 2011</td>
<td>May 24, 2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-876</td>
<td>Add option to use client side mapping for p4 change detection</td>
<td>Andrew Lui [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 11, 2011</td>
<td>May 24, 2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-875</td>
<td>Polish Tasks UI according to usability testing feedback</td>
<td>Jason Berry [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 10, 2011</td>
<td>May 24, 2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-871</td>
<td>CommentManager.getCommentsByUserForPlans does not find comments</td>
<td>Arun Bhalla [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 05, 2011</td>
<td>May 24, 2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
New dialog forms should have a loading spinner while content is waiting to load

BAM-862

memory

Jason Berry [Atlassian]

Jason Berry [Atlassian]

Resolved

Fixed

Apr 28, 2011

May 24, 2011

Bamboo 3.1.1 Upgrade Guide

Upgrading from Bamboo 3.1 to 3.1.1

Please follow the Bamboo generic upgrade guide. No additional upgrade tasks are required to upgrade from Bamboo 3.1 to 3.1.1.

Upgrading from Bamboo 3.0.5 or earlier

In addition to the above, please read the Bamboo 3.1 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo upgrade guides section.

Bamboo 3.1.3 Release Notes

16 June 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.1.3.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.1.3 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.1 yet?

Take a look at all the new features in the Bamboo 3.1 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.1.3 Upgrade Guide. Note, there is no Bamboo 3.1.2.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.1.3 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (19 issues)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summar y</th>
<th>Assigne e</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resoluti on</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>🍃</td>
<td>BAM-896</td>
<td>Disabled</td>
<td>Marek</td>
<td>Przemek</td>
<td>🍃 Resol</td>
<td>Fixed</td>
<td>Jun 01,</td>
<td>Jun 16,</td>
<td></td>
</tr>
</tbody>
</table>

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
<table>
<thead>
<tr>
<th>#</th>
<th>Bamboo Issue</th>
<th>Summary</th>
<th>Resolved By</th>
<th>Resolution</th>
<th>Date Resolved</th>
<th>Date Accepted</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>BAM-904</td>
<td>NPE in Bitbucket repository</td>
<td>Mark Chaimungkalon [Atlassian]</td>
<td>Resolved</td>
<td>Jun 14, 2011</td>
<td>Sep 06, 2011</td>
</tr>
<tr>
<td>9</td>
<td>BAM-892</td>
<td>Artifacts lost when moving plans between projects</td>
<td>Marcin Gardias [Atlassian]</td>
<td>Resolved</td>
<td>May 27, 2011</td>
<td>Jun 14, 2011</td>
</tr>
<tr>
<td>1</td>
<td>BAM-888</td>
<td>Grails task: -non-interactive is added even when already</td>
<td>Marcin Gardias [Atlassian]</td>
<td>Resolved</td>
<td>May 23, 2011</td>
<td>Jun 16, 2011</td>
</tr>
<tr>
<td>Issue</td>
<td>Title</td>
<td>Assigned to</td>
<td>Resolution</td>
<td>Status</td>
<td>Assigned to</td>
<td>Status</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>Ticket</td>
<td>Description</td>
<td>Assigned To</td>
<td>Resolution</td>
<td>Fixed</td>
<td>Created</td>
<td>Updated</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
<td>-------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>BAM-887</td>
<td>After Plan has been moved to another project all artifacts are unaccessible</td>
<td>Krystian Brazulewicz [Atlassian] Krystian Brazulewicz [Atlassian]</td>
<td>Resolved</td>
<td>May 20, 2011</td>
<td>Jun 16, 2011</td>
<td></td>
</tr>
<tr>
<td>BAM-877</td>
<td>Migrate the NCover Result page to the 3.1 format</td>
<td>Piotr Stefan Stefaniak [Atlassian] Renan Battaglin [Atlassian]</td>
<td>Resolved</td>
<td>May 12, 2011</td>
<td>Jun 16, 2011</td>
<td></td>
</tr>
</tbody>
</table>

**Bamboo 3.1.3 Upgrade Guide**

Upgrading from Bamboo 3.1 to 3.1.3

Please follow the [Bamboo generic upgrade guide](#).

No additional upgrade tasks are required to upgrade from Bamboo 3.1.1 to 3.1.3. Note, there is no Bamboo 3.1.2.

Upgrading from Bamboo 3.0.5 or earlier

In addition to the above, please read the [Bamboo 3.1 Upgrade Guide](#) and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the [Bamboo upgrade guides](#) section.
Bamboo 3.1.4 Release Notes

30 June 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.1.4.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.1.4 is of course free to all customers with active Bamboo software maintenance.

Don’t have Bamboo 3.1 yet?

Take a look at all the new features in the Bamboo 3.1 Release Notes and see what you are missing out on!

Download latest version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.1.4 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.1.4 are shown below. To view the list in JIRA, please refer to our main JIRA site.

JIRA Issues (6 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summar y</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td>BAM-907</td>
<td>500 Exception after invalid login, CAPTCHA A image does not show, but user is required to enter it</td>
<td>Przemek Bruski [Atlassian]</td>
<td>Nikolaus Almassy</td>
<td>⬤</td>
<td>⬤ Resolved</td>
<td>Fixed</td>
<td>Jun 20, 2011</td>
<td>Jun 30, 2011</td>
</tr>
<tr>
<td>#</td>
<td>Issue Number</td>
<td>Description</td>
<td>Author(s)</td>
<td>Resolution</td>
<td>Created</td>
<td>Resolved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------</td>
<td>------------</td>
<td>---------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Bamboo 3.1.4 Upgrade Guide

Upgrading from Bamboo 3.1.3 to 3.1.4

Please follow the Bamboo generic upgrade guide.

No additional upgrade tasks are required to upgrade from Bamboo 3.1.3 to 3.1.4.

Upgrading from Bamboo 3.0.5 or earlier

In addition to the above, please read the Bamboo 3.1 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo upgrade guides section.

### Bamboo 3.1 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 3.1 from a previous version of Bamboo. For
details on the Bamboo 3.1 release, see the Bamboo 3.1 Release Notes.

Please follow the Bamboo 3.1-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
- Upgrading from Bamboo 3.0 to 3.1
- Upgrading from Bamboo prior to 3.0
- Developing for Bamboo 3.1
- Checking for Known Issues and Troubleshooting the Bamboo Upgrade

Upgrade Notes

Bamboo Compatibility with Subversion pre-1.6

We have changed the default behaviour of the SVNKit library in Bamboo. As a result, any source code checked out by Bamboo will be automatically upgraded to be compatible with Subversion 1.6. This does not adversely affect any pre-1.6 Subversion servers. However, if you use a pre-1.6 Subversion client to access code checked out by Bamboo, then any Bamboo builds on that code may fail.

If you want to prevent any checked out code from being automatically upgraded to SVN client format 1.6, you will need to run Bamboo with the following system property:

```
-Dbamboo.svn.wc.format=1.5
```

For more information, please see this FAQ: How do I manually set the version of new Subversion workspaces.

End of Support for Java Platform 5 (JDK/JRE 1.5)

We are ending support for Java Platform 5 (JDK/JRE 1.5) in this release. Please see End of Support Announcements for Bamboo for further details.

Conversion of Builders to Tasks

The introduction of the Tasks feature in Bamboo 3.1 means that the following activities will occur during the upgrade to Bamboo 3.1:

- Builder capabilities will be renamed to Executable capabilities.
- Builders will be converted to Tasks. The Tasks will be linked to the Job that the Builders were a part of.
- If one of your Builders cannot be matched to a Task (e.g. you are using a custom plugin), it will be converted to a ‘Compatibility Task’. The configuration for your Builder will be transferred to this Task, and the Task will be linked to the Job that the Builder was a part of. You can view/update the configuration by navigating to the Task and clicking 'Configure Legacy Executable'.

Changes to Bamboo Files/Directories for Bamboo distributions for Windows

If you are using the Bamboo distribution (not EAR-WAR) for Windows, please note that the location of the following files/directories have changed.

- All log files now located at %USERPROFILE%\bamboo.log, rather than in the logs folder of your installation directory. For Bamboo running as a Windows service, log files are located at %WINDIR%\Sys tem32\Config\systemprofile\bamboo.log. Note, existing logs will not be migrated, however new logs will be written to the new location when running Bamboo after the upgrade.
• All temporary directories on windows are now by default in %WINDIR%\Temp, rather than in the user's temporary directory.

Gravatar Support Enabled by Default

The new Gravatar support feature is enabled by default in Bamboo 3.1. For more information, see Configuring Gravatar support.

Upgrading from Bamboo 3.0 to 3.1

Before you begin, do the following:

1. Back up your existing installation of Bamboo

We strongly recommend that you do the following to back up your Bamboo installation:

• Back up your xml-data directory — See the Bamboo generic upgrade guide for instructions.
• Export your Bamboo data for backup — See the Exporting data for backup for instructions. Please note, that this may take a long time to complete depending on the number of builds and tests in your system.

2. Ensure that your plugins work

If you are using plugins, ensure that your plugins are compiled against 3.1 before upgrading.

Before you upgrade, please read the following important points that relate to Bamboo 3.0.

Upgrading from Bamboo prior to 3.0

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 2.6.x

• You will need to upgrade to Bamboo 2.7.4 before upgrading to Bamboo 3.0. If you are not running Bamboo 2.6.3, we recommend that you upgrade to it 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.

Notes for upgrading from Bamboo 2.5 or earlier

• 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.

Notes for upgrading from a version of Bamboo prior to 2.0

• 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. Please read the Bamboo 2.0 Upgrade Guide for important upgrade instructions for upgrading from earlier versions of Bamboo.

Developing for Bamboo 3.1

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 3.1 guide, which outlines changes in Bamboo 3.1 that may affect Bamboo plugins compiled for Bamboo version 3.0.x or earlier.

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 3.1 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 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 3.0 Release Notes

16 February 2011

With great pleasure, Atlassian presents Bamboo 3.0 with artifact sharing, Git support and a revamped user interface.

Upgrading to Bamboo 3.0 is free for all customers with active Bamboo software maintenance.

Highlights of this release:

- Artifact Sharing
- Git Support
- User Interface Overhaul
- Scheduled Repository Polling
- Configuration Changes Captured in Audit Logs
- Plus over 400 fixes and improvements
- The Bamboo 3 Team

Thank you for your feedback:

⭐ 40 new features and improvements implemented
⭐ 125 votes fulfilled

Your votes and issues help us keep improving our products, and are much appreciated.

Please keep logging your votes and issues. They help us decide what needs doing!

Upgrading to Bamboo 3.0

You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 3.0 Upgrade Guide.

1

Artifact Sharing

Bamboo 3.0 allows artifacts produced from a Job to be shared with other Jobs in the same Plan, without being
rebuilt every time. Build your artifacts in the first Stage and pass them through Unit and Acceptance testing Stages. When the build has completed, you will have every confidence that the final artifact has been thorough tested, works and is ready for further deployment.

Artifact sharing for Maven 2 is also supported, but in beta.

![Screenshot above: Build Artifacts](image)

**More...**

### Git Support

Bamboo now supports Git. If you use this distributed version control system (DVCS) or are thinking of migrating to it, you can use Bamboo to build any source code maintained in Git repositories.
User Interface Overhaul

In our previous release, Bamboo 2.7, we introduced Stages and Jobs to improve the way your Builds are structured. The user interface wasn't ideal for representing the new Plan structure though. In this release, we've taken the opportunity to completely overhaul the Bamboo user interface, including the Plans, Jobs and Build Results screens. If you are currently using Atlassian's JIRA, you'll feel right at home with the new look and feel!

Highlights include:

- A more neutral colour scheme to improve readability of the screens.
- Redesigned controls — slicker tabs, svelte forms and tables, Atlassian-standard headings and better buttons.
- Layout changes — functions moved into dropdowns to make key information more prominent.

Plan and Job Summary

As part of our drive to improve the Bamboo user experience, we've implemented better user interfaces for Plan and Jobs. The new Plan Navigator shows you the Stages and Jobs hierarchy in a Plan, as well as allowing quick navigation to Jobs. If you are looking at a Job, it will be highlighted in the Plan Navigator. Common functions have been moved into an 'Actions' menu for easy access. The interface also looks much cleaner, due to a better organised layout and the new colour scheme.
Build Results

The Build Results user interface for Plans and Jobs has also been improved in Bamboo 3.0. This includes a status ribbon that allows you to see whether a build was successful or not, at a glance. We've also added a history navigator that allows you to view the status of and navigate to, prior and subsequent build results. An updated layout and the new colour scheme complements these new features.

Scheduled Repository Polling

Bamboo now allows you to schedule when you want to poll your source repositories for changes. You can create a schedule using Cron expressions, but don't worry if you can't remember all that Cron lingo. Bamboo has an easy-to-use user interface that allows you to create your schedule without any expression magic required.
Configure Changes Captured in Audit Logs

All configuration changes in Bamboo are now recorded in the audit logs. This allows you to track down whether a build failed because of an actual problem in your code, or whether a Plan configuration change was responsible.

Plus over 400 fixes and improvements

The top 10 issues by votes are shown below. For the full list of fixes and improvements, please refer to our public JIRA site to see a full list of issues fixed in this release of Bamboo.

<table>
<thead>
<tr>
<th>JIRA Issues (10 issues)</th>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Votes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-2875</td>
<td>GIT support for BAMBOO</td>
<td>🔄</td>
<td>62</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-1704</td>
<td>New Build Strategy: Polling the repository at a fixed time</td>
<td>🔄</td>
<td>22</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-2496</td>
<td>Conditional, cron-based scheduling</td>
<td>🔄</td>
<td>11</td>
<td>Resolved</td>
</tr>
<tr>
<td>Ticket</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1104</td>
<td>You should be allowed to edit build artifacts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-7218</td>
<td>Maven 3.0 or 3.x Builder</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-7453</td>
<td>BuildState is not set before POST plugins</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-7722</td>
<td>Bamboo fails to auto detect mercurial executable on remote agent.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-7717</td>
<td>Support self signed SSL certificates when accessing Git repositories</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-7038</td>
<td>Maven 2 build processor fails to find parent projects</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-7771</td>
<td>Build plan failed to run - it was marked as queued but was not present in the queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### The Bamboo 3 Team

#### Development

**Core Team**
- Brydie McCoy
- James Dumay
- Jason Berry
- Marek Went
- Krystian Brazulewicz
- Przemek Bruski
- Marcin Gardias
- Michael Truong

**Git**
- Slawek Ginter
- Piotr Stefan Stefaniak

#### Team Lead
- Mark Chaimungkalanont

#### Project Manager
- Anton Mazkovoi

*Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.*
Bamboo 3.0.1 Release Notes

25 February 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.0.1.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.0.1 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.0 yet?

Take a look at all the new features in the Bamboo 3.0 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.0 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.0.1 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues</th>
<th>4 issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Key</td>
</tr>
</tbody>
</table>
for getting build results with Maven builder or build is red when it shouldn't

<table>
<thead>
<tr>
<th>Issue Number</th>
<th>Description</th>
<th>Assigned to</th>
<th>Resolution</th>
<th>Status</th>
<th>Date Resolved</th>
<th>Date Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-620</td>
<td>Builds seem to get picked off the queue in random order</td>
<td>Przemek Bruski [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jul 07, 2010</td>
<td>Mar 04, 2011</td>
</tr>
</tbody>
</table>

**Bamboo 3.0.1 Upgrade Guide**

Upgrading from Bamboo 3.0 to 3.0.1

Please follow the [Bamboo generic upgrade guide](#).

- No additional upgrade tasks are required to upgrade from Bamboo 3.0 to 3.0.1.

Upgrading from Bamboo 3.0 or earlier

In addition to the above, please read the [Bamboo 3.0 Upgrade Guide](#) and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the [Bamboo upgrade guides](#) section.

**Bamboo 3.0.2 Release Notes**

**24 March 2011**

The Atlassian Bamboo team is proud to announce the release of **Bamboo 3.0.2**.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.0.2 is of course free to all customers with active Bamboo software maintenance.

**Don't have Bamboo 3.0 yet?**

Take a look at all the new features in the [Bamboo 3.0 Release Notes](#) and see what you are missing out on!
Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.0 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.0.2 are shown below. To view the list in JIRA, please refer to our main JIRA site.

### JIRA Issues (6 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-806</td>
<td>/build resource does not work</td>
<td>Unassigned</td>
<td>Bulkan Evcimen</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 20, 2011</td>
<td>Mar 03, 2011</td>
</tr>
<tr>
<td></td>
<td>BAM-812</td>
<td>Error collecting changes with multiple shallow plans with different branches</td>
<td>Slawek Ginter</td>
<td>Slawek Ginter</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 28, 2011</td>
<td>Mar 04, 2011</td>
</tr>
</tbody>
</table>
1.3.5.753 9 to fix invalid xml character and svn externals errors


Bamboo 3.0.2 Upgrade Guide

Upgrading from Bamboo 3.0 to 3.0.2

Please follow the Bamboo generic upgrade guide.

No additional upgrade tasks are required to upgrade from Bamboo 3.0.1 to 3.0.2.

Upgrading from Bamboo 3.0 or earlier

In addition to the above, please read the Bamboo 3.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo upgrade guides section.

Bamboo 3.0.3 Release Notes

18 April 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.0.3.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release’ section below for details.

Bamboo 3.0.3 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.0 yet?

Take a look at all the new features in the Bamboo 3.0 Release Notes and see what you are missing out on!

Bamboo 3.0.3 Upgrade Guide

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.0.3 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.0.3 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (10 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>BAM-100</td>
</tr>
<tr>
<td>BAM-100</td>
</tr>
<tr>
<td>BAM-821</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>BAM-783</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
Bamboo 3.0.3 Upgrade Guide

Upgrading from Bamboo 3.0.2 to 3.0.3

Please follow the Bamboo generic upgrade guide.

No additional upgrade tasks are required to upgrade from Bamboo 3.0.2 to 3.0.3.

Upgrading from Bamboo 3.0.1 or earlier

In addition to the above, please read the Bamboo 3.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo upgrade guides section.

Bamboo 3.0 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 3.0 from a previous version of Bamboo. For details on the Bamboo 3.0 release, see the Bamboo 3.0 Release Notes.

Please following the Bamboo 3.0-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.
Upgrade Notes

End of Support for Internet Explorer 7

We are ending support for Internet Explorer 7 (IE7) in this release. Please see End of Support Announcements for Bamboo for further details.

Advance Notice of End of Support for Java Platform 5

We are planning on ending support for Java Platform 5 (JDK/JRE 1.5) in Bamboo 3.1. Please see End of Support Announcements for Bamboo for further details.

All Bamboo versions using MS SQL 2005 and 2008 demand Read Committed with Row Versioning isolation level.

- Before starting the upgrade process ensure that your current Bamboo MS SQL database is set to use Read Committed with Row Versioning as its isolation level. If you are planning to restore a Bamboo Backup Zip, ensure that the new database will have this isolation level as well. For instructions on how to set this isolation level, please see Microsoft SQL Server 2005 and 2008.

Specifying Artifact Location

In Bamboo 2.7 and earlier, artifacts are stored under xml-data/builds under your $bambooHome (unless specified otherwise). An upgrade task for Bamboo 3.0 will move your artifacts out of $bamboo.project.directory into a separate artifacts directory under $bambooHome. If your artifacts are currently not located under your $bambooHome, i.e. you manually changed the location of your $bamboo.project.directory, you will need to do one of the following:

- make sure that there is enough room under $bambooHome to accommodate the artifacts in the new artifacts directory, or
- set the bamboo.artifacts.directory property (in bamboo.cfg.xml) to the preferred location for your artifacts. You must update this property before the upgrade. The upgrade task will use the location specified by this property, rather than moving artifacts to the new artifacts directory under $bambooHome.

If your $bamboo.project.directory currently points to a different physical disk to your $bambooHome, the upgrade process will copy (rather than move) data between locations, unless you set the bamboo.artifacts.directory property.

The new Default Path for Artifacts

- Bamboo 2.6 and earlier versions:

  xml-data/builds/PLAN_KEY/download-data

- Bamboo 2.7:
<bamboo-home>/xml-data/builds/JOB_KEY/download-data/artifacts/build-BUILD_NUMBER

- Bamboo 3.0:

  <bamboo-home>/artifacts/PLAN_KEY/shared/build-BUILD_NUMBER/

In Bamboo 3.0, 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 BUILD_NUMBER will always be composed with a minimum of 5 digits, having the number completed with zeros. For instance, for build "42" the number will be "00042".

Upgrade Exceptions

- If you experience the following exception during the Bamboo 3.0 upgrade, it means that the upgrade task has failed to fully migrate a directory, as part of the internal artifact storage migration. You will need to manually move the directory and restart the upgrade.

Unable to move DIRECTORY_NAME_A -> DIRECTORY_NAME_B, destination directory already exists. This might indicate interrupted upgrade process. To continue upgrade, move directory manually.

Crowd Integration Authenticator

Bamboo 3.0 is using the new 2.4 version of the Seraph authenticator. Please, go through the Integrating Crowd with Atlassian Bamboo steps to ensure that the new necessary configurations will be applied.

Upgrading from Bamboo 2.7 to 3.0

Before you begin, do the following:

1. Back up your existing installation of Bamboo

We strongly recommend that you do the following to back up your Bamboo installation:

- Back up your xml-data directory — See the Bamboo generic upgrade guide for instructions.
- Export your Bamboo data for backup — See the Exporting data for backup for instructions. Please note, that this may take a long time to complete depending on the number of builds and tests in your system.

2. Ensure that your plugins work

If you are using plugins, ensure that your plugins are compiled against 3.0 before upgrading.

Before you upgrade, please read the following important points that relate to Bamboo 2.7.

Upgrading from Bamboo prior to 2.7

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 2.6.x

- You will need to upgrade to Bamboo 2.7.4 before upgrading to Bamboo 3.0. If you are not running
Bamboo 2.6.3, we recommend that you upgrade to it 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.

Notes for upgrading from Bamboo 2.5 or earlier

- 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.

Notes for upgrading from a version of Bamboo prior to 2.0

- 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. Please read the Bamboo 2.0 Upgrade Guide for important upgrade instructions for upgrading from earlier versions of Bamboo.

Developing for Bamboo 3.0

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 3.0 guide, which outlines changes in Bamboo 3.0 that may affect Bamboo plugins compiled for Bamboo version 2.7.x or earlier. In particular, please note that the /build REST endpoint has been replaced with /result. Expand parameters have also been changed from builds.build to results.result.

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 3.0 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 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

No content found for label(s) bambooupgradeguide.

Bamboo generic upgrade guide
Before you begin

This upgrade guide contains generic upgrade tasks that are common across Bamboo versions. Please ensure that you also read the upgrade guide for the version you are upgrading to.

Bamboo 4.0.1 has been released. Read the Bamboo 4.0.1 Release Notes and Upgrade Guide.

Don't have Bamboo 4.0? Take a look at the features of Bamboo's latest major version and try it out!

Step 1. Identify your Bamboo directories

Click Administration, and then System Information in your Bamboo instance, and note the location of the 'Bamboo Home', 'Build Path' and 'Configuration Path' directories:

On this page:

- Step 1. Identify your Bamboo directories
- Step 2. Shut down Bamboo
- Step 3. Back up Bamboo
- Step 4. Re-install Bamboo
- Step 5. Update plugins
- Step 6. Re-configure external user repositories
- Step 7. Start Bamboo
- Step 8. Re-index Bamboo (if indicated in release notes)
- Troubleshooting

Related pages:

- Bamboo installation guide
- Bamboo release notes
- Bamboo security advisories

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 |

Step 2. Shut down Bamboo

You need to shut down Bamboo before backing up or performing the upgrade.

Step 3. Back up Bamboo

Backing up Bamboo consists of two tasks:
1. Back up your 'Bamboo Home', 'Build Path' and 'Configuration Path' directories, as identified in Step 1. For more information about these directories please see Important Directories and Files.
   If you are using an external database, then use native database backup tools to acquire a database dump.
   Note that it is not necessary to back up your 'Working Directory'.

2. Export your Bamboo data for backup. See Exporting data for backup for instructions. Please note, that this may take a long time to complete depending on the number of builds and tests in your system.

**Step 4. Re-install Bamboo**

ℹ️ Please note the following:
If you are installing using the `.zip` or `.tgz` packages:

- When specifying the `{BAMBOO_HOME}` directory, 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 will create a new empty database that can be used during the installation Wizard and that the 'Bamboo Home' directory from the old installation will not be used. The new installation will need to point to a new path so that a new 'Bamboo Home' directory can be created. Once the installation was finished, you can then point the new installation to the old 'Bamboo Home', changing the path at file `<Bamboo-install>/webapps/WEB-INF/classes/bamboo-init.properties`

- 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.

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 5. Update 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.

**Step 6. Re-configure external user repositories**

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>/webapps/WEB-INF/classes/atlassian-user.xml` to it's 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.

- **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](#).

**Step 7. Start Bamboo**
Before you start Bamboo

Bamboo needs to have **write access** to your database 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 has completed successfully.

**Step 8. Re-index Bamboo (if indicated in 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/](https://support.atlassian.com/) and attach your `atlassian-bamboo.log` so we can help you find out what's gone wrong.

**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](https://support.atlassian.comlassian/).

**Publication of Bamboo security advisories**

Atlassian's approach to publishing security advisories is detailed in [Security Advisory Publishing Policy](https://support.atlassian.comlassian/).

**Severity levels**

Atlassian's scale for measuring security issues is detailed in [Severity Levels for Security Issues](https://support.atlassian.comlassian/).
Our patch policy

Atlassian's approach to releasing patches is detailed in our Security Patch Policy.

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)

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

- 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.

<table>
<thead>
<tr>
<th>Bamboo Feature</th>
<th>Affected Bamboo Versions</th>
<th>Fixed Version</th>
<th>Issue Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Picker</td>
<td>all earlier than 2.7.4</td>
<td>2.7.4, 3.0</td>
<td>BAM-10024</td>
</tr>
<tr>
<td>Default 'internal server error' page</td>
<td>all earlier than 3.1</td>
<td>3.1</td>
<td>BAM-10026</td>
</tr>
<tr>
<td>viewAgent.action</td>
<td>all earlier than 3.1</td>
<td>3.1</td>
<td>BAM-10027</td>
</tr>
<tr>
<td>configureAgents resource</td>
<td>all earlier than 3.1</td>
<td>3.1</td>
<td>BAM-10028</td>
</tr>
<tr>
<td>chooseBuildsToMove.action</td>
<td>all earlier than 3.1</td>
<td>3.1</td>
<td>BAM-10029</td>
</tr>
</tbody>
</table>

Our thanks to Marian Ventuneac (http://www.ventuneac.net) who reported several of the vulnerabilities mentioned above. We fully support the reporting of vulnerabilities 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 Perforce server as a 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.

<table>
<thead>
<tr>
<th>Bamboo Feature</th>
<th>Affected Bamboo Versions</th>
<th>Fixed Version</th>
<th>Issue Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS command injection</td>
<td>2.4 – 3.1</td>
<td>3.1.1, 3.2</td>
<td>BAM-10030</td>
</tr>
<tr>
<td>vulnerability in Perforce library</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Patch</th>
<th>Patch File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS command injection vulnerability in Perforce library used by Bamboo</td>
<td>Attached to issue BAM-10030</td>
<td>p4java-0.7.5-atlassian-6.jar</td>
</tr>
</tbody>
</table>

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:

1. Download the p4java-0.7.5-atlassian-6.jar file that is attached to the BAM-10030 issue.
2. Stop Bamboo.
3. Make a backup of the <bamboo_install_dir> directory.
4. Copy the downloaded jar file into <bamboo_install_dir>/Bamboo/webapp/WEB-INF/lib, and delete the existing p4java jar file.
5. 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.

<table>
<thead>
<tr>
<th>Bamboo Feature</th>
<th>Affected Bamboo Versions</th>
<th>Fixed Version</th>
<th>Issue Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information leakage</td>
<td>2.0 – 3.2</td>
<td>3.2.3, 3.3</td>
<td>BAM-10031</td>
</tr>
</tbody>
</table>

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](http://www.example.com). 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 [cgiecurity.com](http://cgiecurity.com), [The Web Application Security Consortium](http://www.example.com) and other places on the web.

Vulnerability

The table below describes the Bamboo versions and the specific functionality affected by the XSS vulnerability.

<table>
<thead>
<tr>
<th>Bamboo Feature</th>
<th>Affected Bamboo Versions</th>
<th>Issue Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo User Management</td>
<td>Bamboo 1.0 - 2.7.3</td>
<td>BAM-8260</td>
</tr>
</tbody>
</table>

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](http://www.example.com). You can download the latest version of Bamboo from the [Bamboo download centre](http://www.example.com). 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**
  - **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](https://www.cvedetails.com/vulnerability-severity.html). 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](https://www.cgisecurity.com), [CERT](https://www.cert.org) 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**

<table>
<thead>
<tr>
<th>Server Administration User Interface — Including the User and Group Security, System and Communication sections.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Bamboo User Interface — Including the Create Plan and Build Configuration areas and Log and various Result views.</td>
</tr>
</tbody>
</table>

**Risk Mitigation**

We recommend that you upgrade your Bamboo installation to fix these vulnerabilities. Please see the 'fix' section.
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
    - Severity
    - Risk Assessment
    - Risk Mitigation
    - Vulnerability
    - Fix
  - XSS vulnerabilities when adding Requirements for a Build
    - Severity
    - Risk Assessment
    - Risk Mitigation
    - Vulnerability
    - Fix
  - XSS vulnerabilities in the user's full name
    - Severity
    - Risk Assessment
    - Risk Mitigation
    - Vulnerability
    - Fix
  - XSS vulnerabilities in build logs
    - Severity
    - Risk Assessment
    - Risk Mitigation
    - Vulnerability
    - Fix
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](http://www.cgisecurity.com/), [CERT](http://www.cert.org/) 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](http://www.cgisecurity.com/), [CERT](http://www.cert.org/) 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.

Supported platforms

This page describes the supported platforms for.
### Key:
- ✔️ = Supported; ✗ = Not Supported

<table>
<thead>
<tr>
<th>Java Version</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle JDK (1.5)</td>
<td>✔️ 1.7</td>
</tr>
<tr>
<td></td>
<td>✔️ 1.6</td>
</tr>
<tr>
<td></td>
<td>✗ 1.5</td>
</tr>
<tr>
<td>OpenJDK</td>
<td>✔️ 1.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operating Systems</th>
<th>✔️</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows (2)</td>
<td></td>
</tr>
<tr>
<td>Linux / Solaris (2)</td>
<td>✔️</td>
</tr>
<tr>
<td>Apple Mac OS X (2)</td>
<td>✔️</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application Servers</th>
<th>✔️ 6.0.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Tomcat (3)</td>
<td>✔️ 5.5.x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Databases</th>
<th>✔️ 5.x with JDBC Connector/J 5.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>✗ 5.0.x</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>✔️ 8.2+ with PostgreSQL Driver 8.4.x</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>✔️ 2008 with JTDS 1.2.2</td>
</tr>
<tr>
<td></td>
<td>✔️ 2005 with JTDS 1.2.2</td>
</tr>
<tr>
<td>Oracle</td>
<td>✔️ 11G with Oracle 11.2.x</td>
</tr>
<tr>
<td></td>
<td>✗ 10G</td>
</tr>
<tr>
<td>HSQLDB (4)</td>
<td>✔️ (for evaluation only)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Browsers</th>
<th>✔️ 9.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Internet Explorer</td>
<td>✔️ 8.0</td>
</tr>
<tr>
<td></td>
<td>✗ 7.0</td>
</tr>
<tr>
<td>Mozilla Firefox</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Latest stable version supported</td>
</tr>
<tr>
<td>Safari</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Latest stable version supported</td>
</tr>
<tr>
<td>Chrome</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Latest stable version supported</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Repositories</th>
<th></th>
</tr>
</thead>
</table>

*Documentation for Bamboo 4.0*

*Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.*
<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 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 | 1.7 command line client interoperation |
| Git | ✔️ with client 1.7 | |
| Perforce | ✔️ | |
| CVS | ✔️ | |

**Notes:**

1. **Oracle JDK:**
   - 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 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.

2. Please note:
   - 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:
     ```
     $ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/bamboo --shell /bin/bash bamboo
     ```

3. 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.

4. 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](#).

5. Note that your agents can build software with any JDK version. You only need to run the agent and server using a supported JDK.

*Created by Atlassian in 2012. Licensed under a [Creative Commons Attribution 2.5 Australia License](#).*
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.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Bamboo End of Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL 5.0</td>
<td>December 2011</td>
</tr>
<tr>
<td>Oracle 10g</td>
<td>December 2011</td>
</tr>
</tbody>
</table>

**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-databases-bamboo)
- [Deprecated Java Platforms for Bamboo (16 February 2011)](deprecated-java-platforms)
- [Deprecated Web Browsers for Bamboo (16 February 2011)](deprecated-web-browsers)

### 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](https://confluence.atlassian.com/display/BAM/Supported+Platforms) for details of platform support for Bamboo. If you have questions or concerns regarding this announcement, please email [eol-announcement at atlassian dot com](mailto:eol-announcement@atlassian.com).

**End of Life Announcement for Database Support**

<table>
<thead>
<tr>
<th>Database</th>
<th>Support End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL 5.0</td>
<td>When Bamboo 3.4 releases, after December 2011</td>
</tr>
<tr>
<td>Oracle 10g</td>
<td>When Bamboo 3.4 releases, after December 2011</td>
</tr>
</tbody>
</table>

- 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](https://www.atlassian.com/software/jira) for more details regarding platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eol-announcement at atlassian dot com.

**End of Life Announcement for Java Platform Support**

<table>
<thead>
<tr>
<th>Java Platform</th>
<th>Support End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Platform 5 (JDK/JRE 1.5)</td>
<td>When Bamboo 3.1 releases, due in the first half of 2011</td>
</tr>
</tbody>
</table>

- **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 at atlassian dot 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](https://www.atlassian.com/software/jira) for details of platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eol-announcement at atlassian dot com.

**End of Life Announcement for Web Browser Support**
<table>
<thead>
<tr>
<th>Web Browser</th>
<th>Support End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Internet Explorer (version 7 only)</td>
<td>When Bamboo 3.0 releases, due February 2011</td>
</tr>
</tbody>
</table>

- **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 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 For Release Management
  - Using Bamboo with Clover
  - 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

---

**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?
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).

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 folder>/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 folder>/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
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.)

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.
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/
```

### Getting Bamboo to use the jetty.xml file

The instructions on this page apply to the Bamboo distribution, not the Bamboo EAR-WAR distribution.

By default Bamboo doesn’t use the `jetty.xml` file to configure itself. If you need to modify the `jetty.xml` for advanced configuration (such as JNDI or https), you will also need to tell Bamboo to use it.

The method for doing this depends on whether you are using the `bamboo.sh` startup script or the Java Service Wrapper.

**Step 1 - Instructing Bamboo to use jetty.xml**
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):

For Bamboo 3.0 and newer:

```bash
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 /
"
```

For Bamboo older than 3.0:

```bash
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:

For Bamboo 3.0 and newer:

```bash
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"
```

For Bamboo older than 3.0:

```bash
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"
```

If you are using the Java Service Wrapper to start Bamboo:

When starting up Bamboo with the Java Service Wrapper, you'll need to modify the `wrapper.conf` file in the `conf` directory.

- You will need to replace the argument which specifies your port number "wrapper.app.parameter.2=8085" with "wrapper.app.parameter.2=../webapp/WEB-INF/classes/jetty.xml".
- You will need to 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).

**Step 2 - Setting root context web application in jetty.xml**

If you are using Bamboo 1.2.4 follow the instructions below:

Edit the webapp root context in your `jetty.xml` file situated in `<Bamboo-install>/webapp/WEB-INF/classes/jetty.xml`. From
To

```xml
<Call name="addWebApplication">
  <Arg>/bamboo</Arg>
  <Arg>
    <SystemProperty
        name="bamboo.webapp"
        default="bamboo-web-app/src/main/webapp"/>
  </Arg>
</Call>
</Configure>
```

If you are using Bamboo 2.0 follow the instructions below:

**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 BAM-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:


```bash
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:


```bash
FROM
# Setup JBoss specific properties
JAVA_OPTS="-Dprogram.name=$PROGNAME $JAVA_OPTS"

TO
```

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
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 Windows as the local user

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 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 a 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 `.keystore` file 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.

2. Configuring Jetty

Using the Sun JVM, add the SunJssseListener as a HttpListeners, in the `../<Bamboo_Application_Directory>/webapp/WEB-INF/classes/jetty.xml` file add the following lines. This will make Bamboo accessible in port 8443 on `https://localhost:8443/`
If you are using Bamboo 1.2.4 (or earlier)

```xml
<Call name="addListener">
  <Arg>
    <New
class="org.mortbay.http.SunJsseListener">
      <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>
```

If you are using Bamboo 2.0 (or newer version)
<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>

The keystore file in this example is given relative to the Bamboo Application Directory.

Please ensure that jcert.jar, jnet.jar and jsse.jar are on your classpath.

3. Getting Bamboo to use the jetty.xml file

Follow this Knowledge Base article, to instruct Bamboo to use the jetty.xml file configured in step 2.

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 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>/default/deploy/mail-service.xml to be

```xml
<?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 -->
    </configuration>
  </configuration>
</mbean>
<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 name="mail.smtp.socketFactory.class" value="javax.net.ssl.SSLSocketFactory"/>
<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>
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):

```xml
<Call name="addWebApplication">
  <Arg>/</Arg>
  <Arg>
    <SystemProperty
      name="bamboo.webapp" default="webapp"/>
  </Arg>
</Call>
```

If you are using Bamboo 2.0:
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:

```xml
<!--
<Call name="addService">
  <Arg>
    ...
  </Arg>
</Call>
-->
```

In Bamboo 2.0:
If you are experiencing class loading problems with your mail session. Try uncommenting the following line in the web applications context (2.0 only):

```xml
<!--
<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](http://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 (i.e. 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 Vulnerability

If you find a security bug in the product, please open an issue on http://jira.atlassian.com in the relevant project.

- Set the priority of the bug to 'Blocker'.
- Provide as much information on reproducing the bug as possible.
- Set the security level of the bug to 'Developer and Reporters only'.

All communication about the vulnerability 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 discover a security vulnerability, please attempt to create a test case that proves this vulnerability locally before opening either a bug or a support issue. When creating an issue, please include information on how the vulnerability can be reproduced; see our Bug Fixing Policy for general bug reporting guidelines. We will prioritise fixing the reported vulnerability if your report has information on how the vulnerability can be exploited.

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 systems, [http://jira.atlassian.com](http://jira.atlassian.com) and [http://studio.atlassian.com](http://studio.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](http://jira.atlassian.com) 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](http://jira.atlassian.com) we aim to implement highly requested features, but it is not the only determining factor. Other factors include:

- **Direct feedback** from face to face meetings with customers, and through our support and sales channels.
- **Availability of staff** to implement features.
- **Impact** of the proposed changes on the application and its underlying architecture.
- **How well defined** the requested feature is (some issues gain in popularity rapidly, allowing little time to plan their implementation).
- **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 systems, [http://jira.atlassian.com](http://jira.atlassian.com) and [http://studio.atlassian.com](http://studio.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](http://jira.atlassian.com).

If you require significant customisations, you may wish to get in touch with our [partners](http://jira.atlassian.com). They specialise in extending Atlassian products and can do this work for you. If you are interested, please [contact us](http://jira.atlassian.com).

Further reading

See [Atlassian Support Offerings](http://jira.atlassian.com) 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 (e.g. JIRA 4.2.4) 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).

- 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 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. This applies to all security advisories, including severity levels of critical, high, medium and low.
- We will send a copy of all 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 'Email Prefs' 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.

Early warning of critical security vulnerabilities:

- If the vulnerability is rated critical (see our criteria for setting severity levels) we may send an early warning to the 'Technical Alerts' mailing list approximately one week before releasing the fix. This early warning is in addition to the security advisory itself, described above.

- However, if the vulnerability is publicly known or being exploited, we will release the security advisory and patches as soon as possible, potentially without early warning.

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 JIRA Studio, since JIRA Studio 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:

<table>
<thead>
<tr>
<th>CVSS score range</th>
<th>Severity in advisory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 2.9</td>
<td>Low</td>
</tr>
<tr>
<td>3 – 5.9</td>
<td>Medium</td>
</tr>
<tr>
<td>6.0 – 7.9</td>
<td>High</td>
</tr>
<tr>
<td>8.0 – 10.0</td>
<td>Critical</td>
</tr>
</tbody>
</table>

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 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 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 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](https://confluence.atlassian.com/display/DOC/Support) 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 For Release Management
- Using Bamboo with Clover
- 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 2.1 Database schema

### Binding Bamboo to one IP address

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:*
<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>
    </New>
  </Arg>
</Call>

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>
    </New>
  </Arg>
</Call>

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>
    </New>
    <!-- Set name="Host">127.0.0.1</Set-->-->
</Call>

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> -->
    </New>
</Call>

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,

```xml
<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,
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](http://my.atlassian.com) to find the SEN for a specific license

---

**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,
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**

They 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 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 *600 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.

### 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 |
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 tools
    adjust 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 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 09:13:39,891 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
In Bamboo 2.0 we introduced compatibility with Oracle and MS SQL Server, this meant we the up 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 itself 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.

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
<?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
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.invo
ke (Unknown Source)
   at
   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
at
java.lang.reflect.Method.invoke(Method.java:585)
at
org.apache.maven.surefire.SurefireBooteer.runTestsInProcess(SurefireBooteer.java:313)
at
org.apache.maven.surefire.SurefireBooteer.run(SurefireBooteer.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

Created by Atlassian in 2012. Licensed under a Creative Commons Attribution 2.5 Australia License.
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
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"/>
  </properties>
</testsuite>
<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"/>
<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"/>
<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 XUInt 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
```
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
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Command Options</th>
<th>Behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINARY (Default)</td>
<td>-b</td>
<td>forces all files to be checked out in binary and won’t convert any line endings</td>
</tr>
<tr>
<td>TEXT</td>
<td>-kv</td>
<td>forces all files to be checked out as text and converts all line endings (even Binary files)</td>
</tr>
<tr>
<td>NONE</td>
<td></td>
<td>lets CVS decide whether or not to convert line endings</td>
</tr>
</tbody>
</table>

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:**


**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 -

   ```xml
   ``

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
Proccess ID and execute the command:

```
jstack <ProccessID>
```

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
```

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','x61Ey612K12gpFL56FT9weDnpSo4AV8j8+qx2AUThdRy036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3pB37A==','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:

```
x61Ey612K12gpFL56FT9weDnpSo4AV8j8+qx2AUThdRy036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3pB37A==
```
Paste the `admin` password hash between the " characters of their existing PASSWORD_HASH. The new administrator login entry should look like:

```sql
INSERT INTO USERS
VALUES(1,'USERNAME','x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3pB37A==','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:

```sql
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`

```text
x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3pB37A==
```

To change the password to `admin` for a given username:

1. Shutdown Bamboo
2. Connect to your database. run this SQL on your database:

   ```sql
   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+XURPWx1ONxp3Y3pB37A==' where NAME = 'USER_NAME_FROM_STAGE_ONE'

3. Start Bamboo
4. Login with your username and your password is now admin

**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

```bash
#!/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

```ruby
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 For Release Management

Suggested reading for users wishing to configure Bamboo for release management:

- [Using Bamboo for JIRA release management](#)
- [Plugins for release management and deployment](#)
- [Forum thread on release management](#)

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 plugin](#) for further details.
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 Bamboo with your build:
   - Clover-for-Ant Installation Guide
   - Clover-for-Maven 2 and 3 Installation Guide
2. And either:
   - call the Clover goal in your plan configuration (see Configuring tasks);
   - or:
   - add the maven-clover-plugin report to the reports section in your POM.
3. Ensure that there are tests present in your build plan that generate test results in JUnit test report format.
4. Ensure that your build creates a Clover report (that is, a clover.xml file). Bamboo will use this Clover report as source.
5. Set up Bamboo to read the Clover report (clover.xml file) generated by Clover. To do this:
   a. Ensure the ‘Clover output will be produced’ check-box is ticked in your plan's build configuration page.
   b. Instruct Bamboo on the location of your ‘Clover XML Directory’ — where Bamboo will look for the XML report output file from Clover. Please specify the path to your clover.xml file relative to your plan's root directory (e.g. your plan's root directory is /home/bamboouser.bamboo-home/xml-data/build-dir/MY_PLAN/ and you would enter target/clover/site/clover.xml). Please do not specify the absolute path.

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.

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:jar for Bamboo to build successfully. The relevant POMs for this look something like this:
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.
2. Install it on your local machine by entering the following command in a terminal:

```shell
mvn install:install-file -DgroupId=javax.mail -DartifactId=mail -Dversion=1.3.3 -Dpackaging=jar -Dfile=YOUR/PATH/TO/FILE
```

**To install the **javax.transaction:jta** JAR into your local Maven2 repository:**

1. Download the **javax.transaction:jta:jta** Jar from Sun's website.
2. Install it on your local machine by entering the following command in a terminal:

```shell
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**

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
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.

<table>
<thead>
<tr>
<th>System Property</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>atlassian.bamboo.builder. successMarker</td>
<td>Specifies the text (or string) that Bamboo looks for in the build log to determine if the build was successful</td>
<td>BUILD SUCCESS</td>
</tr>
<tr>
<td>atlassian.bamboo.builder. failedMarker</td>
<td>Specifies the text (or string) that Bamboo looks for in the build log to determine if the build failed</td>
<td>BUILD FAILED</td>
</tr>
<tr>
<td>SUCCESS_MESSAGE_LINES</td>
<td>Specifies the number of lines from the end of the builder log in which to check for the values of atlassian.bamboo.builder.successMarker or atlassian.bamboo.builder.failedMarker.</td>
<td>250</td>
</tr>
</tbody>
</table>

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:

```xml
...  
<groupId>com.atlassian.boo</groupId>
<artifactId>boo-test</artifactId>
<packaging>jar</packaging>
<version>1.1.{$env.bambooBuildNumber}-SNAP SHOT</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

```xml
...<echo message="bamboo.buildKey = ${build.key}"/>
...```

Bamboo resources

Resources for Evaluators

- Free Trial
- Feature Tour

Resources for Administrators

- 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 Plugin Exchange

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

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.

On this page:
- Blogging your Technical Tips and Guides – Tips of the Trade
- Contributing Documentation in Other Languages
- Updating the Documentation Itself
  - Getting Permission to Update the Documentation
  - Our Style Guide
  - How we Manage Community Updates

Blogging your Technical Tips and Guides – Tips of the Trade

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...

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.

RELATED TOPICS

Tips of the Trade
Author Guidelines
Atlassian Contributor License Agreement

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:
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](#)

**Atlassian Bamboo and Perl Test Harness**
- By: Gary Richardson, on blog 'The Cult of Gary'
- About: Getting your Perl test cases to work with Bamboo, using TAP::Harness::JUnit
- Date: 7 November 2008
- Related documentation: [Can Bamboo build and test non-Java projects](#)

**Setting up phpUnit on Elastic Bamboo**
- By: Michael White
- About: Getting automated unit testing working for your PHP code. (Note, this tutorial is written for JIRA Studio, however the instructions can be applied to non-JIRA Studio installations of Bamboo).
- Date: 20 Oct 2010
- 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

Evaluator resources
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

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

Glossary

activity log
agent
agent-specific capability
artifact
author
build
build activity
build duration
build log
build queue
build result
build strategy
build telemetry
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 provides capabilities to 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 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.

For more information, see:

- Configuring agents
- Agents and capabilities
- Configuring a job’s requirements

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.
author
An **author** is any person who contributes to a **build** by checking-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 [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 strategy**

The build strategy for a [Plan](#) determines how a build is triggered.

This table lists Bamboo's available 'build strategies', which determine how the execution of a plan (i.e. a build) is triggered. You set a Build Strategy option on the Plan Details tab of the plan configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
</tr>
</thead>
</table>
| Code is updated                         | Polling the Repository for changes                                           | • This is a 'pull strategy'.  
• This is the simplest option.  
• But it does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository. |
|                                        |                                                                              | See [Polling the repository for changes](#).                                          |
| Repository triggers the build when changes are committed | Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan. | • This is a 'push strategy'.  
• This option minimises server load as message events are sent only when code changes to this repository are committed.  
• But you must [configure your source code management system](#) to send message events to Bamboo about code changes in this repository. |
<p>|                                        |                                                                              | See <a href="#">Repository triggers the build when changes are committed</a>.                    |</p>
<table>
<thead>
<tr>
<th>Scheduled</th>
<th>Cron Based Scheduling</th>
<th>Manual</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bamboo will trigger a build of this plan based on a cron expression.</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>See Cron-based scheduling.</td>
<td>See Triggering a plan build manually.</td>
<td>A capability is a feature of an agent. A capability can be:</td>
</tr>
<tr>
<td></td>
<td>• Allows you to run builds based on a schedule.</td>
<td>• This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests.</td>
<td>• an executable (e.g. Maven)</td>
</tr>
<tr>
<td></td>
<td>• This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours.</td>
<td>• This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>• a JDK</td>
</tr>
<tr>
<td></td>
<td>• Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td></td>
<td>• 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').</td>
</tr>
</tbody>
</table>

**Scheduled**

**Cron Based Scheduling**

Bamboo will trigger a build of this plan based on a cron expression. See Cron-based scheduling.

- Allows you to run builds based on a schedule.
- 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 repository.

**Single daily build**

Bamboo will trigger a build of this plan once per day at a specified time. See Single daily build.

- 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 repository.

**Manual**

Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency. See Triggering a plan build manually.

- This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests.
- This frees up Bamboo agents to build other plans which are less likely to fail.
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. 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 Configuring a new custom capability.
- To specify a job's requirement for a custom capability, see Configuring a job's requirements.

**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 is a feature in Bamboo that 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.

**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 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 a program external to Bamboo used to automate processes. 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 Configuring a new executable.

**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 favourite plans.
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 defines:

- what gets built (i.e. the source code repository) — this can be a custom, job-specific source repository or the 'default source repository' of the plan that contains this job;
- which agent capabilities are required for the build (based on job-specific requirements and requirements of the job's tasks);
- what tasks make up the job and the order in which they are executed;
- what artifacts the job's build will produce;
- 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.

Plans:

- are organised into one or more stages;
- have one or more jobs in each stage;
- contain a single 'Default job' in a single stage, straight after creating a new plan;
- define default settings for what gets built by jobs in the plan (i.e. the 'default source repository');
- define how the plan's build is triggered;
- define who will be notified of the job's build result;
- define who has permission to view and perform various actions on the plan and its jobs.

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 Granting plan permissions in bulk.

See also global permission.

project

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.

queue

See build queue.

reason

A build's reason is the way in which the build was triggered.

There are several methods by which Bamboo can 'trigger' (i.e. begin executing) a plan's build:

Build Strategy

- Code updated — ensures that a plan build only occurs when something changes in the plan's source repository (which may affect the outcome of a plan build) i.e. whenever one or more authors check in code.
- Scheduled build — can allow a team to structure the day according to a predictable schedule. Scheduled builds are run regardless of whether or not any code changes have occurred i.e. at scheduled times or specified time intervals.
- Manual build — allows you to ensure that builds are only triggered manually (or by the successful builds
of other plans on which this plan is dependent).

- **Initial clean build** — i.e. immediately after a new plan has been created.

**Build Dependency**

- **Dependency** — a build can be triggered whenever another plan's build is successfully completed.

The trigger method for a plan's build is listed in the 'Reason' column on the dashboard.

Note that these trigger methods can only be configured by a Bamboo administrator. For more information please see Triggering builds.

---

**remote agent**

See agent.

**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. Stages have the following characteristics:

- A stage may contain one or more jobs;
- Depending on the availability of Bamboo agents, all jobs in a stage can be processed in parallel;
- Stages are processed consecutively within a plan, one at a time;
- All jobs in a stage must be built and succeed before Bamboo builds any jobs in the next stage. If any job fails in a stage, no further stages in the plan will be processed and the plan's build will fail.

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:

<table>
<thead>
<tr>
<th>Default packages/capabilities</th>
<th>Path/value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Builders</strong></td>
<td></td>
</tr>
<tr>
<td>Ant (version 1.8.2)</td>
<td>/opt/ant-1.8.2</td>
</tr>
<tr>
<td>Bash (Command)</td>
<td>/bin/bash</td>
</tr>
<tr>
<td>Grails 1.2.2</td>
<td>/opt/grails-1.2.5</td>
</tr>
<tr>
<td>Grails 1.3.1</td>
<td>/opt/grails-1.3.7</td>
</tr>
<tr>
<td>Grails 2.0.1</td>
<td>/opt/grails-2.0.1</td>
</tr>
<tr>
<td>Maven 1 (version 1.0.2)</td>
<td>/opt/maven-1.0.2</td>
</tr>
<tr>
<td>Maven 1.1</td>
<td>/opt/maven-1.1</td>
</tr>
<tr>
<td>Maven 2.0 (Maven 2.x) (version 2.0.11)</td>
<td>/opt/maven-2.0.11</td>
</tr>
<tr>
<td>Maven 2.1 (Maven 2.x) (version 2.1.0)</td>
<td>/opt/maven-2.1.0</td>
</tr>
<tr>
<td>Maven 2.2 (Maven 2.x) (version 2.2.1)</td>
<td>/opt/maven-2.2.1</td>
</tr>
<tr>
<td>Maven 3.0 (Maven 3.x) (version 3.0.4)</td>
<td>/opt/maven-3.0.4</td>
</tr>
<tr>
<td><strong>JDKs</strong></td>
<td></td>
</tr>
<tr>
<td>JDK 1.5 (version 5u22)</td>
<td>/opt/jdk-5 (for Java Home)</td>
</tr>
<tr>
<td>JDK 1.6 (version 6u31)</td>
<td>/opt/jdk-6 (for Java Home)</td>
</tr>
<tr>
<td>Oracle JDK 1.7 (version 7u3)</td>
<td>/opt/jdk-7 (for Java Home)</td>
</tr>
<tr>
<td>OpenJDK 1.7 (version 7u2)</td>
<td>/opt/openjdk-7 (for Java Home)</td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
</tr>
<tr>
<td>Mercurial 1.4 Executable (version 1.8.4)</td>
<td>/usr/bin/hg</td>
</tr>
</tbody>
</table>

* 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.
and has the following default packages/capabilities:

<table>
<thead>
<tr>
<th>Default packages/capabilities</th>
<th>Path/value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Builders</strong></td>
<td></td>
</tr>
<tr>
<td>Ant (version 1.8.2)</td>
<td>C:\opt\ant-1.8.2</td>
</tr>
<tr>
<td>Maven 2.0 (Maven 2.x) (version 2.0.11)</td>
<td>C:\opt\maven-2.0.11</td>
</tr>
<tr>
<td>Maven 2.1 (Maven 2.x) (version 2.1.0)</td>
<td>C:\opt\maven-2.1.0</td>
</tr>
<tr>
<td>Maven 2.2 (Maven 2.x) (version 2.2.1)</td>
<td>C:\opt\maven-2.2.1</td>
</tr>
<tr>
<td>Maven 3.0 (Maven 3.x) (version 3.0.3)</td>
<td>C:\opt\maven-3.0.3</td>
</tr>
<tr>
<td>NAnt 0.91</td>
<td>C:\opt\nant-0.91</td>
</tr>
<tr>
<td><strong>JDKs</strong></td>
<td></td>
</tr>
<tr>
<td>JDK 1.6 (version 6u31)</td>
<td>C:\Program Files\Java\jdk1.6.0_31</td>
</tr>
<tr>
<td><strong>Browsers</strong></td>
<td></td>
</tr>
<tr>
<td>Firefox (FF10)</td>
<td>C:\Program Files (x86)\Mozilla Firefox</td>
</tr>
<tr>
<td>Internet Explorer (IE9)</td>
<td>C:\Program Files (x86)\Internet Explorer</td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
</tr>
<tr>
<td>Git</td>
<td>C:\Program Files (x86)\Git\bin\git.exe</td>
</tr>
<tr>
<td>Mercurial 2.0 Executable (version 2.0)</td>
<td>C:\Program Files\TortiseHg\hg.exe</td>
</tr>
<tr>
<td>ssh (cygwin) - for hg and git</td>
<td>C:\cygwin\bin\ssh.exe</td>
</tr>
</tbody>
</table>

* 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 an operation that 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.

See Configuring tasks.

triggering
There are several methods by which Bamboo can 'trigger' (i.e. begin executing) a plan's build:

Build Strategy

- **Code updated** — ensures that a plan build only occurs when something changes in the plan's source repository (which may affect the outcome of a plan build) i.e. whenever one or more authors check in code.
- **Scheduled build** — can allow a team to structure the day according to a predictable schedule. Scheduled builds are run regardless of whether or not any code changes have occurred i.e. at scheduled times or specified time intervals.
- **Manual build** — allows you to ensure that builds are only triggered manually (or by the successful builds of other plans on which this plan is dependent).
- **Initial clean build** — i.e. immediately after a new plan has been created.

Build Dependency

- **Dependency** — a build can be triggered whenever another plan's build is successfully completed.

The trigger method for a plan's build is listed in the 'Reason' column on the dashboard.

Note that these trigger methods can only be configured by a Bamboo administrator. For more information please 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.