

Documentation for Bitbucket Server 4.6

Contents

Bitbucket Server documentation home	5
Getting started	5
Supported platforms	7
Supported platforms details	10
Using Bitbucket Server in the enterprise	12
Installing and upgrading Git	15
Configuring JIRA integration in the Setup Wizard	18
Getting started with Git and Bitbucket Server	25
Importing code from an existing project	28
Bitbucket Server tutorials	30
Tutorial: Learn Git with Bitbucket Server	30
Create a Git repository	31
Copy your Git repository and add files	31
Pull changes from your Git repository on Bitbucket Server	31
Use a Git branch to merge a file	31
Tutorial: Work with Bitbucket Server	31
Set up SourceTree to work with Bitbucket Server	32
Create a personal repository in Bitbucket Server	32
Clone your repository and manage files locally	33
Commit and push changes to Bitbucket Server	36
Tutorial: Learn SourceTree with Bitbucket Server	38
Create a new repository	38
Copy your repository and add files	38
Pull changes from your repository with SourceTree	38
Use SourceTree branches to merge an update	38
Tutorial: Learn about pull requests in Bitbucket Server	38
Fork a repository	38
Copy your fork and make a change to the repository	38
Create a pull request to merge your changes	38
Using Bitbucket Server	38
Creating projects	39
Creating repositories	40
Creating personal repositories	41
Using repository hooks	43
Permanently authenticating with Git repositories	44
Clone a repository	46
Controlling access to code	48
Using branch permissions	50
Branch permission patterns	51
Using repository permissions	52
Using project permissions	53
Allowing public access to code	54
Using SSH keys to secure Git operations	55
Creating SSH keys	56
SSH user keys for personal use	58
SSH access keys for system use	60
Workflow strategies in Bitbucket Server	63
Using branches in Bitbucket Server	64
Automatic branch merging	68
Using forks in Bitbucket Server	70
Keeping forks synchronized	71
Using pull requests in Bitbucket Server	74
Create a pull request	75
Review and discuss a pull request	75
Merge a pull request	78
Search for pull requests	78

Checks for merging pull requests	79
Search for code in Bitbucket Server	80
Notifications	82
Markdown syntax guide	83
Requesting add-ons	88
Integrating Bitbucket Server with Atlassian applications	89
JIRA integration	90
Using Smart Commits in Bitbucket Server	94
HipChat notifications	97
Bamboo integration	98
Set the default time zone in Bitbucket Server	101
Use repository tags	102
Administering Bitbucket Server	103
Users and groups	103
External user directories	106
Connecting Bitbucket Server to an existing LDAP directory	108
Connecting Bitbucket Server to JIRA for user management	115
Delegating Bitbucket Server authentication to an LDAP directory	118
Connecting Bitbucket Server to Crowd	122
Global permissions	125
Setting up your mail server	126
Linking Bitbucket Server with JIRA	128
Using custom JIRA issue keys with Bitbucket Server	133
Connecting Bitbucket Server to an external database	133
Connecting Bitbucket Server to MySQL	135
Connecting Bitbucket Server to Oracle	138
Connecting Bitbucket Server to PostgreSQL	141
Connecting Bitbucket Server to SQL Server	143
Transitioning from jTDS to Microsoft's JDBC driver	148
Migrating Bitbucket Server to another server	149
Specifying the base URL for Bitbucket Server	151
Configuring the application navigator	151
Managing add-ons	152
POST service webhook for Bitbucket Server	153
Audit logging in Bitbucket Server	157
Audit events in Bitbucket Server	157
Advanced actions	162
Running the Bitbucket Server installer	163
Automated setup for Bitbucket Server	165
Starting and stopping Bitbucket Server	167
Install Bitbucket Server from an archive file	169
Install and configure a remote Elasticsearch instance	176
Running Bitbucket Server as a Linux service	179
Running Bitbucket Server as a Windows service	185
Bitbucket Server config properties	188
Proxying and securing Bitbucket Server	216
Securing Bitbucket Server with Tomcat using SSL	219
Integrating Bitbucket Server with Apache HTTP Server	226
Securing Bitbucket Server with Apache using SSL	231
Securing Bitbucket Server behind nginx using SSL	234
Securing Bitbucket Server behind HAProxy using SSL	237
Enabling SSH access to Git repositories in Bitbucket Server	240
Setting up SSH port forwarding	243
Using diff transcoding in Bitbucket Server	246
Changing the port that Bitbucket Server listens on	247
Moving Bitbucket Server to a different context path	248
Running Bitbucket Server with a dedicated user	249
Bitbucket Server debug logging	249
Data recovery and backups	252
Using the Bitbucket Server Backup Client	254
Using Bitbucket Server DIY Backup	260
Lockout recovery process	271

Scaling Bitbucket Server	272
Scaling Bitbucket Server for Continuous Integration performance	276
Bitbucket Server production server data	278
High availability for Bitbucket Server	282
Clustering with Bitbucket Data Center	286
Installing Bitbucket Data Center	287
Adding cluster nodes to Bitbucket Data Center	298
Enabling JMX counters for performance monitoring	301
Getting started with Bitbucket Server and AWS	307
Quick Start with Bitbucket Server and AWS	308
Launching Bitbucket Server in AWS manually	309
Administering Bitbucket Server in AWS	313
Recommendations for running Bitbucket Server in AWS	318
Securing Bitbucket Server in AWS	326
Using Bitbucket Server DIY Backup in AWS	328
Disabling HTTP(S) access to Git repositories in Bitbucket Server	334
Smart Mirroring	334
Set up a mirror	336
Git Large File Storage	340
Copy of Bitbucket Server config properties	344
Updating your Bitbucket Server license details	371
Git resources	372
Basic Git commands	373
Bitbucket Server FAQ	375
Bitbucket rebrand FAQ	379
How do I change the external database password	381
Bitbucket Server home directory	382
Raising a request with Atlassian Support	384
Support policies	386
Bug fixing policy	386
New features policy	387
Security Bugfix Policy	388
Finding Your Bitbucket Server Support Entitlement Number (SEN)	388
Building Bitbucket Server from source	389
Contributing to the Bitbucket Server documentation	389
Collecting analytics for Bitbucket Server	390
Bitbucket Server EAP - How to update your add-on	390
Releases	399

Bitbucket Server documentation home

Bitbucket Server is self-hosted Git repository collaboration and management for professional teams.

Find out more and try it for free [here](#).



Get started

Get started installing Bitbucket Server

Using Bitbucket Server in the enterprise

Importing code from an existing project

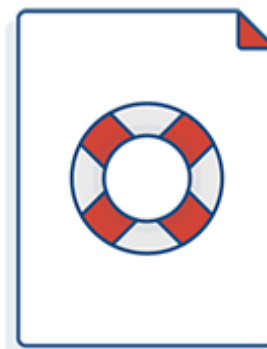


Admin

Bitbucket Server upgrade guide

Bitbucket Server 4.6 release notes

How to update your add-on



Knowledge base

Troubleshooting Git

Troubleshooting Installation

Troubleshooting JIRA Integration

Customer stories

Ecommerce Speed

NASA Migrates to Git

Leading Travel Company Migrates to Git



Users and groups

Connect Bitbucket Server to LDAP

Connect Bitbucket Server to JIRA for user management

Connect Bitbucket Server to Crowd



Integrations

Integrate Bitbucket Server and JIRA

Integrate Bitbucket Server and Bamboo

Integrate Bitbucket Server and HipChat

Getting started

Atlassian Bitbucket Server is the on-premises Git repository management solution for enterprise teams. It allows everyone in your organization to easily collaborate on your Git repositories.

1. Install Git and Perl

Bitbucket Server requires Git on the machine that will run Bitbucket Server. If you need to check, install, or upgrade Git on the Bitbucket Server instance machine, see [Installing and upgrading Git](#).

Check that you have all the other [system requirements](#), including Perl, to avoid any trouble.

2. Install Bitbucket Server

Installers are available for the Linux, Mac OS X and Windows operating systems.

See [Running the Bitbucket Server installer](#) for details.

You may be interested in these alternative provisioning approaches:

- [Installing Bitbucket Server from an archive file](#)
- [Docker repository](#)

3. Set up Bitbucket Server

The Bitbucket Server Setup Wizard runs automatically when you visit Bitbucket Server in your browser the first time Bitbucket Server is started. The Setup Wizard guides you to:

- Specify the default language for Bitbucket Server.
- [Connect Bitbucket Server to an external database](#) (the internal HSQL database is great for evaluating Bitbucket Server, but is not recommended for production installations). You'll need to have created the external database before running the Setup Wizard. For MySQL, you'll need to have [installed the JDBC driver](#) too.

- Enter your Bitbucket Server license key.
- [Set the base URL for Bitbucket Server](#).
- Set up an administrator account.
- [Integrate Bitbucket Server with JIRA](#).
- Log in to Bitbucket Server.

If you are intending to use Bitbucket Server for a production installation, see also [Using Bitbucket Server in the enterprise](#).

4. Set up the mail server

Configuring the Bitbucket Server email server allows users to receive a link from Bitbucket Server that lets them generate their own passwords. See [Setting up your mail server](#).

Download latest version



Make the most of Bitbucket Server

Automate your Bitbucket Server deployments

Bitbucket Data Center for enterprises

Deploy Bitbucket Server in AWS

Learn Git

Getting started with Git

Git resources

Be a Git guru

Bitbucket Server in action

Ecommerce speed

NASA rockets

Orbitz switches to Git

5. Get working with Bitbucket Server

Work with projects

Bitbucket Server manages related Git repositories as projects. Find out how to [set up projects and give your teams access](#) to those.

If you have existing projects that you want to manage in Bitbucket Server, then you'll want to read [Importing code from an existing project](#).

Integrate Bitbucket Server with other Atlassian applications

See [Integrating Bitbucket Server with Atlassian applications](#) for an overview of what is possible.

As a first step, see [JIRA integration](#) for information about using Bitbucket Server with JIRA Software.

If you want to see results from your continuous integration or build server in Bitbucket Server, see [Bamboo integration](#).

Use Bitbucket Server in your enterprise

If you are intending to use Bitbucket Server in large-scale production environments, see:

- [Using Bitbucket Server in the enterprise](#)
- [High availability for Bitbucket Server](#)
- [Scaling Bitbucket Server](#)
- [Scaling Bitbucket Server for Continuous Integration performance](#)
- [Bitbucket Data Center](#)
- [Bitbucket Server production server data](#)

Use Git

We have some information here to help you get going with Git:

- [Git Tutorials and Training](#)
- [Basic Git commands](#)
- [Permanently authenticating with Git repositories](#)
- [Using SSH keys to secure Git operations](#)
- [Git resources](#)

Supported platforms

This page lists the supported platforms for **Bitbucket Server 4.6.x**.

See [End of support announcements for Bitbucket Server](#) for upcoming changes to platforms supported by Bitbucket Server and Bitbucket Data Center.

Key:  = Supported  = Deprecated  = Not Supported

On this page

- [Hardware](#)
- [Operating systems](#)
- [Java](#)
- [Databases](#)
- [Atlassian application integrations](#)
- [Web browsers](#)
- [DVCS](#)
- [Additional tools](#)
- [Mail clients](#)

Hardware	
CPU	Evaluation: 1 core Production: 2+ cores
Memory	3GB+

Amazon Web Services (AWS)	 Bitbucket Server  Bitbucket Data Center  Bitbucket Mirror
Operating systems	
Linux	
Microsoft Windows	Up to 500 users
Apple Mac OS X	Evaluation only
Java	
Oracle Java	1.8
OpenJDK	 1.8u60  1.8u51  1.8u40+  1.8u25+  1.8u0 - 1.8u20
Databases	
PostgreSQL	 9.0 - 9.4  8.2 - 8.4
MySQL	Bitbucket Server:  5.7+  5.6.16+  5.6.0 – 5.6.15  5.5.x  5.1.x  MariaDB 10.1  MariaDB 10.0  MariaDB 5.5  Bitbucket Data Center
Microsoft SQL Server / Microsoft SQL Server Express	 2008 - 2014 See notes for migrating to named instance of SQL .
Oracle	 12c  11g
H2 (bundled)	 Bitbucket Server, evaluation only  Bitbucket Data Center  Bitbucket Mirror
HSQLDB (bundled)	Bitbucket Server only, evaluation only  Deprecated
Atlassian application integrations	
See Integrating Bitbucket Server with Atlassian applications for supported version combinations.	

Web browsers	
Chrome	✓ Latest stable version supported
Firefox	✓ Latest stable version supported
MS Edge	✓ Latest stable version supported
Internet Explorer	✓ 11
Safari	✓ Latest stable version supported
DVCS	
Git – client	✓ 1.6.6+ Or where Git LFS is used 1.8.2+
Git LFS - client	✓ 1.1.0+
Git – server	See notes for: Cygwin See notes for: Security vulnerability CVE-2016-2324 & CVE-2016-2315
✓ 2.8+ ✓ 2.7+ ✓ 2.6+ ✓ 2.5+ ✓ 2.4+ ✓ 2.3+ ✓ 2.2+ ✓ 2.1+	See notes for: 2.2.x - 2.4.0
✓ 2.0.4+ ✓ 2.0.0–2.0.1	See notes for: 2.0.2 and 2.0.3
✓ 1.9+	
✓ 1.8.4.4+ ✓ 1.8.0–1.8.4.2	See notes for: 1.8.4.3 and 1.8.3.x
Additional tools	
Perl (usually provided automatically with Git)	✓ 5.8.8+
Elasticsearch	✓ 2.3 ✗ Amazon Elasticsearch
Mail clients	
Apple Mail	✓ Apple Mail 4
Gmail	✓ Latest
iOS Devices	✓ iPhone, iPad
Microsoft Outlook	✓ Express, 2007, 2010

Outlook.com
Hotmail
Windows Live Mail

✓ Latest

Notes:

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.

Finally, we do not support deploying *any other applications* to the same Tomcat container that runs Bitbucket Server, especially if these other applications have large memory requirements or require additional libraries in Tomcat's `lib` subdirectory.

Supported platforms details

This page lists detailed notes regarding supported platforms for **Bitbucket Server 4.6.x**. See [Supported d platforms](#) for a list of supported platforms.

On this page

- [Hardware](#)
- [Operating systems](#)
- [Java](#)
- [Databases](#)
 - [HSQLDB](#)
 - [MySQL](#)
 - [MS SQL](#)
- [DVCS](#)
 - [Git - server](#)
 - [Git - client](#)

Hardware

- As well as the memory *allocated* for Tomcat (768MB is the default configuration and suitable for most uses), additional memory and CPU capacity is required to support Git operations. We recommend that you use a server with **at least 2GB of available memory**.
- The hardware requirements for a full production deployment depend on the number and frequency of Git operations and the number of active users. See [Scaling Bitbucket Server](#) for further discussion and for details of how memory is allocated for Bitbucket Server and Git.
- **Amazon Web Services (AWS)**
See [Recommendations for running Bitbucket Server in AWS](#) if you're running Bitbucket Server on AWS
 - For instance types and sizes, see [Recommendations for running Bitbucket Server in AWS](#).
 -  Bitbucket Data Center is not supported in AWS at this time.

Operating systems

- Bitbucket Server is a pure Java application and should run on any platform, provided all the Java requirements are satisfied.
- In production environments Bitbucket Server should be [run from a dedicated user account](#).
-  Apple Mac OS X is not supported for production deployment.
-  Microsoft Windows is not supported for 500+ Enterprise tiers.

Java

- Note that the Bitbucket Server installer will install a supported version of the Oracle Java JRE, which is only available to Bitbucket Server, if necessary. See [Running the Bitbucket Server installer](#).
- If you choose to pre-install a JRE, we recommend using Oracle JRE 8, which you can download from the [Oracle website](#).
- We recommend Java 1.8.0u40+ to avoid critical defects in older versions. If you are terminating SSL at Tomcat there is a known Java bug that has been fixed in version 1.8.0u51. Alternatively, you can continue using 1.8u40+ with the addition of a JVM flag. Further details: [Application crashes due to 'Internal Error \(sharedRuntime.cpp:833\)' caused by Java 8 bug](#).
- For OpenJDK, download and install instructions for Linux flavors are at <http://openjdk.java.net/install/>.
-  OpenJDK >= 1.8u25 and < 1.8u40 are not supported due to a severe defect, [Bug 1167153](#). This

- was fixed in 1.8u40.
-  OpenJDK 1.8u51 and 1.8u60 have known issues working with LDAP. See [CWD-4444](#) for details.

Pre-installed Java on some AWS EC2 Linux instances might be installed with a subset of features. See [SSH server fails to start on AWS EC2 instance for more information](#).

Databases

HSQldb

- Please see [connecting Bitbucket Server to an external database](#).
-  HSQldb is not supported in Bitbucket Data Center.
-  HSQldb support is deprecated as of Bitbucket Server 4.0+. New Bitbucket Server installs will bundle and use H2 as the default database for evaluation purposes.

H2

- H2 is bundled with Bitbucket Server and is only intended for evaluation use.
- H2 can be used with Bitbucket Data Center mirrors in production.

MySQL

MySQL, while supported by Bitbucket Server, is currently **not** recommended, especially for larger instances, due to inherent performance and deadlock issues that occur in this database engine under heavy load.

Affected systems may experience slow response times, deadlock errors and in extreme cases errors due to running out of database connections. These issues are intrinsic to MySQL (no other database engine supported by Bitbucket Server shares this behavior) and are due to the way MySQL performs row-level locking in transactions. See <http://dev.mysql.com/doc/refman/5.0/en/innodb-deadlocks.html> for some general information on this.

Bitbucket Server does its best to work around the MySQL behavior - see issues [STASH-4517](#), [STASH-4701](#) and others, for example. But under very heavy load you will generally get better performance with any of the other database engines supported by Bitbucket Server (such as PostgreSQL, which is also freely available) than you will with MySQL.

- MariaDB 10.1 releases are still classed as betas and are not supported
-  MySQL 5.6.15 and earlier: Note that Bitbucket Server *does not support* versions of MySQL 5.6 earlier than 5.6.16 at all, because of bugs in its query optimizer ([#68424](#), [#69005](#)). See [Connecting Bitbucket Server to MySQL](#) for more information.
-  MySQL 5.7+ is not supported.
-  MySQL is not supported at all in Bitbucket Data Center.

MS SQL

- **Named Instances:** If you have a named instance on your server, it is not possible to migrate from the internal database to a named instance of SQL Server using the UI procedure. You will need to manually edit the `bitbucket.properties` file as described on the [Connecting to named instances in SQL Server from Bitbucket Server](#) Knowledge Base article.

DVCS

Git - server

In general, we recommend using the most recent version of Git on both the Bitbucket Server instance and clients, where possible, and subject to the following notes and exceptions.

- The version of Git installed on machines that interact with Bitbucket Server must be compatible with the version of Git installed for use by the Bitbucket Server instance.
-  **Cygwin Git** is *not supported* for use on Windows servers, regardless of version.
-  **Git 1.8.3.x** has some performance regressions which may cause problems in Bitbucket Server with

large repositories.

- ✗ **Git 1.8.4.3** is not supported due to a critical bug in how symbolic refs are handled which breaks pushing and pulling for repositories with pull requests. ([Details](#))

BSERV-4101 - Clone and fetch fail with "protocol error: impossibly long line"
CLOSED

- ✗ **Git 2.0.2 and 2.0.3** are not supported due to a critical bug in `git diff-tree` which breaks Bitbucket Server's commit page. ([Details](#))

BSERV-5052 - Commit messages are wrong when using Git 2.0.2 and 2.0.3
CLOSED

- Git 2.2.x - 2.4.0** have some performance issues when interacting with NFS. Hence, these versions are currently not supported for Bitbucket Data Center or for Bitbucket Server installations that use NFS mounts for the home directory ([Details](#))

[Security vulnerability CVE-2016-2324 & CVE-2016-2315] affects multiple Git versions. Both *server* and *client* Git installations should be updated to a patched maintenance version: 2.4.11, 2.5.5, 2.6.6 or 2.7.4 or newer. For instructions see [Installing and upgrading Git](#).

The table below lists the versions of Git that have been specifically tested against the **Bitbucket Server 4.6.x** releases.

Linux	Windows
2.8.2	2.8.2
2.7.4	2.7.4
2.6.6	2.6.4
2.5.5	2.5.3
2.4.11	2.4.6
2.3.10	2.3.7.1
2.2.3	
2.1.4	
2.0.5	
1.9.5	1.9.5.1
1.8.0.3	1.8.0
1.8.1.5	1.8.1.2
	1.8.3
1.8.2.3	1.8.4
	1.8.5.2
1.8.3.4	
1.8.4.5	
1.8.5.6	

Git - client

[Security vulnerability CVE-2016-2324 & CVE-2016-2315] affects multiple Git versions. Both *server* and *client* Git installations should be updated to a patched maintenance version: 2.4.11, 2.5.5, 2.6.6 or 2.7.4 or newer. For instructions see [Installing and upgrading Git](#).

Using Bitbucket Server in the enterprise

This page...

... describes best practice for using Bitbucket Server in enterprise environments.

If you're evaluating Bitbucket Server...

... we suggest that you begin with [Getting started](#), instead of this page.

See also...

... [Bitbucket Enterprise Resources](#) for a comparison of Bitbucket Server and Bitbucket Data Center, our

clustered Bitbucket Server solution.

Atlassian Bitbucket Server is the Git code management solution for enterprise teams. It allows everyone in your organization to easily collaborate on your Git repositories, while providing enterprise-grade support for:

- user authentication
- repository security
- integration with your existing databases and development environment.

Atlassian offers two deployment options for Bitbucket Server, to provide enterprise scaling and infrastructure flexibility, and to give administrators control over how Bitbucket Server fits into their environment:

Bitbucket Server

For most organizations, a single instance of Bitbucket Server provides good performance. Continue reading this page for guidance on best practices in setting up a Bitbucket Server instance in a production environment.

Bitbucket Data Center

For larger enterprises that require HA and greater performance at scale, [Bitbucket Data Center](#) uses a cluster of Bitbucket Server nodes to provide Active/Active failover, and is the deployment option of choice.

Your single instance of Bitbucket Server can be easily upgraded to Bitbucket Data Center when the time comes.

On this page:

- [Platform requirements for hosting Bitbucket Server](#)
- [Performance considerations with Bitbucket Server](#)
- [High availability with Bitbucket Server](#)
- [Scalability](#)
- [Provisioning Bitbucket Server](#)
- [Setting up Bitbucket Server in a production environment](#)
- [Administering Bitbucket Server in a production environment](#)

Platform requirements for hosting Bitbucket Server

Although Bitbucket Server can be run on Windows, Linux and Mac systems, for enterprise use we only recommend, and support, Linux. This recommendation is based on our own testing and experience with using Bitbucket Server.

See the [Supported platforms](#) page for details of the supported versions of Java, external databases, web browsers and Git.

See [Installing Bitbucket Data Center](#) for detailed information about Bitbucket Data Center requirements.

Performance considerations with Bitbucket Server

In general, Bitbucket Server is very stable and has low memory consumption. There are no scalability limits other than for Git hosting operations (clone in particular). We know this is the scalability limit of the product; the limit is proportional to the number of cores on the system.

As an example, data collected from an internal Bitbucket Server instance indicate that for a team of approximately 50 developers, with associated continuous integration infrastructure, we see a peak concurrency of 30 simultaneous clone operations and a mean of 2 simultaneous clone operations. We conservatively expect that a customer with similar usage patterns would be capable of supporting 1000 users on a machine with 40 cores and a supporting amount of RAM. While we expect a peak concurrency larger than 40, Bitbucket Server is designed to queue incoming requests so as to avoid overwhelming the server.

Bitbucket Server – see [Bitbucket Server production server data](#) for data from the Bitbucket Server production instance we run internally at Atlassian.

Bitbucket Data Center – see [Bitbucket Data Center Performance](#) for the results of our performance testing for clusters of different sizes.

High availability with Bitbucket Server

If Bitbucket Server is a critical part of your development workflow, maximizing Bitbucket Server availability becomes an important consideration.

Bitbucket Server – see [High availability for Bitbucket Server](#) for the background information you need to set up Bitbucket Server in a highly available configuration.

Bitbucket Data Center – see [Failover for Bitbucket Data Center](#) for information about how Bitbucket Data Center provides HA and almost instant failover.

Scalability

Bitbucket Server is built with enterprise scaling and infrastructure flexibility in mind, giving administrators control over how Bitbucket Server fits into their environment:

- For most organizations, a single instance of Bitbucket Server provides good performance. Continue reading this page for guidance on best practice in setting up a Bitbucket Server instance in a production environment.
- For larger enterprises that require HA and greater performance at scale, [Bitbucket Data Center](#) uses a cluster of Bitbucket Server nodes and is the deployment option of choice.

Your single instance of Bitbucket Server can be easily upgraded to Bitbucket Data Center when the time comes.

Bitbucket Server – see [Scaling Bitbucket Server](#) for information about how you can tune your Bitbucket Server instance to grow with your organization's needs. See also [Scaling Bitbucket Server for Continuous Integration performance](#) for information specific to Bitbucket Server performance when CI tools poll Bitbucket Server for changes.

Bitbucket Data Center – see [Adding cluster nodes to Bitbucket Data Center](#) for information about how you can rapidly provision extra capacity without downtime.

Provisioning Bitbucket Server

Some possible approaches to provisioning Bitbucket Server include:

- [Running the Bitbucket Server installer](#) in either console or unattended mode
- [Bitbucket Data Center](#) - clustered Bitbucket Server
- [Docker container image for Bitbucket Server](#) (currently only supported for evaluations)

Setting up Bitbucket Server in a production environment

When setting up Bitbucket Server for a production or enterprise environment, we highly recommend that you configure the following aspects:

Run Bitbucket Server as a dedicated user

- For production environments Bitbucket Server should be run from a dedicated user account with restricted privileges. See [Running Bitbucket Server with a dedicated user](#).

Install Bitbucket Server as a service

- See [Running the Bitbucket Server installer](#).

Use an external database

- For production environments Bitbucket Server should use an external database, rather than the embedded database. Set up your external DBMS (for example MySQL) before starting Bitbucket Server for the first time. This allows you to connect Bitbucket Server to that DBMS using the Setup

Wizard that launches when you first run Bitbucket Server. See [Connecting Bitbucket Server to an external database](#).

Connect to your existing user directory

- Connect Bitbucket Server to your existing user directory (for example Active Directory). See [External user directories](#).

Secure the Bitbucket home directory

- For production environments the Bitbucket Server home directory should be secured against unauthorized access. See [Bitbucket Server home directory](#).

Secure Bitbucket Server with HTTPS

- Access to Bitbucket Server should be secured using HTTP over SSL, especially if your data is sensitive and Bitbucket Server is exposed to the internet. See [Securing Bitbucket Server with HTTPS](#).

Enable SSH access to Git repositories

- Enable SSH access for your Bitbucket Server users to Git repositories in Bitbucket Server so that they can add their own SSH keys to Bitbucket Server, and then use those SSH keys to secure Git operations between their computer and the Bitbucket Server instance. See [Enabling SSH access to Git repositories in Bitbucket Server](#).

Change the context path for Bitbucket Server

- If you are running Bitbucket Server behind a proxy, or you have another Atlassian application (or any Java web application), available at the same hostname and context path as Bitbucket Server, then you should set a unique context path for Bitbucket Server. See [Moving Bitbucket Server to a different context path](#).

Administering Bitbucket Server in a production environment

Upgrading Bitbucket Server

- For production environments we recommend that you test the Bitbucket Server upgrade on a QA server before deploying to production. See the [Bitbucket Server upgrade guide](#).

Backups and recovery

- **We highly recommend** that you establish a data recovery plan that is aligned with your company's policies. See [Data recovery and backups](#) for information about tools and backup strategies for Bitbucket Server.

Logging

- Bitbucket Server instance logs can be found in <Bitbucket home directory>/log. Logs for the bundled Tomcat webserver can be found in <Bitbucket Server installation directory>/log. See [Bitbucket Server debug logging](#).
- Bitbucket Server displays recent audit events for each repository and project (only visible to Bitbucket Server admins and system admins), and also creates full audit log files that can be found in the <Bitbucket home directory>/audit/logs directory. Note that Bitbucket Server has an upper limit to the number of log files it maintains, and deletes the oldest file when a new file is created – we recommend an automated backup of log files. See [Audit logging in Bitbucket Server](#).

Installing and upgrading Git

[Security vulnerability CVE-2016-2324 & CVE-2016-2315] If you are running a Git version older than 2.4.11, 2.5.5, 2.6.6 or 2.7.4 (all released  17 Mar 2016), you should upgrade Git as soon as possible.

This page describes how to:

- [Check your version of Git](#)
- [Install or upgrade Git on Linux](#)
- [Install or upgrade Git on Mac OS X](#)
- [Install or upgrade Git on Windows](#)

The information on this page applies to installing or upgrading Git on either your local machine, or on the Bitbucket Server instance.

Check your version of Git

The versions of Git supported by Bitbucket Server are listed on [Supported platforms](#).

You can check your current version of Git by running the `git --version` command in a terminal (Linux, Mac OS X) or command prompt (Windows).

For example:

```
git --version
git version 2.7.4
```

If you don't see a supported version of Git, you'll need to either upgrade Git or perform a fresh install, as described below.

Install or upgrade Git on Linux

Use your package manager to install Git. For example, on Ubuntu 13.10, use:

```
sudo apt-get install git
```

Alternative download options are:

- Download the latest stable Git release from the [Git website](#).
- If you are using a different Linux distribution, you may need to use a different package repository to get the latest stable version of Git.
- If you need the most recent version of Git, you might need [to install it from source](#).

Now [check the Git version](#) – you should see the new version of Git.

If you still can't see the expected Git version, you may need to add the Git install location to your path. Open your `~/.profile` file in a text editor and add this line, where `<path/to/git>` is the install location for Git:

```
export PATH=$PATH:<path/to/git>
```

You can use the `which git` command to find the install location for Git.

Install or upgrade Git on Mac OS X

Download the latest stable Git release from the [Git website](#).

Click on the downloaded `.dmg` file, then double-click the `.pkg` icon to run the installer. This will install the new version of Git over the existing version:



Alternatively, you can:

- Use the native Git bundled with OS X.
- Use [Homebrew](#) to download and install Git.

Now [check the Git version](#) – you should see the new version of Git.

If you still can't see the Git version, you may need to add the Git install location to your path. Open your `~/.profile` file in a text editor and add this line, where `<path/to/git>` is the install location for Git:

```
export PATH=$PATH:<path/to/git>
```

You can use the `which git` command to find the install location for Git.

Install or upgrade Git on Windows

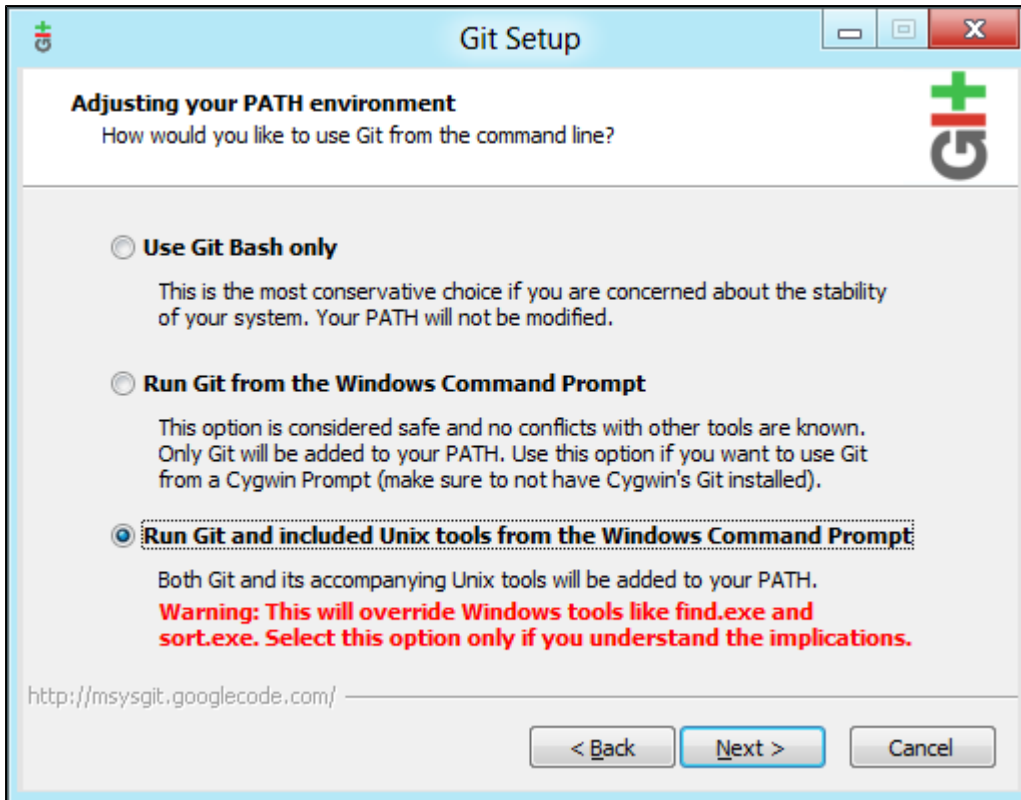
Download the latest stable Git release from the [Git website](#).

Run the Git installer, ensuring that you install into the same location as any existing Git installation. You can use `where git` to locate existing installations. Installing Git for Windows (msysGit) also installs a supported version of Perl.

To ensure that `git.exe` is available in the path, choose either:

- **Run Git from the Windows Command Prompt**, or
- **Run Git and included Unix tools from the Windows Command Prompt**.

Do **not** select **Use Git Bash only** when installing or upgrading Git for the Bitbucket Server instance -- *this will not work with Bitbucket Server*.



Now, [check the Git version](#) – you should see the new version of Git.

! msysGit is the *only supported distribution* when running Bitbucket Server on Windows. Cygwin Git is *not supported* and has known issues.

If you have successfully installed msysGit but you receive the error "Unable to find git!" when installing Bitbucket Server, you should abort the installation, restart the Windows server, then restart the Bitbucket Server installation.

Restart Bitbucket Server if necessary

If you've been installing or upgrading Git for the Bitbucket Server instance, rather than for your local machine, you'll need to stop and restart Bitbucket Server so that it will pick up the upgraded version of Git. See [Starting and stopping Bitbucket Server](#) for details.

Configuring JIRA integration in the Setup Wizard

This page describes the 'JIRA Software integration' screen of the Bitbucket Server Setup Wizard that runs automatically when you launch Bitbucket Server for the first time.

The Setup Wizard guides you in configuring the Bitbucket Server connection with JIRA Software using the most common options. You can also configure JIRA Software integration from the Bitbucket Server administration screens at any time after completing the Setup Wizard.

There are two aspects to integrating Bitbucket Server with JIRA Software:

- Linking JIRA Software and Bitbucket Server to enable the integration features. See [JIRA integration](#).
- Delegating Bitbucket Server user and group management to your JIRA Software server. See [Connecting Bitbucket Server to JIRA for user management](#).

On this page:

- [Connecting to JIRA Software in the Setup Wizard](#)
- [Troubleshooting](#)

- [Notes](#)

Connecting to JIRA Software in the Setup Wizard

To configure JIRA Software integration while running the Bitbucket Server Setup Wizard:

1. Configure the following setting in JIRA Software: [Configuring JIRA Software application options](#).
2. Click **Integrate with JIRA** and enter the following information when you get to the 'Connect to JIRA' step of the setup wizard:

JIRA base URL	The web address of your JIRA server. Examples are: <code>http://www.example.com:8080/jira/</code> <code>http://jira.example.com</code>
JIRA admin username	The credentials for a user with the 'JIRA System Administrators' global permission in JIRA.
JIRA password	
Bitbucket Server base URL	JIRA will use this URL to access your Bitbucket Server instance. The URL you give here will override the base URL specified in your Bitbucket Server administration console, for the purposes of the JIRA connection.

3. Click **Connect**.
4. Finish the setup process.

JIRA integration

Use JIRA as a central server for user management or connect your issues and changesets simply by adding issue keys to your commit messages.

- Automatically import all your JIRA users.
- See what code changes are related to a specific JIRA issue.
- Quickly navigate to JIRA issues that are linked to commits.
- Keep track of bug-fixes.

Create JIRA connection

JIRA base URL *

For example: `http://jira.atlassian.com`

JIRA administrator username *

This user must have system administrator rights in JIRA

JIRA password *

The JIRA user's password

Stash base URL *

JIRA will access Stash from this URL

Using JIRA as my user database

If you have JIRA 4.3 or later, Stash can use JIRA for user management. This is not recommended for more than 500 users. [Learn more about JIRA user management](#).

☐ Use JIRA as my user database

[Connect](#) [Skip](#)

Troubleshooting

▼ [Click to see troubleshooting information...](#)

This section describes the possible problems that may occur when integrating your application with JIRA via the setup wizard, and the solutions for each problem.

Symptom	Cause	Solution
The setup wizard displays one of the following error messages: - Failed to create application link from JIRA server at <URL> to this <application> server at <URL>. - Failed to create application link from this <application> server at <URL> to JIRA server at <URL>. - Failed to authenticate application link from JIRA server at <URL> to this <application> server at <URL>. - Failed to authenticate application link from <application> server at <URL> to this JIRA server at <URL>.	The setup wizard failed to complete registration of the peer-to-peer application link with JIRA. JIRA integration is only partially configured.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
The setup wizard displays one of the following error messages: - Failed to register <application> configuration in JIRA for shared user management. Received invalid response from JIRA: <response> - Failed to register <application> configuration in JIRA for shared user management. Received: <response>	The setup wizard failed to complete registration of the client-server link with JIRA for user management. The peer-to-peer link was successfully created, but integration is only partially configured.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
The setup wizard displays the following error message: Error setting Crowd authentication	The setup wizard successfully established the peer-to-peer link with JIRA, but could not persist the client-server link for user management in your config.xml file. This may be caused by a problem in your environment, such as a full disk.	Please investigate and fix the problem that prevented the application from saving the configuration file to disk. Then remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
The setup wizard displays the following error message: Error reloading Crowd authentication	The setup wizard has completed the integration of your application with JIRA, but is unable to start synchronizing the JIRA users with your application.	Restart your application. You should then be able to continue with the setup wizard. If this solution does not work, please contact Atlassian Support.

The setup wizard displays the following error message: An error occurred: <code>java.lang.IllegalStateException: Could not create the application in JIRA/Crowd (code: 500)</code>. Please refer to the logs for details.	The setup wizard has not completed the integration of your application with JIRA. The links are only partially configured. The problem occurred because there is already a user management configuration in JIRA for this <application> URL.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
No users can log in after you have set up the application with JIRA integration.	Possible causes: - There are no users in the group that you specified on the 'Connect to JIRA' screen. - For FishEye: There are no groups specified in the 'groups to synchronize' section of your administration console. - For Stash: You may not have granted any JIRA groups or users permissions to log in to Stash.	Go to JIRA and add some usernames to the group. - For FishEye: Go to the FishEye administration screens and specify at least one group to synchronize. The default is 'jira-users'. - For Stash: Grant the Stash User permission to the relevant JIRA groups on the Stash Global permissions page. If this solution does not work, please contact Atlassian Support.

Solution 1: Removing a Partial Configuration – The Easiest Way

If the application's setup wizard fails part-way through setting up the JIRA integration, you may need to remove the partial configuration from JIRA before continuing with your application setup. Please follow the steps below.

Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup wizard:

- Log in to JIRA as a user with the '**JIRA System Administrators**' global permission.
- Click the '**Administration**' link on the JIRA top navigation bar.
- Remove the application link from JIRA, if it exists:
 - Click **Application Links** in the JIRA administration menu. The 'Configure Application Links' page will appear, showing the application links that have been set up.
 - Look for a link to your application. It will have a base URL of the application linked to JIRA. For example:
 - If you want to remove a link between JIRA and FishEye, look for the one where the **Application URL** matches the base URL of your FishEye server.
 - If you want to remove a link between JIRA and Confluence, look for the one where the **Application URL** matches the base URL of your Confluence server.
 - If you want to remove a link between JIRA and Stash, look for the one where the **Application URL** matches the base URL of your Stash server.
 - Click **Delete** next to the application link that you want to delete.
 - A confirmation screen will appear. Click **Confirm** to delete the application link.
- Remove the user management configuration from JIRA, if it exists:
 - Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
 - In JIRA 4.3: Click '**Other Applications**' in the '**Users, Groups & Roles**' section of the JIRA administration screen.
 - In JIRA 4.4: Select '**Administration**' > '**Users**' > '**JIRA User Server**'.
 - Look for a link to your application. It will have a name matching this format:

<Type> - <HostName> - <Application ID>

For example:

```
FishEye / Crucible - localhost -
92004b08-5657-3048-b5dc-f886e662ba15
```

Or:

```
Confluence - localhost -
92004b08-5657-3048-b5dc-f886e662ba15
```

If you have multiple servers of the same type running on the same host, you will need to match the application ID of your application with the one shown in JIRA. To find the application ID:

- Go to the following URL in your browser:

```
<baseUrl>/rest/applinks/1.0/manifest
```

Replace <baseUrl> with the base URL of your application.

For example:

```
http://localhost:8060/rest/applinks/1.0/manifest
```

- The application links manifest will appear. Check the application ID in the <id> element.
- c. In JIRA, click '**Delete**' next to the application that you want to remove.
5. Go back to the setup wizard and try the 'Connect to JIRA' step again.

Solution 2: Removing a Partial Configuration – The Longer Way

If solution 1 above does not work, you may need to remove the partial configuration and then add the full integration manually. Please follow these steps:

1. Skip the 'Connect to JIRA' step and continue with the setup wizard, to complete the initial configuration of the application.
2. Log in to JIRA as a user with the '**JIRA System Administrators**' global permission.
3. Click the '**Administration**' link on the JIRA top navigation bar.
4. Remove the application link from JIRA, if it exists:
 - a. Click **Application Links** in the JIRA administration menu. The 'Configure Application Links' page will appear, showing the application links that have been set up.
 - b. Look for a link to your application. It will have a base URL of the application linked to JIRA. For example:
 - If you want to remove a link between JIRA and FishEye, look for the one where the **Application URL** matches the base URL of your FishEye server.
 - If you want to remove a link between JIRA and Confluence, look for the one where the **Application URL** matches the base URL of your Confluence server.
 - If you want to remove a link between JIRA and Stash, look for the one where the **Application URL** matches the base URL of your Stash server.
 - c. Click **Delete** next to the application link that you want to delete.
 - d. A confirmation screen will appear. Click **Confirm** to delete the application link.
5. Remove the user management configuration from JIRA, if it exists:
 - a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
 - In JIRA 4.3: Click '**Other Applications**' in the '**Users, Groups & Roles**' section of the JIRA administration screen.
 - In JIRA 4.4: Select '**Administration**' > '**Users**' > '**JIRA User Server**'.
 - b. Look for a link to your application. It will have a name matching this format:

```
<Type> - <HostName> - <Application ID>
```

For example:

```
FishEye / Crucible - localhost -  
92004b08-5657-3048-b5dc-f886e662ba15
```

Or:

```
Confluence - localhost -  
92004b08-5657-3048-b5dc-f886e662ba15
```

If you have multiple servers of the same type running on the same host, you will need to match the application ID of your application with the one shown in JIRA. To find the application ID:

- Go to the following URL in your browser:

```
<baseUrl>/rest/applinks/1.0/manifest
```

Replace `<baseUrl>` with the base URL of your application.

For example:

```
http://localhost:8060/rest/applinks/1.0/manifest
```

- The application links manifest will appear. Check the application ID in the `<id>` element.
- c. In JIRA, click **Delete** next to the application that you want to remove.
- 6. Add the application link in JIRA again, so that you now have a two-way trusted link between JIRA and your application:
 - a. Click **Add Application Link**. Step 1 of the link wizard will appear.
 - b. Enter the **server URL** of the application that you want to link to (the 'remote application').
 - c. Click **Next**.
 - d. Enter the following information:
 - **Create a link back to this server** – Check to add a two-way link between the two applications.
 - **Username and Password** – Enter the credentials for a username that has administrator access to the remote application.
Note: These credentials are only used to authenticate you to the remote application, so that Application Links can make the changes required for the new link. The credentials are not saved.
 - **Reciprocal Link URL** – The URL you give here will override the base URL specified in your remote application's administration console, for the purposes of the application links connection. Application Links will use this URL to access the remote application.
 - e. Click **Next**.
 - f. Enter the information required to configure authentication for your application link:
 - **The servers have the same set of users** – Check this box, because the users are the same in both applications.
 - **These servers fully trust each other** – Check this box, because you trust the code in both applications and are sure both applications will maintain the security of their private keys.
For more information about configuring authentication, see [Configuring authentication for an application link](#).
 - g. Click **Create**.
- 7. Configure a new connection for user management in JIRA:
 - a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:

- In JIRA 4.3: Click '**Other Applications**' in the '**Users, Groups & Roles**' section of the JIRA administration screen.
- In JIRA 4.4: Select '**Administration**' > '**Users**' > '**JIRA User Server**'.
- b. **Add** an application.
- c. Enter the **application name** and **password** that your application will use when accessing JIRA.
- d. Enter the **IP address** or addresses of your application. Valid values are:
 - A full IP address, e.g. 192.168.10.12.
 - A wildcard IP range, using CIDR notation, e.g. 192.168.10.1/16. For more information, see the introduction to [CIDR notation on Wikipedia](#) and [RFC 4632](#).
 - **Save** the new application.
- 8. Set up the JIRA user directory in the application.
 - For Confluence:
 - a. Go to the **Confluence Administration Console**.
 - b. Click '**User Directories**' in the left-hand panel.
 - c. **Add** a directory and select type '**Atlassian JIRA**'.
 - d. Enter the following information:
 - **Name** – Enter the name of your JIRA server.
 - **Server URL** – Enter web address of your JIRA server. Examples:

```
http://www.example.com:8080/jira/
http://jira.example.com
```

- **Application name** and **Application password** – Enter the values that you defined for Confluence in the settings on JIRA.

- e. Save the directory settings.
- f. Define the **directory order** by clicking the blue up- and down-arrows next to each directory on the '**User Directories**' screen.

For details see [Connecting to Crowd or JIRA for User Management](#).

- For FishEye/Crucible:
 - a. Click **Authentication** (under 'Security Settings').
 - b. Click **Setup JIRA/Crowd authentication**. Note, if LDAP authentication has already been set up, you will need to remove that before connecting to JIRA for user management.
 - c. Make the following settings:

Authenticate against	Select a JIRA instance
Application name and password	Enter the values that you defined for your application in the settings on JIRA.
JIRA URL	The web address of your JIRA server. Examples: <pre>http://www.example.com:8080/jira/ http://jira.example.com</pre>
Auto-add	Select Create a FishEye user on successful login so that your JIRA users will be automatically added as a FishEye user when they first log in.
Periodically synchronise users with JIRA	Select Yes to ensure that JIRA will synchronize all changes in the user information on a regular basis. Change the value for Synchronise Period if required.

When Synchronisation Happens	Select an option depending on whether you want to allow changes to user attributes from within FishEye.
Single Sign On	Select Disabled . SSO is not available when using JIRA for user management and if enabled will make the integration fail.

- d. Click **Next** and select at least one user group to be synchronised from JIRA. If necessary, you could create a new group in JIRA, such as 'fisheye-users', and select this group here.
 - e. Click **Save**.
 - For Stash:
 - a. Go to the Stash administration area.
 - b. Click **User Directories** in the left-hand panel.
 - c. **Add** a directory and select type **Atlassian JIRA**.
 - d. Enter the following information:
 - **Name** – Enter the name of your JIRA server.
 - **Server URL** – Enter web address of your JIRA server. Examples:


```
http://www.example.com:8080/jira/
http://jira.example.com
```
 - **Application name** and **Application password** – Enter the values that you defined for Stash in the settings on JIRA.
 - e. Save the directory settings.
 - f. Define the directory order by clicking the blue up- and down-arrows next to each directory on the 'User Directories' screen.
- For details see [Connecting Stash to JIRA for user management](#).

Notes

When you connect to JIRA Software in the setup wizard, the setup procedure will configure *OAuth authentication* between Bitbucket Server and JIRA Software. See [Configuring OAuth authentication for an application link](#) for more information.

Getting started with Git and Bitbucket Server

Atlassian Bitbucket Server is the Git repository management solution for enterprise teams. It allows everyone in your organization to easily collaborate on your Git repositories.

This page will guide you through the basics of Bitbucket Server. By the end you should know how to:

- [Create accounts for your collaborators, and organize these into groups with permissions.](#)
- [Create a project and set up permissions.](#)
- [Create repositories, and know the basic commands for interacting with them.](#)

Assumptions

This guide assumes that you don't have prior experience with Git. But we do assume that:

- You have Git version 1.7.6 or higher installed on your local computer.
- You are using a [supported browser](#).
- You have Bitbucket Server installed and running. If you haven't, see [Getting started](#).

Please read [Git resources](#) or check out our [Git tutorials](#) for tips on getting

started with Git.

Add users to Bitbucket Server and grant permissions

The first thing you can do in Bitbucket Server is to add collaborators.

To add users within Bitbucket Server

1. Go to the Bitbucket Server administration area by clicking the cog , then click **Users** in the Admin screen (under Accounts):
2. Click **Create user** to go directly to the user creation form.
3. Once you've created a user, click **Change permissions** to set up their access permissions.



Make the most of Bitbucket Server

[Automate your Bitbucket Server deployments](#)

[Bitbucket Data Center for enterprises](#)

[Deploy Bitbucket Server in AWS](#)

Learn Git

[Getting started with Git](#)

[Git resources](#)

[Be a Git guru](#)

Bitbucket Server in action

[Ecommerce speed](#)

[NASA rockets](#)

[Orbitz switches to Git](#)

There are 4 levels of user authentication:

- **System Administrator** — can access all the configuration settings of the Bitbucket Server instance.
- **Administrator** — same as System Admins, but they can't modify file paths or the Bitbucket Server instance settings.
- **Project Creator** — can create, modify and delete projects.
- **Bitbucket Server User** — active users who can access Bitbucket Server.

See [Users and groups](#) for more information about authentication.

See [External user directories](#) if you have existing user identities you wish to use with Bitbucket Server.

Create your first project and share it with collaborators

Create your project

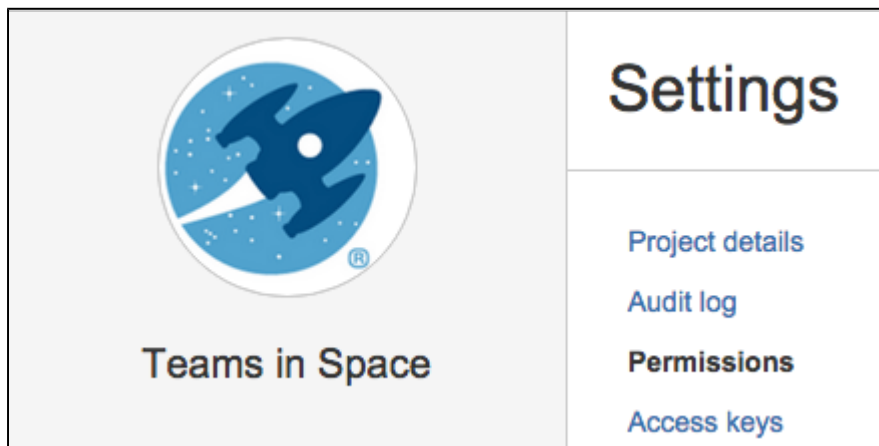
The next thing you do in Bitbucket Server is to create a project. You'll add repositories to this project later.

Go to 'Projects' and click **Create project**. Complete the form and submit it to create your new project. See [Creating projects](#) for more information.

Open project access to others

If you are a project administrator, you can grant project permissions to other collaborators.

Click **Settings** then **Permissions** for the project:



The 'Project permissions' page allows you to add users and groups to a project you've already created.

There are 3 levels of project access:

- **Admin** — can create, edit and delete repositories and projects, and configure permissions for projects.
- **Write** — can push to and pull from all the repositories in the project.
- **Read** — can only browse code and comments in, and pull from, the repositories in the project.

See [Using project permissions](#) for more information.

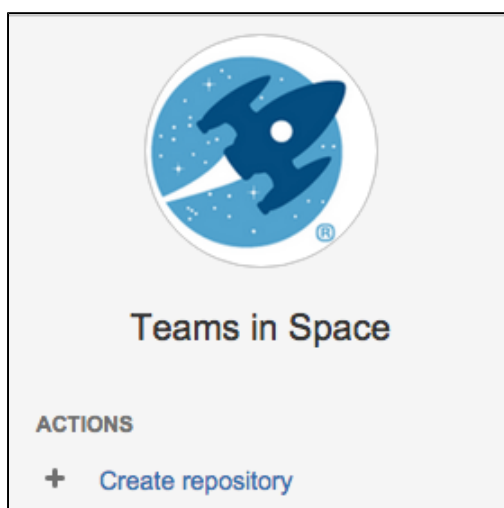
Create a repository and get your code into Bitbucket Server

Create a repository

If you are a project administrator, you can create repositories in the project.

Once a repository is created, the project permissions are applied to the repository. That means all repositories created in a project share the same access and permission settings. If you already have a Git project you'd like to use, see [Importing code from an existing project](#).

Click **Create repository** to open the repository creation form:



Once submitted you will be taken directly to your repository homepage. As there is no content in your

repository yet, you'll see some instructions to help you push code to your repository. See [Creating repositories](#) for more information.

Clone and push

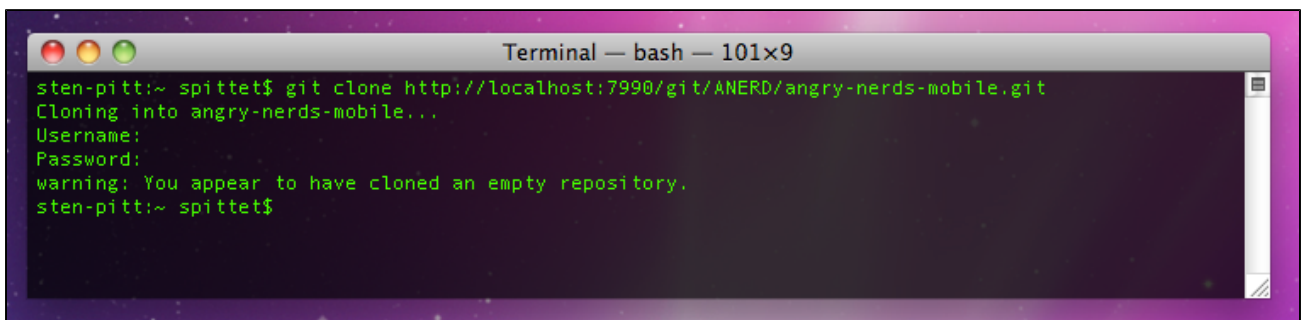
This section describes how to [clone the repository you just created](#) and then [push](#) a [commit](#) back to it. You can see the clone URL to use at the top right of the screen. [SSH access](#) may be available.

In a terminal, run the following command (replace <bitbucketURL> with the URL for your instance of Bitbucket Server):

```
git clone <bitbucketURL>/git/<projectname>/<reponame>.git
```

Use your Bitbucket Server username and password.

The result in your terminal should be similar to what you can see in the screenshot below.



You should now have a new empty directory tracked by Git, in the user space of your local machine. Let's add some content and push it back to Bitbucket Server.

In your <reponame> directory, create a text file named helloworld.txt and write "Hello World" in it.

Now run the following command in your terminal

```
cd <reponame>  
git add .  
git commit -m "My first commit"  
git push origin master
```

If everything went fine, when you refresh the Bitbucket Server screen, you will see that the homepage of your repository has been replaced with a file browser showing you a link to helloworld.txt.

There you go, you're ready to get coding with your collaborators.

For more information about getting your code into Bitbucket Server, see [Importing code from an existing project](#). Note that huge Git repositories (larger than a few GBs) are likely to impact the performance of the Git client – see [this discussion](#).

Check out our [Git tutorials and training](#) for more information, and have a look at this list of [basic Git commands](#) that you will probably use often.

Importing code from an existing project

When creating a new repository, you can import code from an existing project into Bitbucket Server. You can do this by first cloning the repository to your local system and then pushing to an empty Bitbucket Server repository.

On this page:

- [Import an existing, unversioned code project into Bitbucket Server](#)

- [Import an existing Git project into Bitbucket Server](#)
- [Mirror an existing Git repository](#)

Import an existing, unversioned code project into Bitbucket Server

If you have code on your local machine that is not under source control, you can put it under source control and import it into Bitbucket Server.

Assuming you have Git installed on your local machine, then:

1. Locally, change to the root directory of your existing source.
2. [Initialize the project](#) by running the following commands in the terminal:

```
git init
git add --all
git commit -m "Initial Commit"
```

3. Log into Bitbucket Server and [create a new repository](#).
4. Locate the clone URL in the nav panel on the left (for example: `https://username@your.bitbucket.domain:7999/yourproject/repo.git`).
5. Push your files to the repository by running the following commands in the terminal (change the URL accordingly):

```
git remote add origin
https://username@your.bitbucket.domain:7999/yourproject/repo.git
git push -u origin master
```

6. Done! Your repository is now available in Bitbucket Server.

Import an existing Git project into Bitbucket Server

You can import your existing Git repository into an empty repository in Bitbucket Server. When you do this, Bitbucket Server maintains your commit history.

1. [Check out the repository from your existing Git host](#). Use the `--bare` parameter:

```
git clone --bare
https://username@bitbucket.org/exampleuser/old-repository.git
```

2. Log into Bitbucket Server and [create a new repository](#) (we've called it `repo.git` in this example).
3. Locate the clone URL in the nav panel on the left (for example: `https://username@your.bitbucket.domain:7999/yourproject/repo.git`).
4. Add Bitbucket Server as another remote in your local repository:

```
cd old-repository
git remote add bitbucket
https://username@your.bitbucket.domain:7999/yourproject/repo.git
```

5. Push all branches and tags to the new repository in Bitbucket Server:

```
git push --all bitbucket
git push --tags bitbucket
```

6. Remove your temporary local repository:

```
cd ..  
rm -rf old-repository
```

Mirror an existing Git repository

You can mirror an existing repository into a repository hosted in Bitbucket Server.

1. Check out the repository from your existing Git host. Use the `--mirror` parameter:

```
git clone --mirror  
https://username@bitbucket.org/exampleuser/repository-to-mirror.git
```

2. Log into Bitbucket Server and [create a new repository](#) (we've called it `repo.git` in this example).
3. Locate the clone URL in the nav panel on the left (for example: `https://username@your.bitbucket.domain:7999/yourproject/repo.git`).
4. Add Bitbucket Server as another remote in your local repository:

```
git remote add bitbucket  
https://username@your.bitbucket.domain:7999/yourproject/repo.git
```

5. Then push all branches and tags to Bitbucket Server:

```
git push --all bitbucket  
git push --tags bitbucket
```

6. Use `git fetch --prune origin` ('--prune' will remove any branches that no longer exist in the remote) followed by the `git push` commands from step 5 to update the Bitbucket Server mirror with new changes from the upstream repository.

Bitbucket Server tutorials

If you're just starting out with Bitbucket Server, then this is the place for you. Come with us on a journey to discover all that Bitbucket Server has to offer using our Teams in Space scenario.

Before you continue, make sure you've [installed Bitbucket Server](#) already.

Jump into a tutorial when you're ready:

Work with Bitbucket Server and SourceTree

With Bitbucket Server running, learn how to get your work done

More to come!

Tutorial: Learn Git with Bitbucket Server

Objective

Learn Git to keep track of your space station locations.

Mission Brief

Hoping to learn the ropes of Bitbucket? Good news cadet! You've just been put in charge of running your team's space stations. To complete your first mission, take advantage of what Git has to offer to track down all station locations.

Time

30 minutes

Audience

You are new to Git and Bitbucket Server

Prerequisites

- You have [installed Git](#)
- You have [installed Bitbucket Server](#)

[Start the mission!](#)

Create a Git repository

Copy your Git repository and add files

Pull changes from your Git repository on Bitbucket Server

Use a Git branch to merge a file

Tutorial: Work with Bitbucket Server

Teams in Space is a fictional company created by Atlassian that specializes in space travel for teams.

Welcome to the Teams in Space web team! You are joining us as a web developer, and your first assignment is to update our company website to include a link to our Moon Itinerary so that our customers know what to expect on their day trip to the Moon.

Here's what you'll accomplish by the end of this tutorial:

1. [Set up SourceTree to work with Bitbucket Server](#)
2. [Create a personal repository in Bitbucket Server](#)
3. [Clone your repository and manage files locally](#)
4. [Commit and push changes to Bitbucket Server](#)

For this tutorial we'll be using SourceTree, a desktop Git client with a graphical interface, to work with Bitbucket Server. If you're already comfortable using Git from the command line we'll also include the Git command equivalent.

Time needed

5-10 minutes

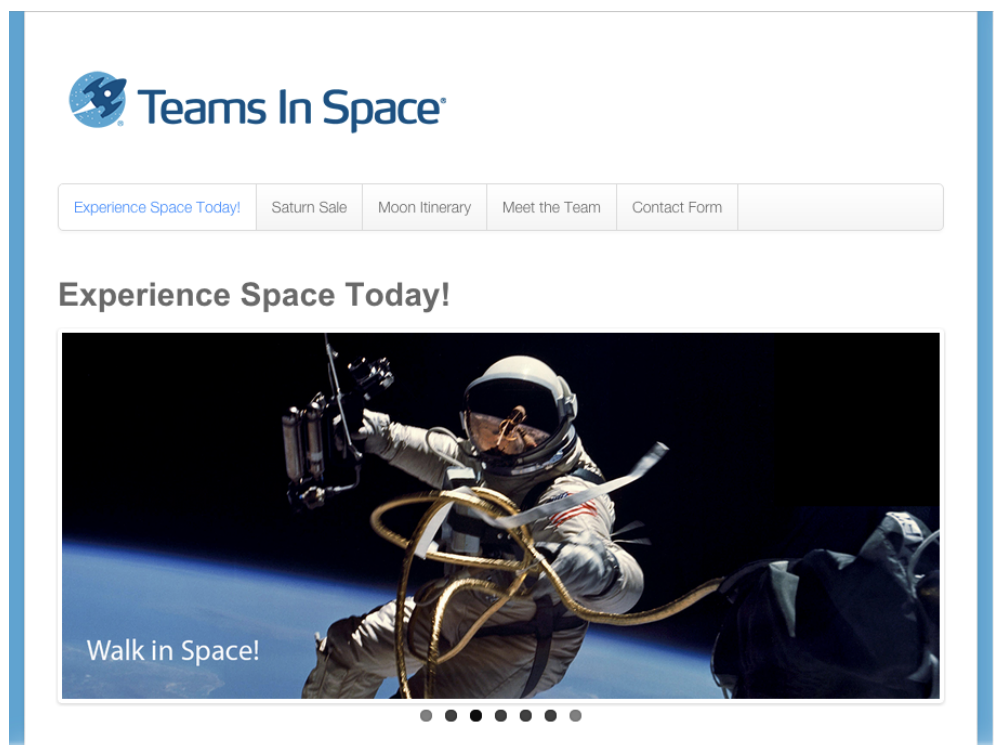
Audience

You're new to working with Bitbucket Server

Prerequisites

- [Bitbucket Server is installed](#)
- [You have login credentials](#)
- [You have a project and repo](#)

Here's what the final version of the HTML page will look like when you're finished (and we've got all the files you need to get this end result).



Let's go!

Set up SourceTree to work with Bitbucket Server

1. Set up SourceTree to work with Bitbucket Server
2. Create a personal repository in Bitbucket Server
3. Clone your repository and manage files locally
4. Commit and push changes to Bitbucket Server

SourceTree provides you with an interface that gives you the same capabilities you have with Git without the need to use the command line. If you prefer to use Git from the command line, feel free to [skip this step](#).

Install SourceTree

1. Click the button for downloading SourceTree from the [SourceTree website](#).
2. Double-click the downloaded file to open it.
3. Install SourceTree as you would any other installation.
4. Open SourceTree and add your Bitbucket Server account credentials and click **Continue**.
5. Click **Skip Setup** from the **Clone your first repo** box (you'll do this from within Bitbucket Server for this tutorial).

Next step

Create a personal repository in Bitbucket Server

1. [Set up SourceTree to work with Bitbucket Server](#)
2. Create a personal repository in Bitbucket Server

3. Clone your repository and manage files locally
4. Commit and push changes to Bitbucket Server

In this step you will create a personal repository in Bitbucket Server to use to keep track of your work for the Teams in Space website.

▼ [About personal repositories...](#)

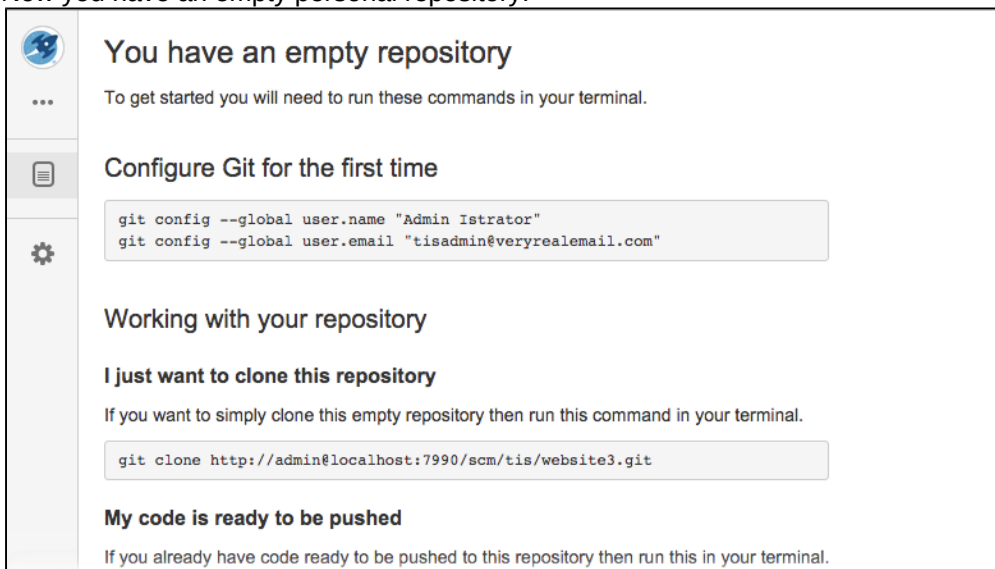
Personal repositories can be used for storing private files or starting your own project and are not visible to other users by default, but you can open access to these repositories whenever you want.

Create a personal repository in Bitbucket Server

1. From within a project, click **Create repository**.



2. Name your repository *Website*, then click **Create**.
Now you have an empty personal repository.



Next step

Clone your repository and manage files locally

1. Set up SourceTree to work with Bitbucket Server
2. Create a personal repository in Bitbucket Server
3. Clone your repository and manage files locally
4. Commit and push changes to Bitbucket Server

In this step you will clone your personal repository to your local computer. Cloning your repository locally

creates a file directory on your computer that will kept in synch with your online repository.

▼ [About cloning...](#)

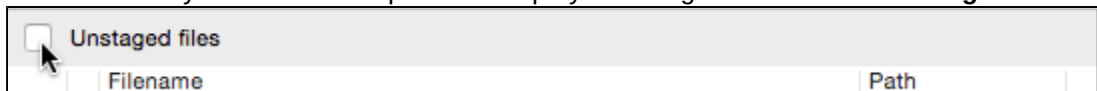
Making changes to live source files makes your website vulnerable to user errors. Since we all make mistakes, we instead clone the source files locally and make our changes on our own computer where we can first test that our changes won't break things in the process. Once we verify things are as they should be we then can push our changes to the live source files (usually a master branch). From there, others can pull in our changes to their local copy, and update files of the website.

1. Clone your personal repository using SourceTree (or the [command line](#))
 - a. On the side navigation, click **Clone**, then **Clone in SourceTree** to create a local directory where you can store the website files.



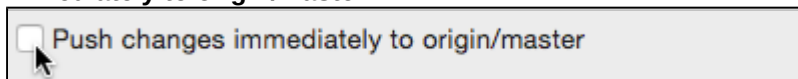
This opens the *Clone New* dialog in SourceTree.

- b. Within SourceTree, choose the appropriate destination for your personal repository, then click **Clone**.
You'll arrive at the empty directory in SourceTree, and an empty directory named *website* was created on your local computer.
2. [Download](#) the source files and unzip them into the empty directory you just created.
3. Add the files to your personal repository using SourceTree (or the [command line](#)).
 - a. Select the files you added in the previous step by checking the box named **Unstaged files**.

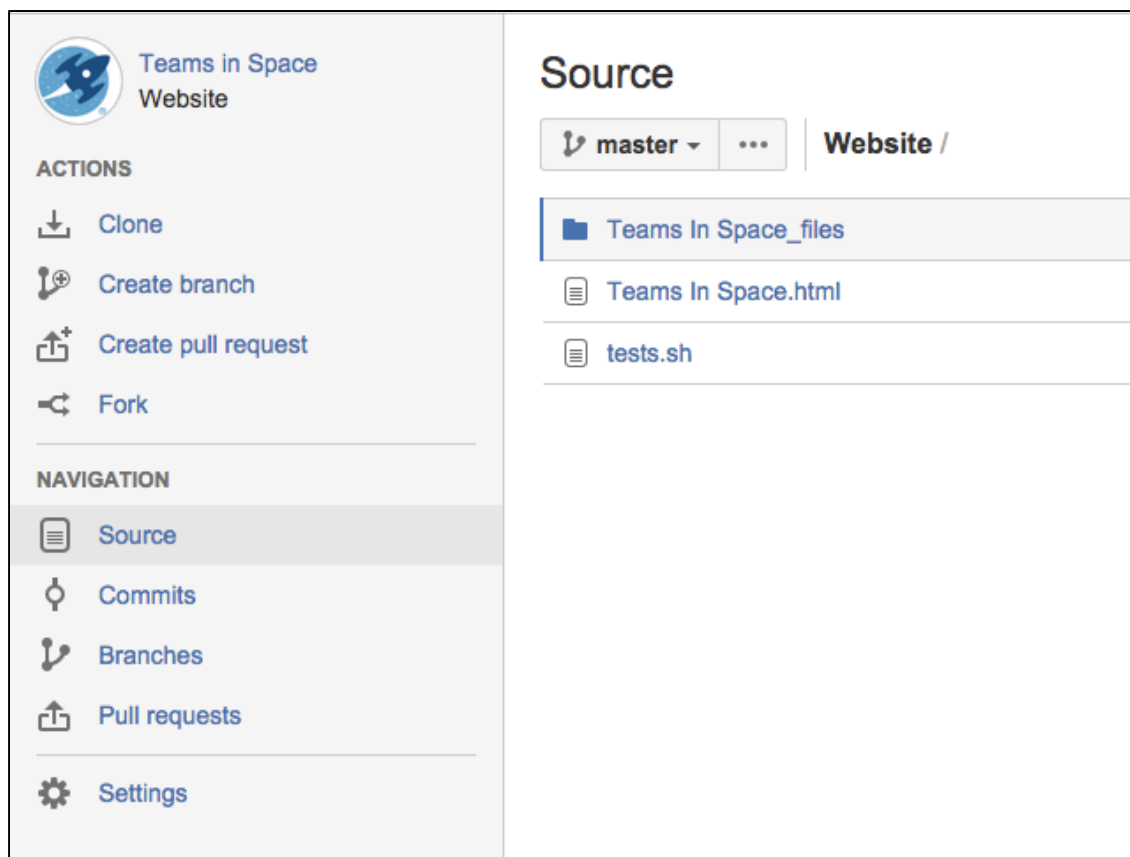


The files then appear in the Staged files pane.

- b. Click **Commit**, add a message in the comment box, and check the box **Push changes immediately to origin/master**.



- c. Go to your personal repository and verify the files were added.


[Next step](#)

Do it from the command line

1. Clone your personal repository from the command line. You can also copy the command directly from your empty repository. Look under **Working with your repository**. From a terminal window, run these commands

```
cd ~
git clone http://<username>@<Bitbucket Server
URL>/scm/<project key>/website.git
```

▼ [Click for an explanation of these commands...](#)

cd ~	Change directory to your home directory
git clone	Command that copies the contents of the repository
<username>	Is the username you use to log in to the instance
<Bitbucket Server URL>	The URL for your Bitbucket Server instance
<project key>	The project key where your personal repository is
website.git	The name of your personal repository

This creates an empty Git repository named *TISwebsite*

2. Add the files to your personal repository from the command line.
From a terminal window

```
cd existing-project
git init
git add --all
git commit -m "Initial Commit"
git remote add origin http://<Bitbucket Server
URL>/scm/tis/website.git
git push -u origin master
```

▼ [Click for an explanation of these commands...](#)

cd existing-project	Change to the directory where you unzipped the files
git init	Initialize the Git repository
git add --all	Adds the files to the repository
git commit -m "Initial Commit"	Adds a comment to the commit
git remote add origin <url> git push -u origin master	Adds the remote repository and pushes your files to the master branch

Next step

Commit and push changes to Bitbucket Server

1. Set up SourceTree to work with Bitbucket Server
2. Create a personal repository in Bitbucket Server
3. Clone your repository and manage files locally
4. Commit and push changes to Bitbucket Server

In this step you are going to make some changes to the HTML files added to your repository in Step 3. Once you make the changes and commit them you can add them to your repository on Bitbucket Server. It's not enough to just make your changes, you have to share them with the world!

Make the changes

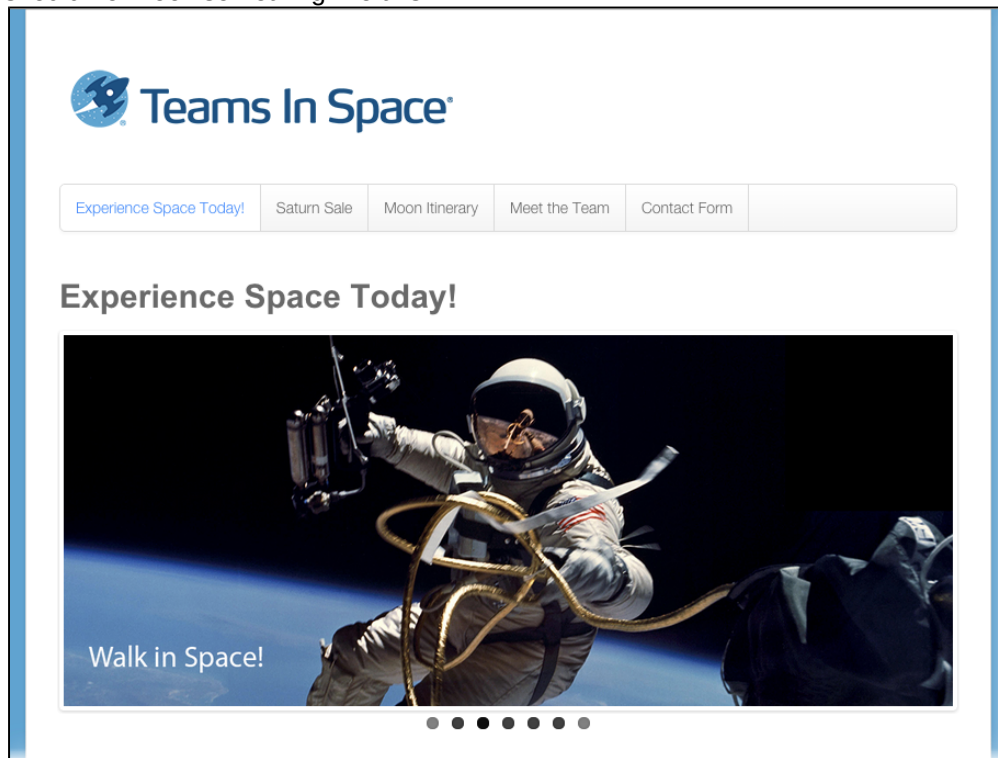
This step explains how to make a simple change in a source file that you'll then commit locally and push to your personal repository.

1. Open the Teams in Space.html file in a text editor.

- Find the `<h3>Main Menu</h3>` line. Within that menu, add another list item to add a link to the itinerary on the Main Menu.
(You can copy this code and add it in your HTML file).

```
<li id="menu-item-65" class="menu-item menu-item-type-post_type
menu-item-object-page menu-item-65"><a
href="http://localhost:2431/?page_id=60">Moon Itinerary</a></li>
```

- Save the file.
- Check your work by going to the `Teams in Space.html` file and opening it in a web browser. It should now look something like this.



Commit and push the changes

Once you've made the changes and verified they work, you'll now commit the changes and push them to your repository.

- Commit the changes using SourceTree (or the command line).
 - From SourceTree, click on **Working Copy** in the upper-left. In the Unstaged files pane on the bottom, you should see the `Teams in Space.html` file.
 - Select the checkbox to left of the file. The file moves to the *Staged files* pane.
 - Click Commit in the upper-left. The Commit dialog opens at the bottom.
 - Enter a commit message in the text field (something like "This is my first commit!" would do).
- Push the changes to the repository.
 - There is now an indicator within the Branches field on the left that there are changes to push, as well as on the Push button on the top toolbar.



- Click the **Push** button, and make sure the master branch is selected, then click **Ok** to push the changes.
- You can verify the changes were pushed by going to the repository and clicking on **Commits**.

Do it from the command line

Commit the files you changed to your personal repository from the command line.
From a terminal window

```
cd website
git commit -m "Website changes"
git push -u origin master
```

Tutorial: Learn SourceTree with Bitbucket Server

Create a new repository

Copy your repository and add files

Pull changes from your repository with SourceTree

Use SourceTree branches to merge an update

Tutorial: Learn about pull requests in Bitbucket Server

Fork a repository

Copy your fork and make a change to the repository

Create a pull request to merge your changes

Using Bitbucket Server

Bitbucket Server is the on-premises Git repository management solution for enterprise teams. It allows everyone in your organization to easily collaborate on your Git repositories.

This section describes the essentials of using Bitbucket Server.

If you are setting up Bitbucket Server, see the [Getting started](#) section. If you want to configure Bitbucket Server, see the [Administering Bitbucket Server](#) section.

See [Getting started with Git and Bitbucket Server](#) for an overview of how to work with Bitbucket Server.

Related pages:

- [Getting started](#)
- [Git Tutorials and Training](#)
- [Git resources](#)
- [Administering Bitbucket Server](#)
- [Bitbucket Server FAQ](#)

Working with projects

Bitbucket Server manages related repositories as projects. Find out how to [set up projects](#) and then [give your teams access](#) to those.

Working with repositories

If you have existing projects that you want to manage in Bitbucket Server, then you'll want to read [Importing code from an existing project](#).

See also:

- [Creating repositories](#)
- [Controlling access to code](#)
- [Using pull requests in Bitbucket Server](#)

Git resources

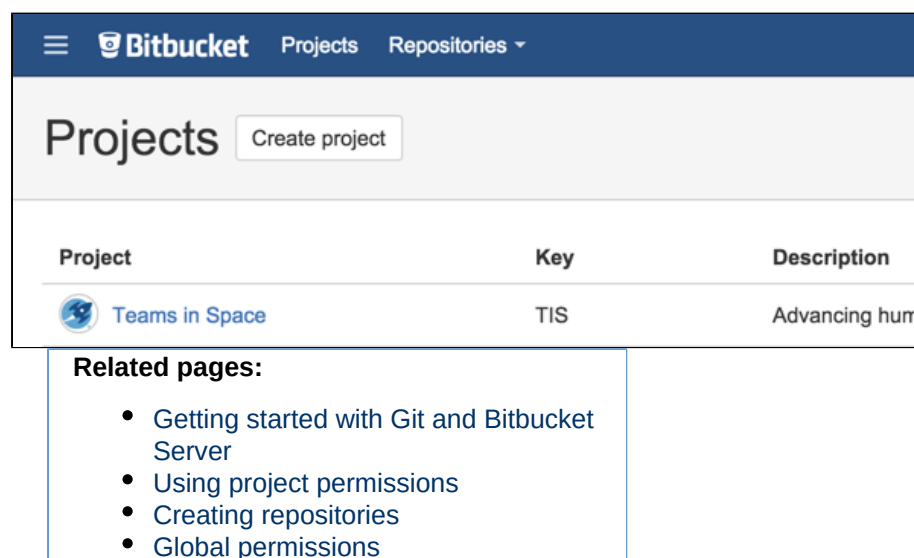
For those who are new to using Git:

- [Using pull requests in Bitbucket Server](#)
- [Basic Git commands](#)
- [Permanently authenticating with Git repositories](#)

Creating projects

Projects allow you to group repositories and to [manage permissions](#) for them in an aggregated way.

To create a project, click on **Create project**:



Fill out the form. We recommend that you use a short project key. It will be used as an identifier for your project and will appear in the URLs.

Optionally, you can choose an avatar for the project. This is displayed throughout Bitbucket Server and helps to identify your project.

Click **Create project** when you're done.

Create a Project

Project name *

Teams in Space

Project key *

TIS

Eg. AT (for a project named Atlassian)

Description

Advancing humanity into the cosmos!

Project Avatar



Change avatar

Create project

Cancel

You'll want to add repositories to the project. See [Creating repositories](#) for details.

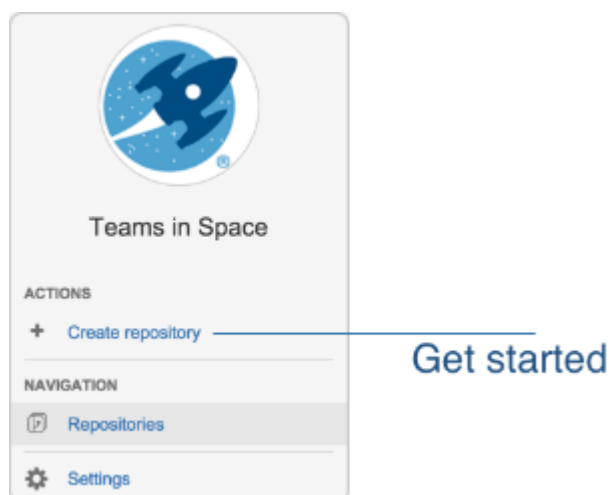
Creating repositories

Repositories allow you to collaborate on code with your co-workers.

In order to create repositories you need to have [Project Admin permission](#) for the project to which you want to add a repository.

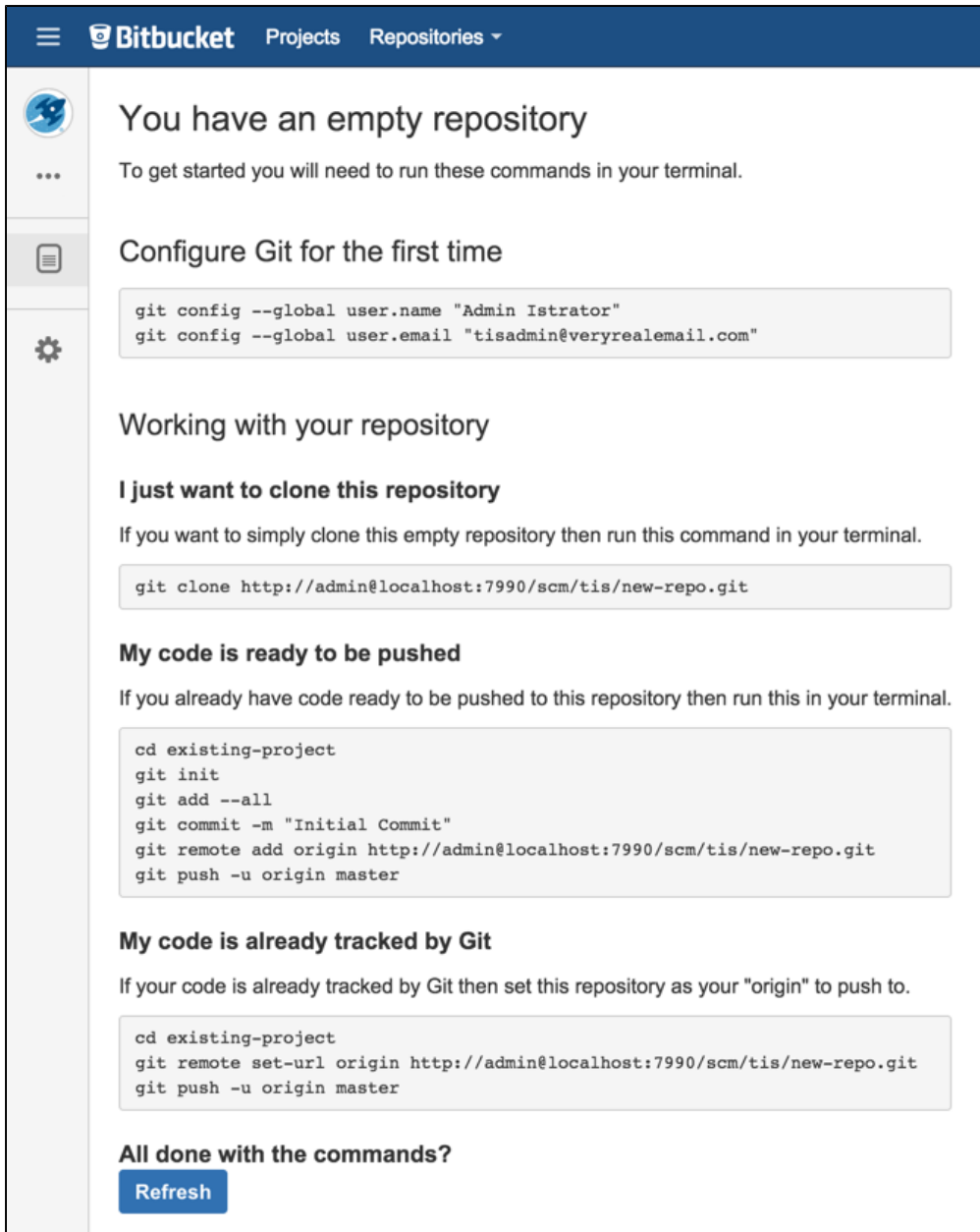
When a repository is created, the project permissions are applied to the repository. That means all repositories created in a project share the same access and permission settings.

Go to the project and click **Create repository** to open the repository creation form:



Once submitted you will be taken directly to your repository homepage.

There won't be any content in your repository yet, so you'll see some instructions to help you push code to your repository:



You have an empty repository

To get started you will need to run these commands in your terminal.

Configure Git for the first time

```
git config --global user.name "Admin Istrator"
git config --global user.email "tisadmin@veryrealemail.com"
```

Working with your repository

I just want to clone this repository

If you want to simply clone this empty repository then run this command in your terminal.

```
git clone http://admin@localhost:7990/scm/tis/new-repo.git
```

My code is ready to be pushed

If you already have code ready to be pushed to this repository then run this in your terminal.

```
cd existing-project
git init
git add --all
git commit -m "Initial Commit"
git remote add origin http://admin@localhost:7990/scm/tis/new-repo.git
git push -u origin master
```

My code is already tracked by Git

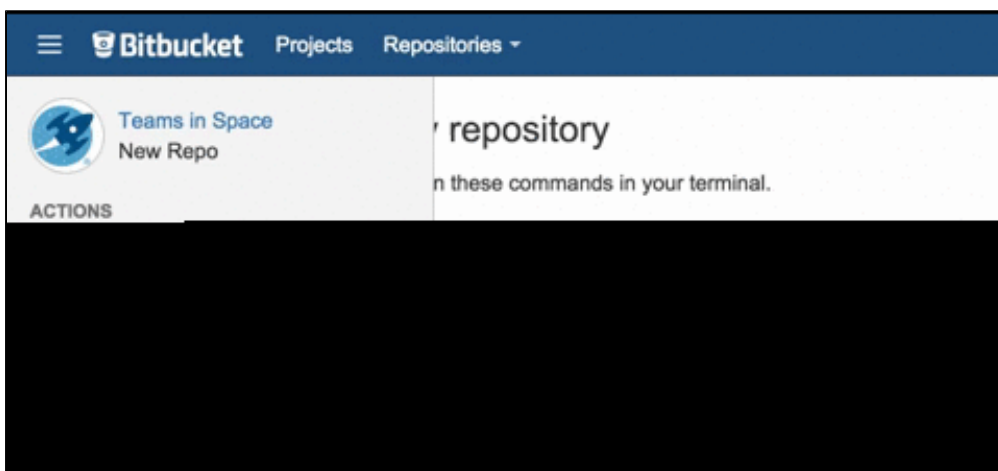
If your code is already tracked by Git then set this repository as your "origin" to push to.

```
cd existing-project
git remote set-url origin http://admin@localhost:7990/scm/tis/new-repo.git
git push -u origin master
```

All done with the commands?

[Refresh](#)

You will find your clone URL in the lefthand sidebar of the repository homepage. You can use this URL and share it with other people.



Teams in Space
New Repo

ACTIONS

repository

in these commands in your terminal.

Let other people collaborate with you

In order to grant users access to this repository you have to set up permissions at the parent project level. More information is available on [Creating projects](#).

Creating personal repositories

Bitbucket Server allows you to create personal repositories, unrelated to other projects, that you can use for such purposes as storing private snippets of work, kick-starting your own project, or contributing a bug-fix for a project you are not a member of.

By default, personal repositories are not visible to other Bitbucket Server users (unless they are a Bitbucket Server [system administrator](#)). However, you can:

- use [repository permissions](#) to open up access to other Bitbucket Server users and groups, for collaboration or review.
- allow [public access](#) (read-only) to your project, for anonymous users.

You can create personal repositories in 2 ways:

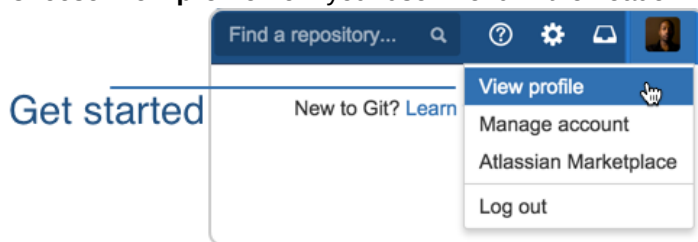
- [Directly](#), from your profile.
- By [forking](#) another repository.

Your personal repositories are listed on the **Repositories** tab of your profile page. Every Bitbucket Server user can see your profile page, but they can only see those repositories that you have given them permission to view.

Directly creating a personal repository

You can create a personal repository at any time from your Bitbucket Server profile:

1. Choose **View profile** from your user menu in the header.

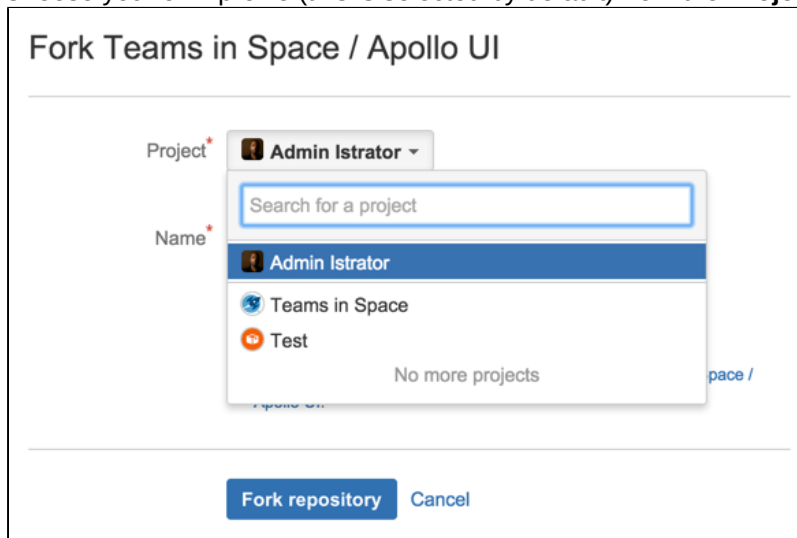


2. Click **Create repository**.
3. Set [repository permissions](#) on the new repository, if required.

Forking another repository

You can create a personal fork of any other repository in Bitbucket Server for which you have [permission](#):

1. Go to the repository that you wish to fork.
2. Click **Fork** in the sidebar.
3. Choose your own profile (this is selected by default) from the **Project** list:



4. Click **Fork repository**.
5. Set [repository permissions](#) on the new repository, if required.

Using repository hooks

Hooks in Bitbucket Server provide a way to customize a team's workflow and integrate with other systems. Bitbucket Server currently supports two types of hooks, pre and post-receive.

On this page

- [Managing hooks](#)
- [Getting hooks from the Atlassian Marketplace](#)
- [Creating your own hooks](#)

Managing hooks

Administrators can see the hooks that are available in Bitbucket Server by going to **Settings > Hooks** for a Bitbucket Server repository. Once installed, hooks are available across all repositories in a Bitbucket Server instance, but are enabled separately on each repository in a project.

Settings

Repository details

- Hooks**
- Pull requests
- Branching model
- Audit log
- Access keys

PERMISSIONS

- Repository
- Branch

Hooks [Add hook](#) [Learn more](#)

Hooks allow you to extend what Stash does every time the repository changes (for example, when new code is pushed or when a pull request is merged). Hooks are installed by the system administrator and can be enabled by project administrators on a per-repository basis.

Pre receive - reject commits that don't match your policies

Reject Force Push [Add](#) [Edit](#) [Delete](#) [Learn more](#)

Reject all force pushes (git push --force) to this repository

Disabled Enabled

Post receive - perform actions after commits are processed

HipChat Push Notification [Add](#) [Edit](#) [Delete](#) [Learn more](#)

Sends a notice to the specified HipChat room whenever someone pushes to the repository.

Disabled Enabled

Click the 'pen' icon beside the name of a hook to edit configuration details for the hook.

HipChat Push Notification

Room Name*

Name of the HipChat room (or ID)

API Token*

Your HipChat API token

Specify the HipChat room to receive notifications about pushes and merges.

You can get a list of available rooms from your HipChat client or through the [rooms page](#).

You must specify an [API token](#) (only a user token is required).

Save Disable Cancel

Bitbucket Server currently ships with the following hooks:

- **Reject Force Push** – block all Git force pushes (git push --force).

- [HipChat Push Notifications](#) – send a message to a HipChat room when someone pushes to the repository.

Pre-receive hooks

The first hook to run when handling a push from a client is the pre-receive hook. It can reject pushes to the repository if certain conditions are not fulfilled. You can use this hook to prevent force pushes to the repository or check whether all commits contain a valid JIRA application issue key.

Post-receive hooks

The post-receive hook runs after the commits have been processed and can be used to update other services or notify users. For example [this post-receive hook](#) could be used to send a message to a chat server or notify a continuous integration server such as [Atlassian Bamboo](#) of the newly pushed changes.

Bitbucket Server supports two types of post-receive hook:

- [PostReceiveHooks](#) map to Git's post - receive hooks. They run on the Bitbucket Server instance after a push.
- [AsyncPostReceiveRepositoryHooks](#), executed by the Bitbucket Server instance.

Note that a Git PostReceiveHook won't be triggered after a [pull request](#) merge. The mechanism that performs the pull request merge is actually based on a `git fetch into` the repository, which doesn't trigger Git post - receive hooks. To trigger functionality based on a pull request merge, you should write an [AsyncPostReceiveRepositoryHook](#) for the Bitbucket Server repository.

Getting hooks from the Atlassian Marketplace

A number of hooks are available from the [Atlassian Marketplace](#). You can find and install these from within Bitbucket Server – simply use the **Add hook** button on the hooks settings page to view available hooks from the marketplace. See [Managing add-ons](#) for details.

Creating your own hooks

Developers can write receive hook plugins for Bitbucket Server using a simple API that provides a simple way to create a configuration interface, and stores the hook's configuration settings on a per-repository basis.

For information about how to write your own hooks please see the [Bitbucket Server developer docs](#).

In particular, these pages will be helpful:

- [Repository hooks](#)
- [Repository hook plugin module](#)

See too this blog post about hooks for Bitbucket Server: <http://blogs.atlassian.com/2013/03/stash-git-hooks-a-pi/>

For a quick video demo on how to get started on Bitbucket Server hooks: <https://developer.atlassian.com/blog/2015/01/beer-o-clock-stash-plugin-tutorial/>

Permanently authenticating with Git repositories

In addition to SSH, Bitbucket Server supports HTTP or HTTPS for pushing and pulling from managed Git repositories. However, Git does not cache the user's credentials by default, so you need to re-enter them each time you perform a clone, push or pull.

This page describes two methods for permanently authenticating with Git repositories so that you can avoid typing your username and password each time you are pushing to or pulling from Bitbucket Server.

On this page:

- [Using credential caching](#)
- [Using the .netrc file](#)

Related pages:

- [Getting started with Git and Bitbucket Server](#)
- [Creating repositories](#)
- [Global permissions](#)
- [Git resources](#)

Using credential caching

You need Git 1.7.9 or above to use the HTTPS Credentials Caching feature.

Windows

On Windows you can use the application [git-credential-winstore](#).

1. [Download the software](#).
2. Run it.
3. You will be prompted for credentials the first time you access a repository, and Windows will store your credentials for use in the future.

Linux

On Linux you can use the 'cache' authentication helper that is bundled with Git 1.7.9 and higher. From the Git documentation:

This command caches credentials in memory for use by future git programs. The stored credentials never touch the disk, and are forgotten after a configurable timeout. The cache is accessible over a Unix domain socket, restricted to the current user by filesystem permissions.

Run the command below to enable credential caching. After enabling credential caching any time you enter your password it will be cached for 1 hour (3600 seconds):

```
git config --global credential.helper 'cache --timeout 3600'
```

Run the command below for an overview of all configuration options for the 'cache' authentication helper:

```
git help credential-cache
```

OS X

Follow these steps to use Git with credential caching on OS X:

1. Download the binary [git-credential-osxkeychain](#).
2. Run the command below to ensure the binary is executable:

```
chmod a+x git-credential-osxkeychain
```

3. Put it in the directory `/usr/local/bin`.
4. Run the command below:

```
git config --global credential.helper osxkeychain
```

Using the .netrc file

The `.netrc` file is a mechanism that allows you to specify which credentials to use for which server. This method allows you to avoid entering a username and password every time you push to or pull from Git, but your Git password is stored in plain text.

Warning!

- Git uses a utility called `cURL` under the covers, which respects the use of the `.netrc` file. Be aware that other applications that use `cURL` to make requests to servers defined in your `.netrc` file will also now be authenticated using these credentials. Also, this method of authentication is potentially unsuitable if you are accessing your Bitbucket Server instance via a proxy, as all `cURL` requests that target a path on that proxy server will be authenticated using your `.netrc` credentials.
- `cURL` will not match the machine name in your `.netrc` if it has a username in it, so make sure you edit your `.git/config` file in the root of your clone of the repository and remove the user and '@' part from any clone URL's (URL fields) that look like `https://user@machine.domain.com/...` to make them look like `http://machine.domain.com/...`

Windows

1. Create a text file called `_netrc` in your home directory (e.g. `c:\users\kannonboy\_netrc`). `cURL` has problems resolving your home directory if it contains spaces in its path (e.g. `c:\Documents and Settings\kannonboy`). However, you can update your `%HOME%` environment variable to point to any directory, so create your `_netrc` in a directory with no spaces in it (for example `c:\curl-auth\`) then set your `%HOME%` environment variable to point to the newly created directory.
2. Add credentials to the file for the server or servers you want to store credentials for, using the format described below:

```
machine stash1.mycompany.com
login myusername
password mypassword
machine stash2.mycompany.com
login myotherusername
password myotherpassword
```

Linux or OS X

1. Create a file called `.netrc` in your home directory (`~/ .netrc`). Unfortunately, the syntax requires you to store your passwords in plain text - so make sure you modify the file permissions to make it readable only to you.
2. Add credentials to the file for the server or servers you want to store credentials for, using the format described in the 'Windows' section above. You may use either IP addresses or hostnames, and you do *not* need to specify a port number, even if you're running Bitbucket Server on a non-standard port.
3. And that's it! Subsequent `git clone`, `git pull` and `git push` requests will now be authenticated using the credentials specified in this file.

Clone a repository**Cloning a repository**

You can use Sourcetree, Git from the terminal, or any client you like to clone your Git repository. These instructions show you how to clone your repository using Git from the terminal.

1. Navigate to the repository in Bitbucket.
2. Click the **Clone** button.
3. Copy the clone command (either the SSH format or the HTTPS).
If you are using the SSH protocol, ensure your public key is in Bitbucket and loaded on the local system to which you are cloning.
4. Launch a terminal window.
5. Change to the local directory where you want to clone your repository.
6. Paste the command you copied from Bitbucket, for example:

```
$ git clone ssh://git@bitbucket.example.com:7999/PROJ/repo.git
```

If the clone was successful, a new sub-directory appears on your local drive. This directory has the same name as the Bitbucket repository that you cloned. The clone contains the files and metadata that Git requires to maintain the changes you make to the source files.

Cloning a mirror repository

You can use Sourcetree, Git from the terminal, or any client you like to clone your Git repository. These instructions show you how to clone a mirrored repository using Git from the terminal. Read more about [Smart Mirrors](#).

1. Navigate to the repository in Bitbucket.
2. Click the **Clone** button.
3. If [mirrors are configured](#), use the **Clone from** dropdown to select the mirror closest to you—the clone URL changes.
4. Copy the clone URL (either SSH or HTTPS).
If you are using the SSH protocol, ensure your public key is correctly configured.
5. Launch a terminal window.
6. Change to the local directory where you want to clone your repository.
7. Enter `git clone` followed by the copied clone URL.
The command and clone URL together would look like this:

```
$ git clone  
ssh://git@bitbucket-au.example.com:7999/upstream/PROJ/repo.git
```

If the clone was successful, a new sub-directory appears on your local drive. This directory has the same name as the Bitbucket repository that you cloned. The clone contains the files and metadata that Git requires to maintain the changes you make to the source files.

You cannot push to a mirror!

Mirrors are read-only. After cloning from a mirror, you must [update your remote push URL](#) to point to the primary Bitbucket Data Center instance.

Update your remote push URL

Since you cannot push to a mirror, after cloning from a mirror you need to update your remote push URL to point to the primary (upstream) instance.

To update your push URL

1. In the repository on Bitbucket Server, click the **Clone** button.
2. Select a mirror in the **Clone from** dropdown, then copy the command that looks like this:

```
git remote set-url --push origin  
ssh://git@bitbucket.example.com:7999/PROJ/repo.git
```

3. In the terminal, navigate to the cloned directory then run the command.

Your remote push URL now points to the primary Bitbucket Data Center instance.

Update your remote push URL in SourceTree

Since you cannot push to a mirror, after cloning from a mirror you need to update your remote push URL to

point to the primary (upstream) instance within SourceTree.

To update your push URL in SourceTree

1. In the repository on Bitbucket Server, click **Clone**.
2. Select a mirror in the **Clone from** dropdown, then copy the command that looks like this:

```
git remote set-url --push origin  
ssh://git@bitbucket.example.com:7999/PROJ/repo.git
```

You won't need the commands, only the push URL (the bit that starts after `ssh://`).

3. Within SourceTree, open your repository settings.
4. Select the origin repository path, then click **Edit Config File...**
A text editor appears.
5. Within the Config File, under `[remote "origin"]`, enter `pushurl =` then paste *only the push URL* after `=`.

The config file should look like this:

```
pushurl = ssh://git@bitbucket.example.com:7999/PROJ/repo.git
```

6. Save and close the Config File. Click **OK** to close the Repository Settings screen.

Your remote push URL within SourceTree now points to the primary Bitbucket Data Center instance.

Controlling access to code

Bitbucket Server provides the following types of permissions to allow fully customizable control of access to code.

Note that you can also:

- allow public (anonymous) access to projects and repositories. See [Allowing public access to code](#).
- use SSH keys to allow user accounts and other systems to connect securely to Bitbucket Server repositories for Git operations. See [Using SSH keys to secure Git operations](#).

Global permissions

- Control user and group access to Bitbucket Server projects and to the Bitbucket Server instance configuration.
- For example, these can be used to control the number of user accounts that can access Bitbucket Server for licensing purposes.
- See [Global permissions](#).

Project permissions

- Apply the same access permissions to all repositories in a project.
- For example, these can be used to define the core development team for a project.
- See [Using project permissions](#).

Repository permissions

- Extend access to a particular repository for other, non-core, users.
- For example, these can be used to allow external developers or consultants access to a repository for special tasks or responsibilities.
- See [Using repository permissions](#).

Branch permissions

- Control commits to specific branches within a repository.
- For example, these can provide a way to enforce workflow roles such as the Release Manager, who needs to control merges to the release branch.
- See [Using branch permissions](#).

Permissions matrix

The table below summarizes the cumulative effect of the permissions described above for anonymous and logged in users. In general, repository permissions override project permissions. A [personal project](#) can not be made public.

Key

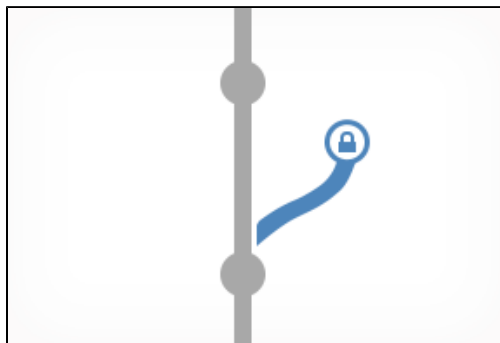
Permission	Effect
BROWSE	Can view repository files, clone, pull to local
READ	Can browse, clone, pull, create pull requests, fork to a personal project
WRITE	Can merge pull requests
ADMIN	Can edit settings and permissions

Global (logged in)	Project	Repository	Branch	Effective permission
✗	Personal	Personal	NA	No access
✗	Personal	Public access	NA	BROWSE just that repo
✗	No access	No access	NA	No access
✗	No access	Public access	NA	BROWSE just that repo
✗	Public access	Public access	NA	BROWSE all repos in project
✓	Personal	Personal	NA	No access
✓	Personal	Public access	NA	READ just that repo
✓	No access	No access	NA	No access
✓	No access	Public access	NA	READ just that repo
✓	Public access	No access	NA	READ all repos in project
✓	Public access	Public access	NA	READ
✓	Public access	Public access	For this user	READ that branch, no WRITE
✓	No access	READ	NA	READ just that repo
✓	Public access	READ	NA	READ just that repo
✓	READ	No access	NA	READ all repos in project
✓	READ	Public access	NA	READ all repos in project
✓	READ	READ	NA	READ all repos in project
✓	READ	No access	For this user	READ that branch, no WRITE
✓	No access	WRITE	NA	WRITE just that repo
✓	Public access	WRITE	NA	WRITE just that repo
✓	WRITE	No access	NA	WRITE all repos in project
✓	WRITE	WRITE	NA	WRITE all repos in project

✓	WRITE	WRITE	For other users	WRITE to other branches only
✓	ADMIN			Can edit settings and permissions

Using branch permissions

Branch permissions allow you to control the actions users can perform on a single branch, branch type, or branch pattern within a repository. Branch permissions provide another level of security within Bitbucket Server, along with [user authentication](#) and [project, repository](#) and [global permissions](#), that together allow you to control, or enforce, your own workflow or process.



Branch permissions:

- are based on users or groups.
- are actually restrictions, which are checked after project and repository level permissions.
- are used to limit branch access to specific people who must still have write access to the project or repository.
- prevent unauthorized users pushing to or deleting the branch.
- can be based on explicit branch names, branch pattern, or branching model.

For example, if two developers Xavier and Yves have write access to repository R, but only Xavier has branch permissions on branch B, then Yves won't be able to push to B.

If a user does not have commit access to the branch, an error message will be shown on the Git command line when they try to push a change to the branch. If no branch permissions are defined then anyone with commit access to the repository can push to any branch.

Adding branch permissions

Branch permissions in Bitbucket Server control access to repository branches. You need either project admin, admin or sys-admin [permission](#) to set or modify branch permissions.

To add branch permissions:

1. Go to a repository in a project.
2. Choose **Settings > Branch permissions**.
3. Click **Add permission**.

Add a branch permission

Branches

Branch name

Select branch

Select the branch you want to restrict access to.

Restrictions

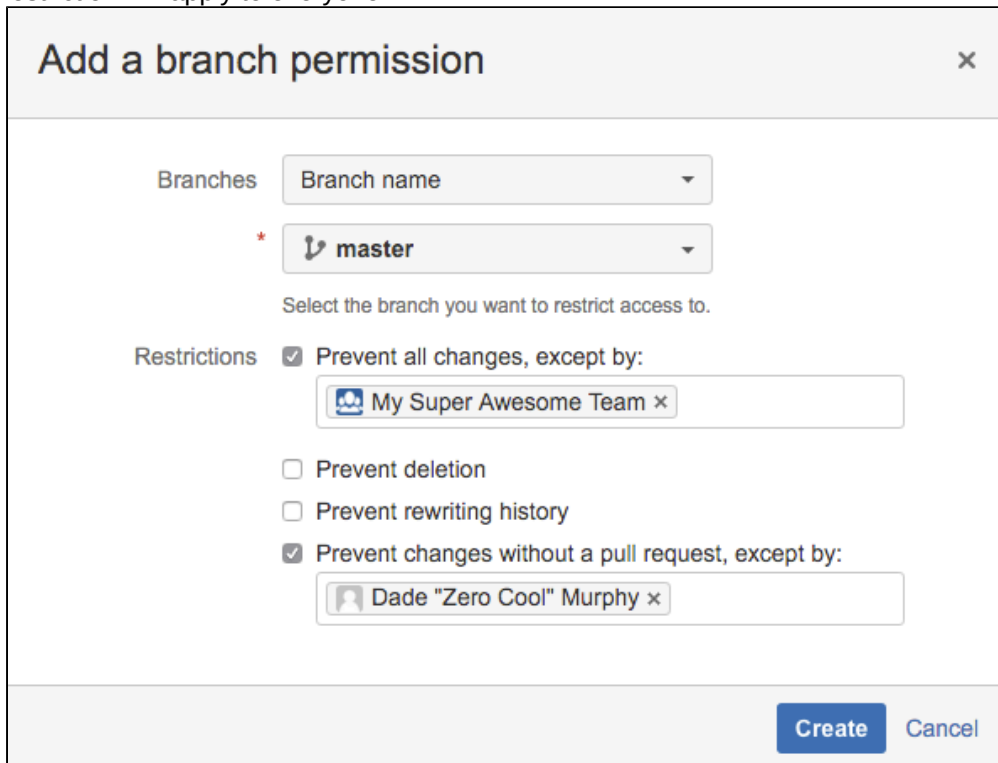
☐ Prevent all changes
 ☐ Prevent deletion
 ☐ Prevent rewriting history
 ☐ Prevent changes without a pull request

Create

Cancel

4. In the Branches field, select either **Branch name**, **Branch pattern**, or **Branching model**.
 - a. Branch name - select an existing branch by name.

- b. Branch pattern - specify a branch using branch pattern syntax for matching branch names. See [Branch permission patterns](#) for more information about this syntax.
 - c. Branching model - select the branch type to restrict access to. Read more about [branching models](#).
5. Select the type of actions you want to prevent.
 - a. Branch deletion - prevents branch and tag deletion. See [Branch permission patterns](#) for information about specifying tags.
 - b. Rewriting history - prevents history rewrites on the specified branch(es) - for example by a force push or rebase.
 - c. Changes without a pull request - prevents pushing changes directly to the specified branch(es); changes are allowed only with a pull request.
 - d. All modifications - prevents pushes to the specified branch(es) and restricts creating new branches matching the specified branch(es) or pattern.
6. Optional: Add exemptions for any of the selected restrictions. Adding a user or group as an exemption means that it will not apply to them. *This is not required*; not adding any exemptions means the restriction will apply to everyone.



7. Click **Create** to finish.

You can always change the permissions for a branch later, if necessary.

Branch permission patterns

Bitbucket Server supports a powerful type of pattern syntax for matching branch names (similar to pattern matching in Apache Ant).

These expressions use the following wild cards:

?	Matches one character (any character except path separators)
*	Matches zero or more characters (not including path separators)
**	Matches zero or more <i>path segments</i> .

Pattern used in branch permissions match against all refs pushed to Bitbucket Server (i.e. branches and tags).

In git, branch and tag names can be nested in a namespace by using directory syntax within your branch names, e.g. `stable/1.1`. The `***` wild card selector enables you to match arbitrary directories.

- A pattern can contain any number of wild cards.

- If the pattern ends with / then ** is automatically appended - e.g. foo/ will match any branches or tags containing a foo path segment
- Patterns only need to match a suffix of the fully qualified branch or tag name. Fully qualified branch names look like refs/heads/master, while fully qualified tags look like refs/tags/1.1.

Also see the [Ant documentation](#).

Examples

*	Matches everything
PROJECT-*	Matches and branch or tag named PROJECT-*, even in a name space. e.g. refs/heads/PROJECT-1234, refs/heads/stable/PROJECT-new or refs/tags/PROJECT-1.1
?.?	Matches any branch or tag of 2 characters separated by a '.'. e.g. refs/heads/1.1, refs/heads/stable/2.X or refs/tags/3.1
tags/ or tags/**	Matches all tags and any branches with 'tags' as a namespace. e.g. refs/heads/stable/tags/some_branch, refs/tags/project-1.1.0
heads/**/master	Matches all branches called master. e.g. refs/heads/master, refs/heads/stable/master

Using repository permissions

Bitbucket Server allows you to manage the permissions for just a single repository, or for a group of repositories together from the [project](#).

Repository permissions allow you to extend access to a repository, for those who don't have project permissions. For example, you might use repository permissions to allow external developers or consultants access to a repository for special tasks or responsibilities.

Bitbucket Server supports 3 levels of permissions for repositories:

- Admin
- Write
- Read

Depending on the permission level for the repository that has been granted to you, you can perform different actions in the repository:

Related pages:

- [Using project permissions](#)
- [Using branch permissions](#)
- [Global permissions](#)
- [Allowing public access to code](#)

	Browse	Clone, fork, pull	Create, browse or comment on a pull request	Merge a pull request	Push	Edit settings and permissions
Admin	✓	✓	✓	✓	✓	✓
Write	✓	✓	✓	✓	✓	✗
Read	✓	✓	✓	✗	✗	✗

Note that:

- Anyone with permission to browse a pull request can create a task on any comment, and can browse,

- resolve and reopen existing tasks in the pull request.
- Repository admins and pull request authors can edit and delete *any* task in the pull request. Reviewers and others can only edit or delete their *own* tasks.

Granting access to a repository

To modify its permissions, go to the repository's settings and click on **Repository** (under 'Permissions'). Click in the **Add Users** or **Add Groups** field in the relevant section to search for, and bulk add, users or groups. Now choose a permission from the list and click **Add**.

Once added, you can use the checkboxes to edit specific permissions for an individual user or a particular group.

Granting access to all repositories within a project

If you have a large number of repositories in a project, [project level permissions](#) provide a convenient way to grant access to *all* repositories within that project. For example you can grant a group, say "Team A", *Write* access at the project level, which will automatically give them *Write* access to all existing repositories in the project, as well as any repositories that are subsequently created in the project.

To modify permissions for a project, click the **Permissions** tab when viewing the project. You can add, or modify, permissions for individual users, and groups, in the same way as described above for a single repository.

Granting permission to create repositories

Only users with [project administration](#) permission can create *new repositories*.

Using project permissions

Bitbucket Server allows you to manage the permissions for the repositories in a project in an aggregated way.

There are 3 levels of project permission that you can assign to a user or group for a project: *Admin*, *Write* and *Read*.

Related pages:

- [Creating projects](#)
- [Global permissions](#)
- [Using branch permissions](#)
- [Using repository permissions](#)
- [Allowing public access to code](#)

	Browse	Clone / Pull	Create, browse, comment on pull request	Merge pull request	Push	Create repositories	Edit settings / permissions
Project Admin	✓	✓	✓	✓	✓	✓	✓
Write	✓	✓	✓	✓	✓	✗	✗
Read	✓	✓	✓	✗	✗	✗	✗

To modify permissions for a project, go to **Settings > Permissions** for the project. Click in the **Add Users** or **Add Groups** fields in the relevant section to search for, and bulk add, users or groups. Now choose a permission from the drop-down list, and click **Add**.

Once added, you can use the checkboxes to edit specific permissions for particular users or groups.

Project permissions

Public Access

Allow users without a Bitbucket account to clone and browse repositories within this project.

☐ Enable

Default Permission

Specify the default level of access to this project for logged-in users for specific user and group access below.

- ☐ Write
☐ Read
☒ No access

User access

Name	Admin ?	Write ?	Read ?
Add Users			Rea
 Admin Istrator	☒	☒	☒

Group access

Name	Admin ?	Write ?	Read ?
Add Groups			Rea
 confluence-users	☐	☐	☒

Allowing public access to code

You can open up public access for anonymous (unauthenticated) users to projects and repositories in Bitbucket Server. This allows you to:

- Broadcast your repositories to a wider audience who generally don't have access to your source.
- Utilize unauthenticated cloning of repositories when setting up continuous integration servers to work with Bitbucket Server.
- Link from other systems, for example JIRA applications or Confluence, to give users access to code without requiring authentication.
- Create open-source projects or repositories.

On this page:

- [Making a repository publicly accessible](#)
- [Making a project publicly accessible](#)
- [Viewing public repositories](#)
- [Disabling public access globally](#)

Related pages:

- [Using project permissions](#)
- [Using repository permissions](#)

Public access allows anonymous users to browse the files, pull requests and commits for a specific repository or an entire project, and to clone repositories, without needing to log in, or have an account in Bitbucket Server.

In Bitbucket Server, you can:

- Configure a specific repository for public access.
- Configure a project to allow public access to all repositories in the project.
- Disable anonymous access by setting a global system property.

Making a repository publicly accessible

You can open up a specific repository for public (anonymous) access.

You need admin permission for the repository.

Go to the repository and click **Settings**, then **Repository** (under 'Permissions'). Check **Enable** (under 'Public Access') to allow users without a Bitbucket Server account to clone and browse the repository.

Making a project publicly accessible

You can open up a whole project (but not a private project) for public (anonymous) access.

You need admin permission for the project.

Go to the project and choose **Settings**, then **Permissions**. Check **Enable** (under 'Public Access') to allow users without a Bitbucket Server account to clone and browse any repository in the project.

Viewing public repositories

Bitbucket Server displays a list of repositories for which anonymous access has been enabled.

Anonymous and logged-in users can choose **Repositories** > **View all public repositories** to see these.

Disabling public access globally

Bitbucket Server provides a [system property](#) that allows you to turn off public access for the whole instance.

To do this, set the `feature.public.access` property to `false` in the `bitbucket.properties` file in your [Bitbucket Server home directory](#).

Using SSH keys to secure Git operations

Bitbucket Server provides a simple way for user accounts and other systems to connect securely to Bitbucket Server repositories, using SSH keys, in order to perform Git operations. You can:

- add a personal key to a Bitbucket Server user account to allow a developer to easily authenticate when performing read operations from his or her local machine. A Bitbucket Server user can add any number of keys to their account. Read more at [SSH user keys for personal use](#).
- add an access key to a Bitbucket Server project or repository to allow other systems, such as build servers like Atlassian's [Bamboo](#), to authenticate for either read-only (pull, clone) or read-write (push, merge) operations, without the need to store user credentials. Read more at [SSH access keys for system use](#).

Related pages:

- [Creating SSH keys](#)
- [Enabling SSH access to Git repositories in Bitbucket Server](#)
- [Permanently authenticating with Git repositories](#)

Before you can use SSH keys to secure a connection with Bitbucket Server the following must have already been done:

- your Bitbucket Server administrator must have already [enabled SSH access](#) in Bitbucket Server.
- you need an SSH key! See [Creating SSH keys](#). Alternatively, you can use an existing key, if it isn't already being used as a repository or project access key in Bitbucket Server.

Note that:

- You can use the same SSH system access key for multiple repositories or projects.
- A Bitbucket Server user can add any number of keys to their account.
- Keys used for personal user accounts can't be re-used as a project or repository access key, and keys used as a project or repository access key can't be re-used for user accounts.
- Bitbucket Server supports DSA and RSA2 key types – RSA1 is not supported.

Creating SSH keys

This page describes how to create SSH keys.

SSH keys can be used to establish a secure connection with Bitbucket Server for:

- when you are performing Git operations from your local machine
- when another system or process needs access to repositories in Bitbucket Server (for example your build server)

The SSH key needs to be added to Bitbucket Server, and your Bitbucket Server administrator must have [enabled SSH access](#) to Git repositories, before you can make use of the key.

Supported key types are DSA and RSA2 – RSA1 is not supported.

You can use an existing SSH key with Bitbucket Server if you want, in which case you can go straight to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).

On this page:

Related pages:

- [Using SSH keys to secure Git operations](#)
- [Enabling SSH access to Git repositories in Bitbucket Server](#)
- [Permanently authenticating with Git repositories](#)

Creating an SSH key on Windows

1. Check for existing SSH keys

You should check for existing SSH keys on your local computer. *You can use an existing SSH key with Bitbucket Server if you want, in which case you can go straight to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).*

Open a command prompt, and run:

```
cd %userprofile%\.ssh
```

- If you see "No such file or directory", then there aren't any existing keys: [go to step 3](#).
- Check to see if you have a key already:

```
dir id_*
```

If there are existing keys, you may want to use those: go to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).

2. Back up old SSH keys

If you have existing SSH keys, but you don't want to use them when connecting to Bitbucket Server, you should back those up.

In a command prompt on your local computer, run:

```
mkdir key_backup  
copy id_rsa* key_backup
```

3. Generate a new SSH key

If you don't have an existing SSH key that you wish to use, generate one as follows:

1. Log in to your local computer as an administrator.
2. In a command prompt, run:

```
ssh-keygen -t rsa -C "your_email@example.com"
```

Associating the key with your email address helps you to identify the key later on.

Note that the `ssh-keygen` command is only available if you have already [installed Git](#) (with Git Bash). You'll see a response similar to this:

```
C:\Users\ASUS>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:/Users/ASUS/.ssh/id_rsa):
```

3. Just press <Enter> to accept the default location and file name. If the `.ssh` directory doesn't exist, the system creates one for you.
4. Enter, and re-enter, a passphrase when prompted. The whole interaction will look similar to this:

```
C:\Users\ASUS>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:/Users/ASUS/.ssh/id_rsa):
Created directory 'C:/Users/ASUS/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:/Users/ASUS/.ssh/id_rsa.
Your public key has been saved in C:/Users/ASUS/.ssh/id_rsa.pub.
The key fingerprint is:
e6:99:c3:3c:52:fb:9c:e4:3f:df:4d:b2:80:11:a5:1e ASUS@ASUS-PC
C:\Users\ASUS>
```

5. You're done! Now go to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).

Creating an SSH key on Linux & Mac OS X

1. Check for existing SSH keys

You should check for existing SSH keys on your local computer. *You can use an existing SSH key with Bitbucket Server if you want, in which case you can go straight to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).*

Open a terminal and run the following:

```
cd ~/.ssh
```

- If you see "No such file or directory, then there aren't any existing keys: [go to step 3](#).
- Check to see if you have a key already:

```
ls id_*
```

- If there are existing keys, you may want to use them; go to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).

2. Back up old SSH keys

If you have existing SSH keys, but you don't want to use them when connecting to Bitbucket Server, you should back those up.

Do this in a terminal on your local computer, by running:

```
mkdir key_backup
cp id_rsa* key_backup
```

3. Generate a new key

If you don't have an existing SSH key that you wish to use, generate one as follows:

1. Open a terminal on your local computer and enter the following:

```
ssh-keygen -t rsa -C "your_email@example.com"
```

Associating the key with your email address helps you to identify the key later on.

You'll see a response similar to this:

```
pwatsons-Mac-Pro: ~ pwatson$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/pwatson/.ssh/id_rsa):
```

2. Just press <Enter> to accept the default location and file name. If the `.ssh` directory doesn't exist, the system creates one for you.
3. Enter, and re-enter, a passphrase when prompted.

The whole interaction will look similar to this:

```
pwatsons-Mac-Pro: ~ pwatson$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/pwatson/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/pwatson/.ssh/id_rsa.
Your public key has been saved in /Users/pwatson/.ssh/id_rsa.pub.
The key fingerprint is:
47: 68: 04: f3: 93: 53: a3: af: bd: fc: 86: 60: 30: 47: cd: ea pwatson@pwatsons-Mac-Pro.local
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      o..oo          |
|      +. =o.         |
|      .O. .          |
|      o.o=           |
|      =S o           |
|      E+             |
|      .....         |
|      ....           |
|      oo.            |
+-----+
pwatsons-Mac-Pro: ~ pwatson$
```

4. You're done! Now go to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).

SSH user keys for personal use

You can use SSH keys to establish a secure connection between your computer and Bitbucket Server for when you are performing read-only (pull, clone) Git operations from your local machine. Personal keys are attached to your Bitbucket Server account – they are bound by that account's permissions and use the account's identity for any operations.

Before you can use SSH keys to secure a connection with Bitbucket Server the following must have already been done:

- your Bitbucket Server administrator must have already [enabled SSH](#)

[access](#) in Bitbucket Server.

- you need an SSH key! See [Creating SSH keys](#). Alternatively, you can use an existing key, if it isn't already being used as a repository or project access key.
- you need to have added your personal SSH key to your Bitbucket Server account – see the following section.

Once you have an SSH key associated with your Bitbucket Server account, using it is easy! See [Use SSH keys to connect to Bitbucket Server repositories](#) below.

Related pages:

- [Creating SSH keys](#)
- [Enabling SSH access to Git repositories in Bitbucket Server](#)
- [Permanently authenticating with Git repositories](#)

Note that:

- Bitbucket Server supports DSA and RSA2 key types – RSA1 is not supported.
- A Bitbucket Server user can add any number of keys to their account.
- You can use the same SSH access key for multiple repositories or projects.
- Keys used for personal user accounts can't be re-used as a [project or repository access key](#), and keys used as a project or repository access key can't be re-used for user accounts.

Add an SSH key to your Bitbucket Server account

1. On Windows, in your command prompt, change directory to your .ssh directory, and copy the public key file to your clipboard by running:

Windows

```
cd %userprofile%\.ssh
clip < id_rsa.pub
```

On Mac OS X or Linux simply run the following in a terminal:

Mac OS X

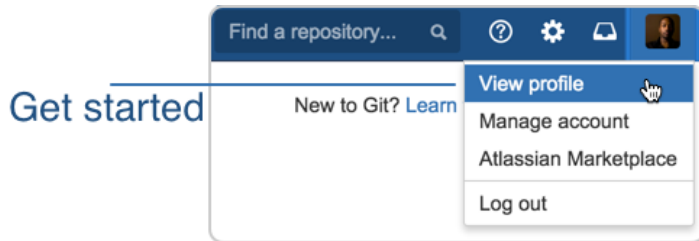
```
pbcopy < ~/.ssh/id_rsa.pub
```

Note that on Linux, you may need to download and install xclip, then use that, as shown in this code snippet:

Linux

```
sudo apt-get install xclip
xclip -sel clip < ~/.ssh/id_rsa.pub
```

2. In Bitbucket Server, go to your account:



3. Click on **SSH keys** and then **Add key**.
4. Paste the key into the text box:

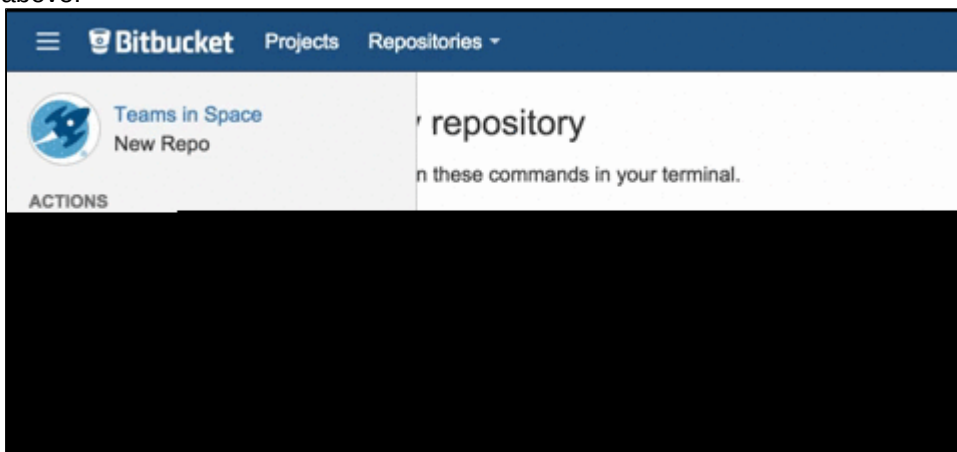


5. Click **Add key**. You're done!

Use SSH keys to connect to Bitbucket Server repositories

SSH access needs to have been set up, as [described above](#). Once this is done, you can use SSH keys as follows:

1. Go to **Projects**, click a project, and choose a repository from the list.
2. Click **Clone** in the sidebar to see the clone URLs for the repository.
3. Choose the clone URL you want to use. SSH is available if you have already added an SSH key to your account. If you haven't done that yet, see [Add an SSH key to your Bitbucket Server account](#), above.



SSH access keys for system use

Bitbucket Server administrators can set up SSH access keys to secure the Git operations that other systems perform on the repositories managed in Bitbucket Server. Using access keys avoids the need to store user credentials on another system, and means that the other system doesn't have to use a specific user account in Bitbucket Server. For example, access keys can be used to allow your build and deploy server to authenticate with Bitbucket Server to check out and test source code.

- **Project admins** can add and manage SSH access keys for a project.

- The keys apply to every repository in the project.
- [Repository admins](#) can add and manage SSH access keys for a particular repository.
- The access key can allow either *read-only* or *read-write* Git operations.

Related pages:

- [Creating SSH keys](#)
- [Enabling SSH access to Git repositories in Bitbucket Server](#)
- [Permanently authenticating with Git repositories](#)

Settings

- Project details
- Audit log
- Permissions
- Access keys**

Access keys Add key

Access keys provide a simple way for other systems to access repositories using SSH without storing user credentials.

Label	Key
mobileapp@team-game.com	ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA0Mhb5KFkuHbZYvZFNvpI0yum8O1D...
twitterapp@ridinghigh.com	ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6CYOy4zGRuXf3wHsgITnoVJLWYFqx...

Note that Bitbucket Server supports DSA and RSA2 key types – RSA1 is not supported.

Before you can use SSH keys to secure a connection with Bitbucket Server the following must have already been done:

- Your Bitbucket Server administrator must have already [enabled SSH access](#), on Bitbucket Server.
- You must have already created an SSL key. See [Creating SSH keys](#). Alternatively, you can use an existing key, if it isn't already being used for a personal account in Bitbucket Server.

Using SSH keys to allow access to Bitbucket Server repositories

To get the SSH key to work with your build, or other, system, you need to:

- Add the private key to that system. For Bamboo, see this page: [Shared credentials](#).
- Add the public key to Bitbucket Server as described here:

Add an SSH access key to either a Bitbucket Server project or repository

You simply copy the public key, from the system for which you want to allow access, and paste it into Bitbucket Server.

- Copy the public key. One approach is to display the key on-screen using `cat`, and copy it from there:

```
cat < ~/.ssh/id_rsa.pub
```

- Now, in Bitbucket Server, go to the **Settings** tab for the project or repository.
- Click **Access keys** and then **Add key**.
- Choose the **Read** permission, for `git pull` or `git clone` operations for example, where you want to be sure that the system will *not* be able to write back to the Bitbucket Server repository.
- Choose the **Read / Write** permission, for `git push` or `git merge` operations for example, where

you may want your build system to merge successful feature branch builds to the default branch in the Bitbucket Server repository, or so that deployments can be tagged.

Note that if you attempt to add a key already present on a project or repository but with a different permission to what it currently has, the permission will simply be updated.

5. Paste the key into the text box and click **Add key**.

Bitbucket Server license implications

- System access keys do not require an additional Bitbucket Server user license.

Reusing access keys

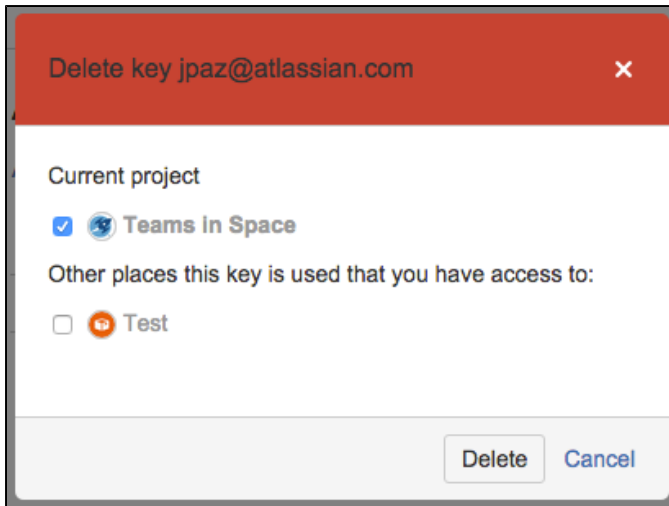
- You can use the same SSH access key for multiple repositories or projects.
- Keys used for [personal user accounts](#) can't be re-used as a project or repository system access key, and keys used as a project or repository access key can't be re-used for user accounts.

Deleting an access key

You can delete an access key by going to **Settings** > **Access keys** for the repository, and clicking the cross for the key (the cross only appears when you move the mouse pointer there):

Label	Key	Permission
csmajda@csm...	ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAC2ABpxSHgfgx0...	Read
cszmajda@sta...	ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADYgllH2fkBQtW...	Read

If the key is used for multiple projects or repositories, you can select the other places that you want the key to be deleted from:



Note that the dialog only displays the projects and repositories that you have permission to see. Be aware that the key may also be used in other places that are not listed in the dialog. To be 100% sure that *all* uses of the key are deleted, this operation must be performed by someone with the administrator or sysadmin [global permission](#).

Workflow strategies in Bitbucket Server

Various Git workflows are supported by Bitbucket Server:

- [Centralized Workflow](#)
- [Feature Branch Workflow](#)
- [Gitflow Workflow](#)
- [Forking Workflow](#)

For information about setting up Git workflows in Bitbucket Server see [Using branches in Bitbucket Server](#) and [Using forks in Bitbucket Server](#).

Centralized Workflow

Like Subversion, the



Centralized Workflow uses a central repository to serve as the single point-of-entry for all changes to the project. Instead of trunk, the default development branch is called master and all changes are committed into this branch. This workflow doesn't require any other branches besides master.

Read more about the [Centralized Workflow...](#)

Feature Branch Workflow

The core idea behind the Feature Branch Workflow is that all feature development should take place in a dedicated branch instead of the master branch. This encapsulation makes it easy for multiple developers to work on a particular feature without disturbing the main codebase. It also means the master branch will never contain broken code, which is a huge advantage for continuous integration environments.

Read more about the [Feature Branch Workflow...](#)

Gitflow Workflow

The Gitflow Workflow defines a strict branching model designed around the project release. While somewhat more complicated than the Feature Branch Workflow, this provides a robust framework for managing larger projects.

Read more about the [Gitflow Workflow...](#)

Forking Workflow

The Forking Workflow is fundamentally different than the other workflows discussed in this tutorial. Instead of using a single server-side repository to act as the “central” codebase, it gives every developer a server-side repository. This means that each contributor has not one, but two Git repositories: a private local one and a public server-side one.

Read more about the [Forking Workflow...](#)

Using branches in Bitbucket Server

Bitbucket Server makes it easy for each member of your team to use a [branching workflow](#) for your Git development process. Your workflow can be mapped to branches in the Bitbucket Server 'branching model', allowing Bitbucket Server to:

- guide your developers into making consistent naming decisions when creating branches.
- identify the type of each branch and apply actions like automatic merging accordingly.

On this page:

- [Configuring the branching model](#)
- [Creating branches](#)
- [Automating the branch workflow](#)
- [Managing all your branches](#)
- [Read more](#)

See also [Using branch permissions](#) for information about restricting access to branches in Bitbucket Server.

Configuring the branching model

Bitbucket Server uses a 'branching model' to define the branch workflow for each repository. As a project administrator, configuring the model lets you:

- enable the branch types that will be available in your workflow.
- specify the naming convention to be used for each branch type.

The naming convention simply adds prefixes to branch names, so that branches of the same type get the same prefix.

A Bitbucket Server admin can configure the branching model for a repository, by going to **Settings > Branching model** for the repository and clicking **Enable branching model**. Note that for *new* repositories, the branching model is enabled by default, and uses the default branch prefixes.

Bitbucket Server makes a number of branch types available, as described below. Use the checkboxes to enable just those branch types that map to your workflow. Note that several branch types have default branch naming prefixes (for example the default prefix for the 'feature' branch type is `feature/`), as shown:

Development

This is generally the integration branch for feature work and is often the default branch (e.g. `master`) or a named branch such as `develop`. In a workflow using pull requests, this is usually the branch where new feature branches are targeted. In other cases, developers might commit directly to this branch.

Feature

feature/

Feature branches are used for specific feature work or improvements. They generally branch from, and merge back

into, the development branch, by means of pull requests. See [Feature branch workflow](#).

Production

The production branch is used while deploying a release. It branches from, and merges back into, the development branch. In a Gitflow-based workflow it is used to prepare for a new production release.

Release

release/

Release branches are used for release task and long-term maintenance of software versions. Typically, they branch from, and fixes are merged back into, the development branch. Merging into an older release branch allows for [automatic merging](#) to newer release branches as well as the development branch.

Bugfix

bugfix/

Bugfix branches are typically used to fix release branches.

Hotfix

hotfix/

Hotfix branches are used to quickly fix the production branch without interrupting changes in the development branch. In a Gitflow-based workflow, changes are usually merged into the production and development branches.

Note that:

- Prefixes can't be empty.
- Prefixes can't be longer than 30 characters.
- Prefixes can't overlap; for example PROD and PRODUCT would be overlapping prefixes.
- For Bitbucket Server instances using Microsoft SQL Server, prefixes can't use non-ASCII characters.

See [BSERV-3884](#) - Non-ASCII values used as branch model prefixes/branch names don't work in MSSQL

CLOSED

Creating branches

You can create a new branch when [in JIRA Software](#) (version 6.1 and above) or [in Bitbucket Server](#). Either way, you can [override the settings](#) that Bitbucket Server suggests for the repository, branch type, branching point and branch name.

Create a branch from a JIRA Software issue

i JIRA Software must be connected with Bitbucket Server by an [application link](#) for this functionality to be available.

When viewing an issue in JIRA Software, click **Create Branch** (under 'Development' – you'll need the 'View Development Tools' project permission within JIRA Software to see this):

Dates

Created:

03/Jul/15 3:21 PM

Updated:

04/Jul/15 2:32 PM

Development

[Create branch](#)

Choose the SCM, if more than one is available, where you want to create the branch.

Bitbucket Server suggests the **Branch type** and **Branch name** based on the JIRA Software issue type and summary. Change the [settings](#) suggested by Bitbucket Server, if necessary:

Create branch for TIS-72

Repository

Teams in Space / Website ▾

Branch type

Feature ▾

[Learn about branch types](#)

Branch from

master ▾

Branch name

feature/ TIS-72-add-video-chat-interface



master

feature/TIS-72-add-video-chat-interface

Create branch

Cancel

Create a branch from within Bitbucket Server

In Bitbucket Server, choose **Create branch** from the sidebar.

Bitbucket Server will suggest the **Branch type** and **Branch name** based on the JIRA Software issue type and summary. Notice that Bitbucket Server displays the current [build status](#) beside the source branch picker. Change the [settings](#) suggested by Bitbucket Server if necessary:

Create branch

Repository

Teams in Space / Website ▾

Branch type

Bugfix ▾

[Learn about branch types](#)

Branch from

release/1.8 ▾

Branch name

bugfix/ TIS-72-add-video-chat-interface



release/1.8

bugfix/TIS-72-add-video-chat-interface

Create branch

Cancel

Creating the branch

You can specify:

- the **Repository**
- the **Branch type**, if a [branching model](#) has been previously configured – choose **Custom** if you need an *ad hoc* branch type
- the **Branch from** point – you can choose either a branch or a tag.
- the **Branch name** – the prefix is based on the branch type you selected, and as defined by the [branching model](#). Note that the branch name should follow your team's convention for this.

Note that Bitbucket Server suggests a **Branch type** based on the JIRA Software issue type, when a [branching model](#) is configured. The mapping is:

JIRA Software issue type	Bitbucket Server branch type
Bug	Bugfix
Story	Feature
New Feature	Feature

Once the new branch is created, Bitbucket Server takes you to the file listing for that. You can now pull to your local repository and switch to the new branch.

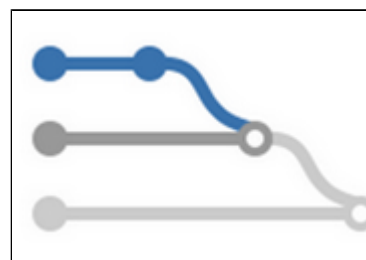
Automating the branch workflow

Bitbucket Server can automate some merges in the branch workflow, based on the branching model for the repository. This allows merges to be cascaded to newer branches of the same parent, subject to a few conditions, so reducing the need for manual maintenance of branches.

As a project administrator you can turn on automatic merging for a particular repository. Go to **Settings > Branching model** for the repository, and select **Enable automatic merging** (under 'Automatic merge').

If Bitbucket Server cannot perform an automatic merge, perhaps because branch permissions prevent it, Bitbucket Server creates a new pull request for that merge, and the automatic merge operation stops. This allows you to resolve the conflict locally before approving the new pull request, which may involve further cascading merges.

See [Automatic branch merging](#) for more information about the conditions for automatic merging, and how Bitbucket Server determines the ordering of branches.



Managing all your branches

The branch listing page makes it easy to keep track of all the branches in your repository.

Searching for branches

You can easily find branches by using the search at the top of the table. Furthermore, if you're using the Bitbucket Server [branch model](#), you can filter by branch type simply by searching for the prefix – for example, search for "feature/" to see all your feature branches.

You can find the feature and bugfix branches that haven't yet been merged into a particular release (for example, "release/2.10") by changing the 'base branch' – just use the branch selector (arrowed in the screenshot below) to change the base branch, and refer to the **Behind/Ahead** and **Pull requests** columns.

Reading the table

Behind/Ahead

The Behind/Ahead column shows by how many commits a branch has diverged from the 'base branch' (for example, master). Use the branch selector (arrowed in the screenshot below) to change the base branch.

Pull requests

The Pull requests column shows the most relevant status from the pull requests against each branch – click an icon to see details. The status is:

- OPEN if there is at least one open pull request.
- MERGED if there are no open pull requests, and at least one pull request has been merged.
- DECLINED if there are no open or merged pull requests, and at least one pull request has been declined.

Builds

If you have an [integrated build server](#), the Builds column shows the status of the latest build results published to Bitbucket Server. The overall status is 'passed' if all the different builds (for example, unit tests, functional tests, deploy to staging) succeeded and 'failed' if at least one run failed for any of those. Click an icon to see details of the builds.

Actions

The Actions menus include tasks for working with branches:

- Check out in SourceTree
- Create a pull request
- Edit permissions
- Delete branch

Navigation

Choose **Keyboard shortcuts** from the Bitbucket Server Help menu to see shortcuts to help you navigate quickly around the branch listing.

Checking on your branches

The branch listing allows you to:

- See how many commits behind or ahead your branch is compared to a chosen 'base branch'.
- See the latest status for pull requests originating from branches.
- See the build status of branches at a glance.
- The **Pull requests** status helps you to track the review and merge work that still needs to be done and can help with branch cleanup. For example, in combination with the **Behind/Ahead** information, you can decide whether to remove a feature branch that has already been merged.
- The **Behind/Ahead** column can help you to identify work in progress as well as stale branches. It is calculated for each branch against the base branch.

Branches				
bugfix/TIS-73-fix-audio-in-v... ... Search branches				
Branch		Behind/Ahead	Updated	P
bugfix/TIS-73-fix-audio-in-video-chat	BASE BRANCH		1 min ago	
feature/TIS-77-add-screenshare-for-video-chat		5 1	22 Aug 2014	
AUI-2167-allow-html-in-checkbox-labels		3	14 Aug 2014	
AUI-2568-responsiveheader-improvements		5 2	14 Aug 2014	

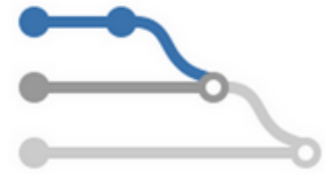
Read more

- <http://blogs.atlassian.com/2013/10/inside-atlassian-feature-branching-on-the-stash-team/>

Automatic branch merging

Bitbucket Server can automatically merge changes to newer release

branches, thus reducing the need for manual maintenance of branches. To be able to do this, Bitbucket Server has to be able to determine the [ordering of branches](#), and relies on [semantic versioning](#) of branch names – for example Bitbucket Server will order these branch names like this: 1.0.0 < 2.0.0 < 2.1.0 < 2.1.1.



Note that:

- Automatic branch merging is subject to a few [conditions](#).
- Automatic merging is off by default for new and existing repositories.
- You must explicitly enable automatic merging for each repository.
- The commit message will indicate that the merge was automatic.
- Bitbucket Server records full audit log entries for automatic merges.
- Bitbucket Server sends [notifications](#) when automatic merges succeed (or fail).

As a project administrator, turn on automatic merging by going to **Settings** > **Branching model** for a repository, and selecting **Enable automatic merging** (under 'Automatic merge').

On this page:

- [Conditions for automatic merging](#)
 - [What happens if the automatic merge fails?](#)
- [Branch ordering algorithm](#)
- [Ordering examples](#)

Conditions for automatic merging

The following conditions must be satisfied for Bitbucket Server to be able to automatically cascade changes:

- The Bitbucket Server [branching model](#) must be configured for the repository.
- The 'release' branch type must be enabled or a 'production' branch must be set for the repository.
- The merge must go via a pull request.
- The pull request must be made to a branch that is of the 'release' type or the 'production' branch.
- The target branch of the pull request must have branches that are [newer](#) than it.

Note that Bitbucket Server expects that the 'development' branch (commonly the 'default' branch) is always a head of any 'release' branches. The final merge in the automatic cascade will be to the 'development' branch.

What happens if the automatic merge fails?

The automatic merge can fail for reasons such as:

- Branch permissions prevent cascading changes to a particular branch.
- Bitbucket Server detects a conflict that prevents the merge.
- There is already an open pull request with the same source and target that the automatic merge would close.

For the first two cases, Bitbucket Server creates a new pull request for the failed merge, and the automatic merge operation stops. This allows you to resolve the conflict locally before approving the new merge, which may start a new series of cascading merges. Note that a pull request that gets automatically opened when a merge fails won't trigger the continuation of the initial merge chain if resolved locally (which is the approach that we recommend).

Branch ordering algorithm

Bitbucket Server is able to automatically merge changes to newer release branches, as long as Bitbucket Server can determine the ordering of those branches. Ordering is based on [semantic versioning](#) in the naming pattern for branches.

Bitbucket Server uses the following ordering algorithm to determine the branches in the merge chain:

- Branches are selected and ordered on the basis of the name of the branch that started the cascade (i.e. the target of the pull request for the merge).

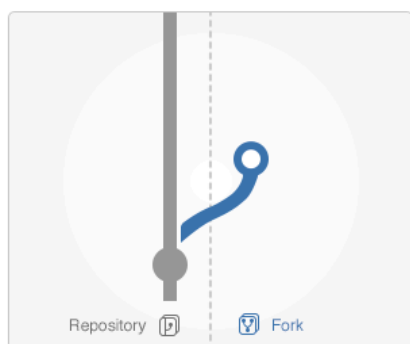
- Branch names are split into tokens using any of these characters: underscore '_', hyphen '-', plus '+', or period '.'.
- Only branches *matching* the name of the pull request target are added into the merge path. Matching means that every token before the first numeric token must be equal to the corresponding tokens of the target branch's name.
- Branches are ordered by number, if a given token is numeric. When comparing a numeric token with an ASCII token, the numeric is ranked higher (i.e. is considered as being a newer version).
- If both tokens are non-numeric, a simple ASCII comparison is used.
- In the unlikely case of the above algorithm resulting in equality of 2 branch names, a simple string comparison is performed on the whole branch name.
- There is a limit of 30 merges.

Ordering examples

The table below provides examples of branch naming patterns that Bitbucket Server is able, and not able, to order correctly:

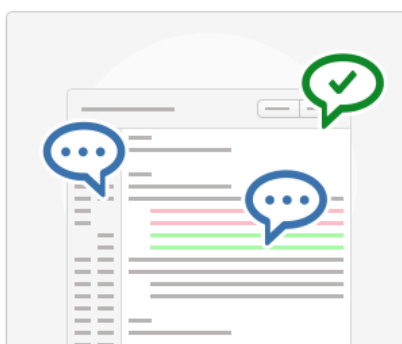
GOOD	• release/1.0 • release/1.1-rc1 • release/1.1 • release/1.2 • release/2.0	Bitbucket Server tokenizes on the '.' and the '-' of '1.1-rc1' and is able to order these branch names correctly.
GOOD	• release/bitbucket_1.1 • release/bitbucket_1.2 • release/bitbucket_2.0	Bitbucket Server tokenizes on the '.' and the '_' and orders the numeric parts of these branch names correctly.
BAD	• release/1.0 • release/bitbucket_1.1	Bitbucket Server tokenizes on the '.' and the '_' but cannot recognize that 'bitbucket_1.1' should follow '1.0'.

Using forks in Bitbucket Server



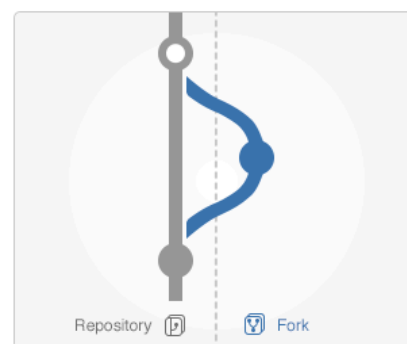
Fork

Develop features on a branch and create a pull request to get changes reviewed.



Discuss

Discuss and approve code changes related to the pull request.



Merge

Merge the branch with the click of a button.

Forks provide an alternative workflow to using branches, for where particular developers have restricted (read-only) access to a repository. See [Workflow strategies in Bitbucket Server](#) for more information.

You can fork a repository into any other project in Bitbucket Server for which you have admin access. You can also create [personal forks](#) and give other developers access to that using repository permissions.

Creating a fork

You can create a fork for any repository that you can see in Bitbucket Server (that is, for which you have 'read' permission).

Simply click **Fork** in the sidebar. You can choose the location for the newly forked repository. Note that when a

repository is forked into another project it will get that project's permissions, which may be less restrictive.

When creating the fork you can enable [fork syncing](#) to have Bitbucket Server automatically keep your fork up-to-date with changes in the upstream repository.

On this page:

- [Creating a fork](#)
- [Issuing a pull request for a fork](#)
- [Merging a fork](#)
- [Synchronizing with upstream](#)
- [Disabling forking](#)
- [Pre-receive hooks and forks](#)

Related pages:

- [Workflow strategies in Bitbucket Server](#)
- [Controlling access to code](#)
- [Creating personal repositories](#)

Issuing a pull request for a fork

Pull requests for forks in Bitbucket Server work just the way you'd expect. See [Using pull requests in Bitbucket Server](#).

When creating the pull request, you can choose the fork and the branch that contains the source to be pulled, as well as the destination fork and branch.

Merging a fork

Once a pull request has been approved by reviewers, it can be merged as usual. See [Using pull requests in Bitbucket Server](#).

Synchronizing with upstream

Once you fork a repository, your fork can be kept up-to-date with changes in the upstream repo either automatically by Bitbucket Server or you can synchronize manually. You will still need to keep your remote working repository synced with your fork in Bitbucket Server yourself. See [Keeping forks synchronized](#) for more details.

Disabling forking

Forking of repositories is available by default. However, you can turn off forking, on a per-repository basis, if this helps you to control your development process. You can do this on the **Repository details** tab of the repository settings.

Note that disabling forking on the parent repo doesn't delete any existing forks, and doesn't prevent those existing forks from being forked. Pull requests will still work from the existing forks. Furthermore, commits in the parent are viewable via the fork if the SHA1 hash is known to the user.

Pre-receive hooks and forks

Pre-receive hooks aren't copied with the fork and so are not run when code is merged in a pull-request. This means that custom hooks are unable to prevent certain changes from being merged by pull requests from forks. Instead, the hook would have to also implement a merge-check (see <https://developer.atlassian.com/bitbucket/server/docs/latest/how-tos/repository-hooks.html>).

Keeping forks synchronized

Fork syncing helps you to keep your fork in Bitbucket Server up-to-date with changes in the upstream repository. Bitbucket Server can do this automatically for all branches and tags you haven't modified in the fork.

If you have modified branches or tags in the fork, Bitbucket Server will offer syncing strategies. Bitbucket Server will never update your branch or tag in your fork if this means that your changes would be lost.

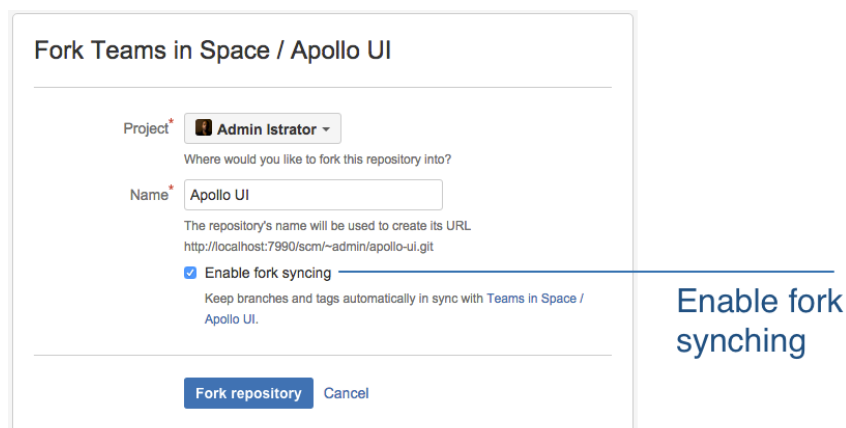
Note that syncing is about pulling recent upstream changes into your fork, whereas pull requests are about pushing your changes back to the upstream repository.

On this page:

- [Enabling automatic fork syncing](#)
- [What gets synced?](#)
- [Manual synchronization strategies](#)

Enabling automatic fork syncing

You can enable automatic fork syncing when you first fork the repository:



You can also enable fork syncing at any later time by going to **Settings > Fork syncing** for the forked repository. Syncing is disabled by default.

What gets synced?

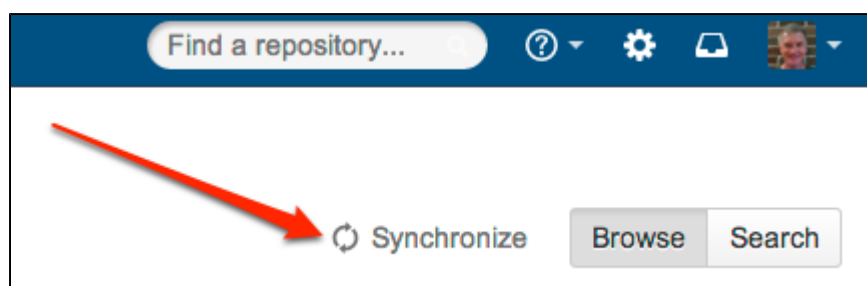
When performing automatic synchronization, Bitbucket Server updates the fork as follows:

- for branches - Bitbucket Server makes any fast-forward change, where there is no need to merge work and there is no risk of losing changes.
- for tags - Bitbucket Server makes updates only if the current state is the same as what upstream pointed to. So, a new tag in upstream will create a new tag in the fork, unless you have a tag of the same name, when the update will fail.

Manual syncing

If upstream and your fork have diverged, so that each has changes that are not in the other, Bitbucket Server will not perform a merge automatically. When you visit the branch in Bitbucket Server, you have the option to manually synchronize the branch.

You can manually synchronize your branch at any time using **Synchronize** by going to the **Settings > Fork syncing** tab for the forked repository, or on either of the **Source** or **Commits** tabs for a repository:

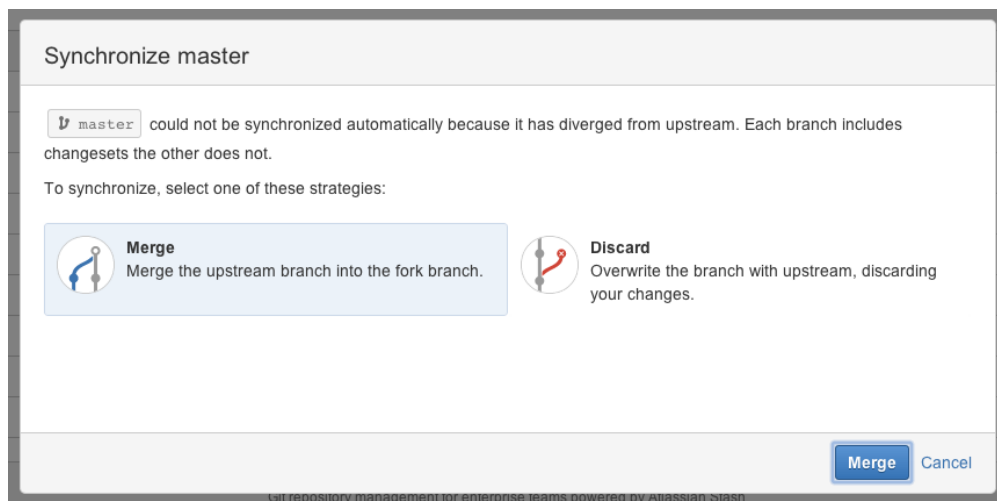


Manual synchronization strategies

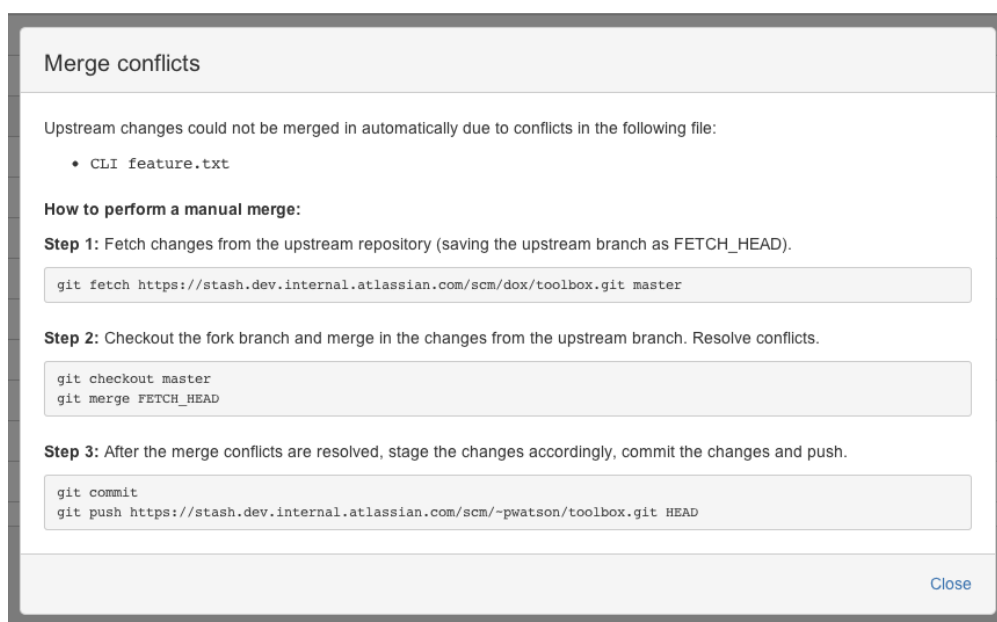
When you initiate a manual synchronization, Bitbucket Server will ask you to choose one of the following synchronization strategies.

Merge strategy

Merge the upstream branch into the fork branch.



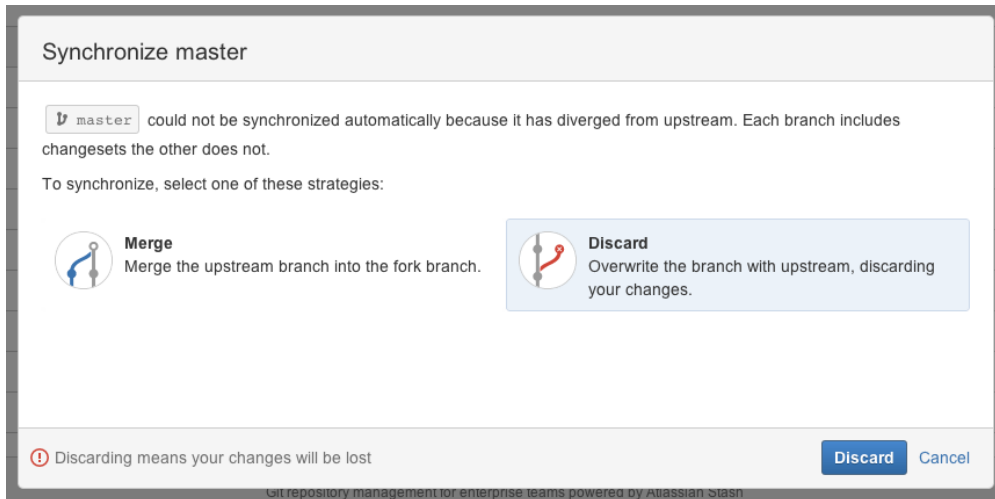
If Bitbucket Server detects conflicts when trying to perform the merge it will offer hints on how to resolve those:



Once the merge is complete, your branch will have incorporated all the commits on the branch in the parent repository, but your branch will still be ahead of the parent (it has your changes on it). This means automatic synchronization for this branch will not occur until your changes are pushed to the parent repository.

Discard strategy

Overwrite your changes in your fork with the upstream branch. Your changes will be lost.



Using pull requests in Bitbucket Server

Pull requests in Bitbucket Server provide a quick and easy way for software teams to collaborate on code. A pull request is a dedicated forum for discussing a proposed feature. If there are problems with the changes, teammates can post feedback in the pull request and even tweak the feature by pushing follow-up commits. All of this activity is tracked directly inside of the pull request.

A pull request requires differences between two distinct branches. When you create a pull request you'll specify the branch to merge changes into.

Create a branch

Pull requests can be used with the [Feature Branch Workflow](#), the [Gitflow Workflow](#), or the [Forking Workflow](#). You can create branches from the Bitbucket UI, from the command line using Git, or from within a connected JIRA Software instance.

[Read about how to create a branch.](#)

Create a pull request

In their simplest form, pull requests allow a developer to notify team members that they've completed a feature. Once their feature branch is ready, the developer files a pull request via their Bitbucket account. This lets everybody involved know that they need to review the code and merge it into the master branch.

[Read about how to create a pull request.](#)

Review and discuss a pull request

As a reviewer of a pull request, your colleagues are counting on you to review changes to the code then provide feedback. Use reviewer status indicators to let a pull request author know you approve the changes or that changes need more work before you can approve.

To review a pull request, select either



Approve or



Needs work within the header of a pull request. Click the button again or click a different one to change your status.

[Read about how to review and discuss a pull request.](#)

Merge a pull request

Once you are ready to merge a pull request, and when the reviewers have approved it, click **Merge** at the top right of the pull request view. You can merge a pull request if you have write (or admin) [permission](#) on the project.

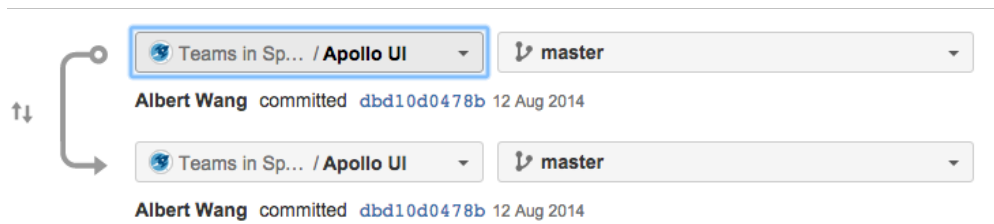
[Read about how to merge a pull request.](#)

Create a pull request

When you are ready to start a discussion about your code changes, it's time to create a pull request. A pull request is a dedicated forum for discussing proposed changes to a project. To create a pull request you'll need a feature branch that you've made changes to.

To create a pull request

1. Click **Create pull request** in the sidebar.
2. Choose the source and destination branches. The source branch is where you made your code changes and the destination is the branch you want to merge to. The source and target branches may be located in different forks:



3. Use the Diff and Commits tabs to compare the source and destination branches before creating the pull request.
4. Click either **Create pull request**, or **Continue**, and enter a title and description that will help people understand what your pull request is about. Use [mentions](#) (to notify another Bitbucket Server user), and [markdown](#) (to add formatting) in your description.
5. Add reviewers – they will receive a notification by email. Other people who have [permissions](#) on the project can participate in the discussion if it interests them.
6. Click **Create**.

You will receive email notifications when your reviewers and other participants comment on the pull request, or commit changes to it.

Edit a pull request

After creating a pull request, you can modify it by clicking **Edit** on the pull request's page. You can edit details such as the **Title**, **Description** and the **Reviewers**. In particular, you can change the **Destination** branch for the pull request – you'll need [Read permission](#) on the branch you want to set.

Review and discuss a pull request

The review phase of a pull request typically involves reviewers making comments and the author pushing additional changes and commenting in response, until the pull request is ultimately approved. The pull request author usually starts by adding colleagues as reviewers. Reviewers then [leave comments](#)—either on the entire pull request or on a specific part of the code changes—and then [updates their reviewer status](#) to notify the author that they've completed their review. Create reminders within a pull request to ensure suggestions are incorporated by creating a [pull request task](#).

Depending on the feedback provided by reviewers, the author may then update a pull request with new commits. This may be to clean up the code, resolve any outstanding tasks, or improve the quality of the code.

Once the cycles of reviewer feedback and new commits has reached a conclusion, a pull request can either be [merged to its target branch](#), or declined if the changes are not to be merged.

Review a pull request

Bitbucket allows you to add one or more reviewers to a single pull request who can then approve (or reject) the request. Pull requests give those who maintain the repository the ability to review the quality of the code that's specified in the pull request.

To review a pull request

1. **Access the pull request** by either following links from an email notification, selecting a notification within the pull request inbox (in the upper-right), or searching for a pull request by clicking **Pull requests** on the sidebar (read more about [searching for pull requests](#)).
2. **Review the changes** and comments left by your teammates within the pull request.
3. **Leave some feedback** about the changes, in any of the views, and use @ mentions to ask questions directly of your colleagues, who will receive a notification after you enter your comment. See the [Discuss a pull request](#) section for more details about the various ways you can leave comments, including [pull request tasks](#).
4. **Finish your review** by indicating if you feel the pull request can be merged, or if the author of the pull needs to make additional changes before you can provide your stamp of approval. Select either the [status indicators](#) to let your team know you've reviewed the changes and the ball is now in their court:
 - a. Approve - indicates you've reviewed the changes and the code is ready to be merged.
 - b. Needs work - indicates you've reviewed the changes, but the code is not quite ready to be merged.

Discuss a pull request

The most important aspect of a pull request is the discussion it generates. You can comment on the entire pull request, a particular file, or on specific lines of code in a file. Comment likes are a quick way of amplifying review feedback – effectively saying "also consider this person's feedback". You can also attach a task to any comment, so actions identified during the review can be easily tracked and resolved. Read more about [pull request tasks](#).

There are three main ways to view changes:

1. The overview tab - lists all of the activity for a pull request since the pull request was created.
2. The diff tab - highlights which lines of code have been added, deleted, or modified.
3. The commits tab - lists all the commits that will get merged. You can click to view a single commit.

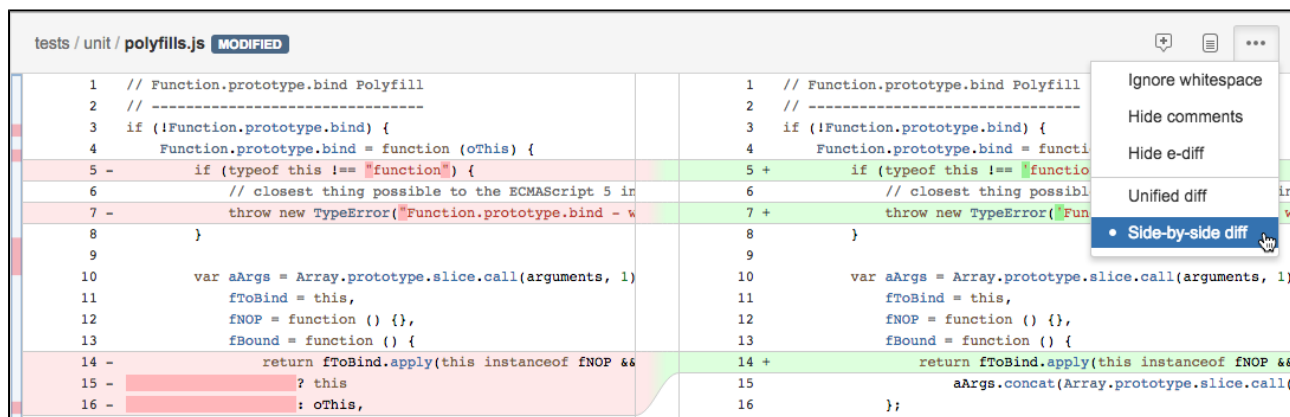
Overview

The Overview tab captures all of the team's activity on the pull request in one place, right from the initial creation, through to when it is finally merged (or declined), with all the comments, replies and commits that happen along the way.

You can add a comment on the **Overview** tab (just under 'Activity'), or reply to a previous comment. Use [mentions](#) to alert another Bitbucket Server user to your comment, and use [markdown](#) to add formatting, for example headings or lists.

Diff view

The diff view highlights the changes that will result when the merge occurs, so you can see exactly what the effect of the merge will be. The **Side-by-side diff** option from the Action menu lets you easily compare the changes that will be merged. Use the N (next) and P (previous) keyboard shortcuts to move between hunks in a diff. Use Shift+N (next) and Shift+P (previous) to move between comments in a diff. The 'map' in each margin of the side-by-side diff provides a visual summary of the diff hunks, and indicates which part of the file you're currently viewing:



Commits

The **Commits** tab lists all the commits that will get merged (those that are greyed out have already been merged). Clicking through to a commit takes you out of the pull request context.

When viewing a commit you can comment on the whole file, or a particular line of code, just as for a diff, for any file in the commit.

Participants can commit new changes to the branch. Bitbucket Server auto-updates the **Commits** tab of the pull request, so you can see exactly which commits will be merged. Bitbucket Server is smart about comments, moving them along when lines are added or removed. If a line with a comment gets removed, you can still view the comment in the activity, but Bitbucket Server marks the diff as *outdated* to let you know that this piece of code has been changed in recent commits.

Pull request tasks

You can attach one or more tasks to any pull request comment, to track required work identified during a review. Anyone with permission to browse a pull request can create a task on any comment, and can browse, resolve or reopen existing tasks in the pull request. Repository admins and pull request authors can edit and delete any task in the pull request. Reviewers and others can only edit or delete their own tasks. A Bitbucket Server administrator can set a merge check that requires all tasks to be resolved before the pull request can be merged. See [Checks for merging pull requests](#).

To create a pull request task, highlight some text in the comment, then click **Create task** – the task is automatically created and saved with that text. **To indicate a pull request task is done**, tick the box in front of the task.

To see all the unresolved tasks for a pull request, use Shift+T when viewing the pull request. Click the 'link' icon for a task to see the task in the context of the comment and source code.

Watching and notifications

You automatically get added as a watcher of a pull request when you are added to the pull request as a reviewer, or when you perform an action related to the pull request (such as adding a comment):

Action	You're now a watcher
You are added as a reviewer	✓
You comment on a pull request	✓
You reply to a comment	✓
You push to the source branch	✓

You approve the pull request



You can manually add yourself as a watcher by clicking the **Watch** button on the pull request screen.

You can stop watching a pull request by clicking the link in the email notification, or the **Unwatch** button on the pull request screen. If you stop watching a pull request you will not automatically be added as a watcher again if you subsequently perform an action that would otherwise have added you.

Bitbucket Server sends email notifications to watchers when certain [pull request events](#) occur. By default, email notifications are batched, but you can change your personal account settings (on the **Notification settings** tab) so that you get notifications immediately. Note that notifications are *always* sent immediately:

- To the reviewers when a pull request is created.
- To a user when they are added as a reviewer to a pull request.
- To a user when they are mentioned in the description of a pull request.

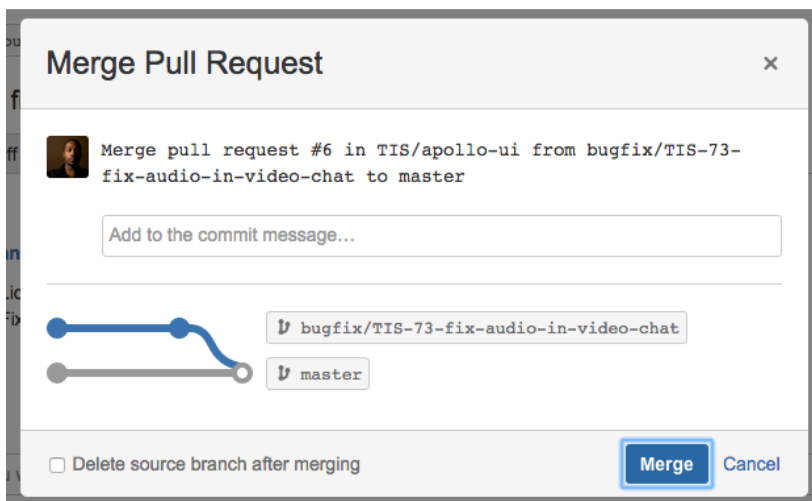
See [Notifications](#) for details.

Merge a pull request

Once you are ready to merge a pull request, and when the reviewers have approved it, click **Merge** at the top right of the pull request view. You can merge a pull request if you have write (or admin) [permission](#) on the project.

Bitbucket Server does not enforce particular review workflows, so anyone with write permission on the repository can merge a pull request, including the person who opened it. This flexibility allows different teams to have different approaches to using Bitbucket Server. If your team requires stricter control, consider using [branch permissions](#) to restrict who can merge a pull request to particular users or groups. You might also want to consider using a plugin to enforce a particular workflow, for example to ensure that only approvals from members of your review team allow merging. See [Checks for merging pull requests](#).

In the 'Merge Pull Request' dialog, you can add extra information about the pull request:



The text you add appears between the subject line and the log lines that Bitbucket Server and Git generate.

Check **Delete source branch after merging** if you no longer need that branch in the repository. Bitbucket Server checks on a few things before allowing the deletion – the branch being merged will not be deleted if:

- The branch is the default repository branch.
- The user does not have permission to delete the branch.
- The branch is subject to an open pull request.

Once accepted, the pull request is marked as merged on the **Pull requests** tab.

If Bitbucket Server detects a conflict that prevents the merge, notifications are displayed on the **Overview** and **Diffs** tabs of the pull request. Click **More information** to see instructions for how to resolve the conflict in your local repository.

Search for pull requests

Find any pull request by clicking the pull requests icon on the sidebar, then using the pull request search header to filter which pull requests you're seeing.

Pull request search allows you to filter all pull requests for a repository by

- status
- author (the person who created the pull request)
- branch
- pull requests you—the active user—are reviewing.

Teams in Space / Appllo-ui

Pull requests

FILTER BY: [Open](#) ▾ [Author](#) ▾ [Target branch](#) ▾ [Reviewer is me](#)

Summary	Reviewers	Builds
 STASHDEV-8359: First cut of SetupService, InternalSetupService and SetupResource → Master Last updated 3 minutes ago	   +3 	
 Improve soke shell scripts → Master Last updated 3 days ago	  	
 Bugfix/STASHDEV-8237: clicking on a hit of diff search → release/3.5 Last updated 14 days ago	   +6	
 STASHDEV-8410: add checkstyle options and implement ignoreMissingDeps option → Master Last updated 29 Oct 2014	 	
 #1234: [UPM-4577] Include active property for addon and cloud property if UPM is in OnDemand → Last updated 30 Oct 2014		

Checks for merging pull requests

To help customize your workflow, you can set checks to control when a pull request can be merged. Pull requests cannot be merged if the required checks have not been met. These checks are set separately on each repository in a Bitbucket Server project.

You'll need either project admin, admin or sys-admin [permissions](#) to set merge checks for pull requests.

So, to set merge checks for pull requests, go to a repository in a project and choose **Settings > Pull requests**. Bitbucket Server includes the merge checks described below, or you can write your own [merge request check plugin](#).

Requires a minimum number of successful builds

Select this option to stop pull requests from being merged if they have any unsuccessful builds. For a pull request, this checks builds that run against the latest commit on the source branch.

You must also specify a minimum number of builds - the pull request will not be able to be merged until at least this many builds have completed. Ideally, you should set this to the number of different builds that are configured to run against the branches in your repository.

See [Bamboo integration](#) for more information about integrating Bitbucket Server with your build server.

Requires a minimum number of approvers

Select this option to block merging of a pull request until it has been approved by at least the selected number of participants.

Requires all tasks to be resolved

Select this option to stop a pull request from being merged if any review tasks are still unresolved. Read more about [pull request tasks](#).

Pull requests

☒ Requires approvers

At a minimum, pull requests must be approved by the number of users above before it can be merged

☐ Requires all tasks to be resolved

☐ Requires a minimum of successful builds

If there are more than the specified number of builds, all of them will have to be successful in order to merge the pull request

Search for code in Bitbucket Server

This page explains Bitbucket Server's search syntax to help you find exactly what you're looking for.

Considerations

There are some restrictions on how searches are performed:

- Only the default branch is searchable (for most repositories the default branch will be master).
- Only files smaller than 512 KiB are searchable.
- Wildcard searches (e.g. `qu?ck buil*`) are not supported.
- All punctuation characters are removed. `!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~`
- Regular expressions in queries are not supported, but you can search for special characters.
- Case is not preserved, however search operators must be in ALL CAPS.
- Queries cannot have more than 9 expressions (e.g. combinations of terms and operators).
- The maximum length of a query cannot exceed 250 characters.
- Only code you have permission to view will appear in the search results.

A query in Bitbucket has to contain at least one search **term**, which can either be a *single word* or a *phrase* surrounded by quotes.

Operators

Search operators are words that can be added to searches to help narrow down the results. Operators must be in ALL CAPS. These are the search operators that can be used to search for files:

- AND
- OR
- NOT
- -
- ()

Multiple terms can be used, and they form a boolean query that implicitly uses the AND operator. So a query for `"bitbucket server"` is equivalent to `"bitbucket AND server"`.

Term	Example query	Usage
AND	<code>bitbucket AND server</code>	Matches files that contain both "bitbucket" and "server".

OR	bitbucket OR server	Matches files that contain either "bitbucket" or "server" (or both).
NOT	bitbucket NOT jira	Matches files that contain "bitbucket" but don't contain "jira".
-	bitbucket -jira	Use before a term, matches files that contain "bitbucket" but don't contain "jira".
()	bitbucket AND (server OR cloud)	Matches files that contain "bitbucket" and either "server" or "cloud"

Modifiers

Modifiers can be used to further restrict search results. Use a modifier in the form "key: value". If there are multiple modifiers in a query they are implicitly combined using "AND" and apply to the whole search expression. These are the search modifiers that can be used to search for files.

repo: <reponame> or **repository:** <reponame>

Search within a particular repository. Must be used with a **proj:** modifier.

Term	Example query	Usage
repo:	jira repo:bitbucket proj:atlassian or jira repository:bitbucket proj:atlassian or jira repo:atlassian/bitbucket	Matches files within the "bitbucket" repository within the "atlassian" project that contain the term "jira".

proj: <project key/name> or **project:** <project key/name>

Search all repositories within a particular project for the search term.

Term	Example query	Usage
proj:	jira proj:atlassian or jira project:atlassian	Matches files within the "atlassian" project that contain the term "jira".

Language and file extension modifiers

Code search can be restricted to only consider a particular language or a particular file extension. For some languages adding a lang criteria is equivalent to specifying the file extension. For example, "lang: java" is

equivalent to "ext:java". For other languages multiple file extensions are mapped to a single language. For example, ".hs", ".lhs" and ".hs-bootare" file extensions are used for the Haskell programming language, and will be matched when specifying "lang:haskell".

lang: or language: <language>

Term	Example query	Usage
lang:	jira lang:java or jira language:java	Matches files that contain the term "jira" within Java files, files with .java, .class, or .jar extension.

ext: or extension: <file extension>

Term	Example query	Usage
ext:	jira ext:lhs or jira extension:lhs	Matches files that contain the term "jira" within Haskell files with the .lhs extension.

Notifications

An email server must be configured in Bitbucket Server for email notifications to be sent. See [Setting up your mail server](#). Note that if the mail server fails, notifications will be dropped. See also [HipChat notifications](#).

Pull request notifications

Bitbucket Server sends email notifications to the [watchers](#) and reviewers of a pull request when the following events occur:

Pull request event	Notification
A reviewer is added	IMMEDIATE
A comment is added	BATCHED
A comment is edited	BATCHED
A comment is replied to	BATCHED
A commit is made to the source branch	BATCHED
A pull request is opened	IMMEDIATE
The pull request is approved	BATCHED
The pull request is merged	BATCHED
An automatic merge fails	?
The pull request is declined	BATCHED
The pull request is reopened	BATCHED or IMMEDIATE

By default, email notifications are [batched](#). However, in the following situations notifications are sent immediately:

- When a pull request is first opened, notifications are immediately sent to the reviewers.
- When a pull request is reopened, notifications are immediately sent to the reviewers who have opted in for immediate notifications.
- When someone is added as a reviewer to a pull request, a notification is immediately sent to them.
- When someone is mentioned in the description of a pull request, a notification is immediately sent to them.

You can change your personal account settings (on the **Notification settings** tab) so that you get notifications immediately.

You don't receive notifications for events you initiate yourself. See also [Using pull requests in Bitbucket Server](#).

Batched email notifications

Bitbucket Server sends email notifications to:

- the watchers of a pull request, when certain [pull request events](#) occur,
- those who are mentioned in pull request descriptions or comments,
- comment and pull request authors when their comments get liked.

Notifications are aggregated by user for each pull request and are emailed in a batch. The batch gets sent if things go quiet for a while (10 mins by default), or when the oldest notification gets 'stale' (30 mins by default), whichever comes first.

By default, email notifications are batched. However:

- You can change your personal account settings (on the **Notification settings** tab) so that you get notifications immediately.
- A Bitbucket Server admin can configure the period of inactivity and the staleness timeout period in the [Bitbucket Server config properties file](#).
- A Bitbucket Server admin can change the notification mode for the Bitbucket Server instance to 'immediate' using a [system property](#), but users can still opt in for batched notifications.

Using 'mentions' to notify someone

From Bitbucket Server 2.0 you can use 'mentions' to notify another Bitbucket Server user about the pull request description or comment you are writing. Bitbucket Server sends an email to that person – the emails are [batched](#) if they have opted for batching in their personal account settings.

To use mentions, simply start typing '@' and then the users display name, username or email address, and choose from the list that Bitbucket Server offers. You can use quotes for unusual names, for example if it has spaces. Use Control-Shift-P or Command-Shift-P to preview the mention.

Markdown syntax guide

Bitbucket Server uses [Markdown](#) for formatting text, as specified in [CommonMark](#) (with a few extensions). You can use Markdown in the following places:

- any pull request's descriptions or comments, or
- in [README](#) files (if they have the .md file extension).

Use *Control-Shift-P* or *Command-Shift-P* to preview your markdown.

Markdown syntax

The page below contains examples of Markdown syntax. For a full list of all the Markdown syntax, consult the official documentation on John Gruber's [Daring Fireball](#) site or the [CommonMark](#) specification.

On this page:

- [Markdown syntax](#)
 - [Headings](#)
 - [Paragraphs](#)
 - [Character styles](#)
 - [Unordered list](#)
 - [Ordered list](#)
 - [List in list](#)
 - [Quotes or citations](#)
 - [Inline code characters](#)
 - [Code blocks](#)
 - [Links to external websites](#)
 - [Linking issue keys to JIRA applications](#)
 - [Images](#)
 - [Tables](#)
- [Backslash escapes](#)
- [README files](#)

Headings

```
# This is an H1
## This is an H2
##### This is an H6

This is also an H1
=====

This is also an H2
-----
```

Paragraphs

Paragraphs are separated by empty lines. Within a paragraph it's possible to have a line break, simply press <return> for a new line.

For example,
like this.

Character styles

```
*Italic characters*
_Italic characters_
**bold characters**
__bold characters__
~~strikethrough text~~
```

Unordered list

- * Item 1
- * Item 2
- * Item 3
 - * Item 3a
 - * Item 3b
 - * Item 3c

Ordered list

1. Step 1
2. Step 2
3. Step 3
 1. Step 3.1
 2. Step 3.2
 3. Step 3.3

List in list

1. Step 1
2. Step 2
3. Step 3
 - * Item 3a
 - * Item 3b
 - * Item 3c

Quotes or citations

Introducing my quote:

```
> Neque porro quisquam est qui  
> dolore ipsum quia dolor sit amet,  
> consectetur, adipisci velit...
```

Inline code characters

Use the backtick to refer to a `function()`.

There is a literal ```backtick (`)``` here.

Code blocks

Indent every line of the block by at least 4 spaces.

This is a normal paragraph:

This is a code block.
With multiple lines.

Alternatively, you can use 3 backtick quote marks before and after the block, like this:

```
```
This is a code block
```
```

To add syntax highlighting to a code block, add the name of the language immediately after the backticks:

```
```javascript
var oldUnload = window.onbeforeunload;
window.onbeforeunload = function() {
 saveCoverage();
 if (oldUnload) {
 return oldUnload.apply(this, arguments);
 }
};
```
```

Bitbucket Server uses CodeMirror to apply syntax highlighting to the rendered markdown in comments, READMEs and pull request descriptions. All the common coding languages are supported, including C, C++, Java, Scala, Python and JavaScript. See [Configuring syntax highlighting for file extensions](#).

Within a code block, ampersands (&) and angle brackets (< and >) are automatically converted into HTML entities.

Links to external websites

This is [an example](http://www.example.com/) inline link.

[This link](http://example.com/ "Title") has a title attribute.

Links are also auto-detected in text: <http://example.com/>

Linking issue keys to JIRA applications

When you use JIRA application issue keys (of the default format) in comments and pull request descriptions Bitbucket Server automatically links them to the JIRA application instance.

The default JIRA application issue key format is two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example TEST-123.

Images

Inline image syntax looks like this:

```
![Alt text](/path/to/image.jpg)
![Alt text](/path/to/image.png "Optional title attribute")
![Alt text](/url/to/image.jpg)
```

For example:

```
...
![Mockup for feature A](http://monosnap.com/image/b0cxxxxLGF.png)
...
```

Reference image links look like this:

```
![Alt text][id]
```

where 'id' is the name of a previously defined image reference, using syntax similar to link references:

```
[id]: url/to/image.jpg "Optional title attribute"
```

For example:

```
...
<--Collected image definitions-->
[MockupA]: http://monosnap.com/image/b0cxxxxLGF.png "Screenshot of
Feature A mockup"
...
<!--Using an image reference-->
![Mockup for feature A][MockupA]
...
```

Tables

Day	Meal	Price
Monday	pasta	\$6
Tuesday	chicken	\$8

Backslash escapes

Certain characters can be escaped with a preceding backslash to preserve the literal display of a character instead of its special Markdown meaning. This applies to the following characters:

- \ backslash
- ` backtick
- * asterisk
- _ underscore
- { } curly braces
- [] square brackets

() parentheses
 # hash mark
 > greater than
 + plus sign
 - minus sign (hyphen)
 . dot
 ! exclamation mark

README files

If your repository contains a `README.md` file at the root level, Bitbucket Server displays its contents on the repository's **Overview** page if the file has the `.md` extension. The file can contain [Markdown](#) and a restricted set of HTML tags.

Requesting add-ons

The [Atlassian Marketplace](#) website offers hundreds of add-ons that your administrator can install to enhance and extend Atlassian Bitbucket Server. If the add-on request feature is enabled for your Bitbucket Server instance, you can submit requests for add-ons from the Marketplace to your Bitbucket Server administrator.

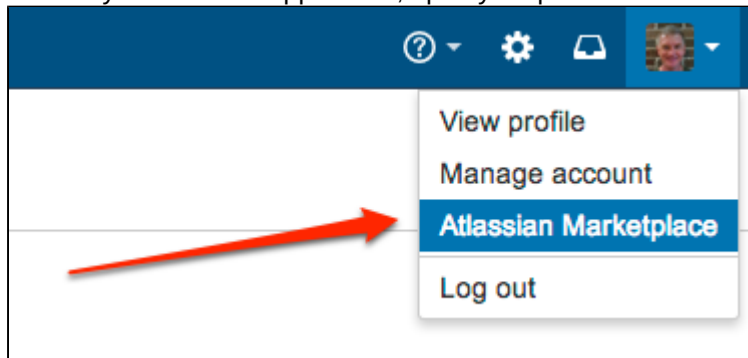
The 'Atlassian Marketplace for Bitbucket Server' page provides an integrated view of the Atlassian Marketplace from within your Bitbucket Server instance. The page offers the same features as the Marketplace website, such as searching and category filtering, but tailors the browsing experience to Bitbucket Server.

This in-product view of the Marketplace gives day-to-day Bitbucket Server users, not just administrators, an easy way to discover add-ons that can help them get work done. When you find an add-on of interest, you can submit a request to your administrator for the add-on with just a few clicks.

Submitting an add-on request

To browse for add-ons in the Atlassian Marketplace, follow these steps:

1. From anywhere in the application, open your profile menu and choose **Atlassian Marketplace**:



2. In the Atlassian Marketplace page, use the search box to find add-ons or use the category menus to browse or filter by add-ons by type, popularity, price or other criteria. You can see what your fellow users have requested by choosing the **Most Requested** filter.
3. When you find an add-on that interests you, click **Request** to generate a request for your administrator.
4. Optionally, type a personal message to your administrators in the text box. This message is visible to administrators in the details view for the add-on.
5. Click **Submit Request** when done.
6. Click **Close** to dismiss the 'Success!' message dialog box.

At this point, a notification appears in the interface your administrators use to administer add-ons. Also your request message will appear in the add-on details view, visible from the administrator's 'Find New Add-ons' page. From there, your administrator can purchase the add-on, try it out or dismiss requests.

Updating an add-on request

After submitting the request, you can update your message at any time. Click the **Update Request** button next to the listing in the Atlassian Marketplace page to modify the message to your administrator.

The administrator is not notified of the update. However, your updated message will appear as you have modified it in the details view for the add-on immediately.

Integrating Bitbucket Server with Atlassian applications

When you integrate Bitbucket Server with Atlassian applications you get the following benefits:

Application	Integration feature	Compatibility	
 	Related branches, commits and pull requests are all summarized in the Development panel in a JIRA issue.	JIRA 6.2+	Stash 2.10+
	Create Git branches from within JIRA and JIRA Agile.	JIRA 6.1+	Stash 2.8+
	Transition JIRA issues from within Bitbucket Server.	JIRA 5.0+	Stash 2.7+
	See the JIRA issues related to Bitbucket Server commits and pull requests.	JIRA 5.0+	Stash 2.1+
	See all the code changes committed for the issue (on the JIRA Source tab). Click through to see a changed file, or the full commit , in Bitbucket Server.	JIRA 5.0.4+	Plugin version bundled in JIRA
		JIRA 5.0–5.0.3	JIRA FishEye Plugin 5.0.4.1
		JIRA 4.4.x	JIRA FishEye Plugin 3.4.12
		JIRA 4.3.x	JIRA FishEye Plugin 3.1.8
	When Bitbucket Server is integrated with HipChat, notifications are sent to a HipChat room whenever someone pushes to a repository in Bitbucket Server.		Stash 2.2+

	Bamboo responds to repository events published by Bitbucket Server to: • Trigger a plan build when a developer pushes to the connected repository. • Create or delete plan branches when a developer creates or removes a branch in the connected repository. When you link a build plan to a Bitbucket Server repository, build notifications are automatically enabled. See Bamboo integration.	Bamboo 5.6+	Stash 3.1+
	See the latest build status for a commit when viewing Bitbucket Server commits and pull requests.	Bamboo 4.4+	Stash 2.1+
	When you have SourceTree installed, you can: • clone a Bitbucket Server repository using SourceTree. • check out a branch in SourceTree, when viewing files, commits or branches in a Bitbucket Server repository.	SourceTree 1.7+	Stash 2.7+
	When Bitbucket Server is integrated with Crowd, you can: • use Crowd for user and group management, and for authentication.		

JIRA integration

When Bitbucket Server is integrated with [JIRA](#) Software, you and your team get all these benefits:

- See all the [related commits, branches and pull requests](#) in an issue.
- [Create Git branches](#) from within JIRA Software.
- [Transition issues automatically](#).
- [Transition issues](#) from within Bitbucket Server.
- [Use issue keys](#) in Bitbucket Server markdown.
- [See the details for issues](#) in Bitbucket Server.
- [See issues related to Bitbucket Server commits and pull requests](#).
- Check commits, branches and pull requests [for an entire version within JIRA Software](#).

You can also use JIRA Software for delegated management of your Bitbucket Server users. See [External user directories](#).

Your Bitbucket Server administrator needs to set up [linking with JIRA Software](#) before you'll see these work.

Check development progress of a version in JIRA Software

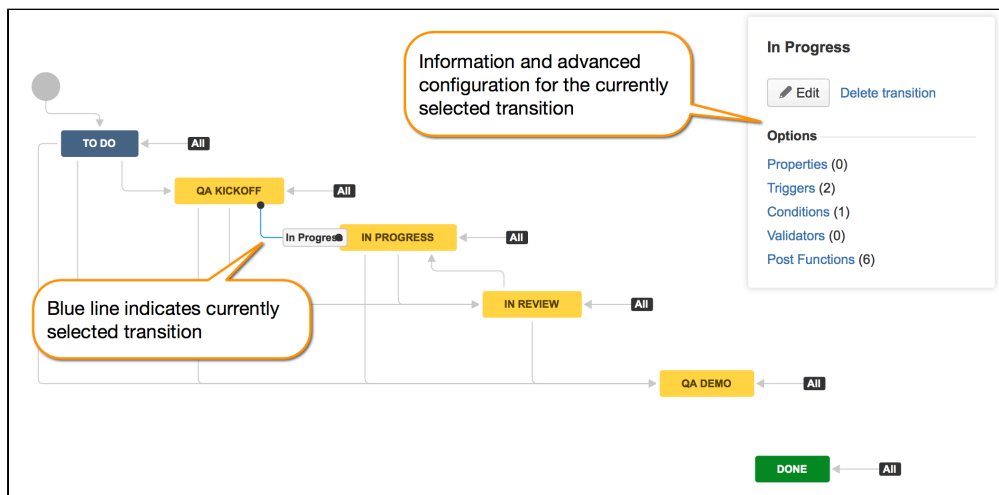
The Release Hub shows the progress of a version, so you can determine which issues are likely to ship at a glance. With JIRA Software and Bitbucket Server connected, the commits related to each issue are shown, helping you to spot potential development issues that could cause problems for a release.

When you are ready, you can also release the version from the Release Hub, which marks the version as complete, moves incomplete issues to other versions, and triggers release builds (if JIRA Software is connected to Bamboo).

To view the Release Hub (with the project sidebar enabled), navigate to a project, click on **Releases**, then select a version listed. See [Checking the progress of a version](#) for more detailed information about using the Release Hub in JIRA Software.

Transition issues automatically

Your workflow can now respond to events in your linked development tools. For example, when a pull request is created, your workflow can be configured to automatically transition the related issue. Configure this from transitions within the JIRA workflow editor – see [Advanced workflow configuration](#) in the JIRA Software documentation:



The events available in Bitbucket Server are:

- Branch created
- Commit created
- Pull request created
- Pull request merged
- Pull request declined

Bitbucket Server events are published by default. We recommend that you use the latest version of JIRA Software to ensure that duplicate events are handled correctly. JIRA Software automatically removes duplicate commit events and duplicate branch creation events.

See all related branches, commits and pull requests in an issue

Get visibility into the Bitbucket Server branches, commits and pull requests related to work on an issue, right in the context of the issue in JIRA Software.

Click the links in the Development panel to see details of the work that's been done. You can start creating a pull request from the Commits details dialog, or click through to see a changed file, or the full commit, in Bitbucket Server.

Create Git branches from within JIRA Software

You can start creating a branch from an issue. This gives you a faster workflow from picking an issue to starting coding.

Dates

Created: 03/Jul/15 3:21 PM

Updated: 04/Jul/15 2:32 PM

Development

Create branch

Bitbucket Server will suggest the branch type and branch name, based on the issue type and summary – you can change these, of course.

Transition issues from within Bitbucket Server

You can easily [transition](#) an issue from within Bitbucket Server. For example, when creating a pull request you may want to transition the issue into review. Click on a linked issue anywhere in Bitbucket Server to see a dialog with the available workflow steps:

JIRA Issues

 Teams in Space / TIS-58

Add feedback button to the plugin sample code

Reopen Code Review Failed

Details

Type:  Improvement

Priority:  Minor

Status: **CLOSED**

Assignee:  Kevin Campbell

Description

Click on a step and complete the fields as required. If there are custom fields that are unsupported by Bitbucket Server, just click **Edit this field in JIRA** to transition the issue directly in JIRA Software.

See issues from multiple instances of JIRA Software

Bitbucket Server can link to more than one JIRA Software server at a time, so different teams can work with their own projects in different instances, or a single team can link to issues across multiple JIRA Software servers. Read more about [linking Bitbucket Server with JIRA Software](#).

Use issue keys in markdown

When you mention an issue key in Bitbucket Server, for example in a pull request description or a comment, the key gets automatically linked:



Click on the linked key to see [details](#) for the issue.

See issue details

Click a linked issue key anywhere in Bitbucket Server to see the details of that issue in a dialog. And you can just click the issue key at the top of the dialog to go straight to the issue in JIRA Software:



See issues related to commits and pull requests

Bitbucket Server recognizes issue keys in commit messages, and displays the keys as links on the Commits tabs for both the repository and [pull requests](#):

Commits		
 master ▾ ...		
Author	Commit	Message
 Albert Wang	dbd10d0478b	Accidentally duplicated form-notification-test for Notification states changing, also fixed
 Albert Wang	ec66645907b 	Merged in AUI-2694-qunit-to-mocha (pull request #1010) Merging QUnit to Mocha epi
 Albert Wang	459716b5383 	AUI-2694 - fixed merge conflicts and added async test
 Trey Shugart	748db23bde1 	Merged in AUI-2751-skate-0-9 (pull request #1007) Upgrade to Skate 0.9.

Click on the linked key to see [details](#) for the issue.

Using Smart Commits in Bitbucket Server

Smart Commits allow repository committers to process JIRA Software issues using special commands in your commit messages.

You can:

- comment on issues
- record time tracking information against issues
- transition issues to any status defined in the JIRA Software project's workflow.

A single Smart Commit command cannot span more than one line (you cannot use carriage returns in the commit message), but you can add multiple commands to the same line, or multiple commands on separate lines. See [this example](#) below.

Smart Commits work with Bitbucket Server 4.2+ and JIRA Software 7.1+.

On this page:

- [Smart Commit commands](#)
- [Advanced examples](#)
- [Get Smart Commits working](#)

Related pages:

- [Enabling DVCS Smart Commits](#)

Smart Commit commands

The basic command line syntax for a smart commit message is:

```
<ignored text> <ISSUE_KEY> <ignored text> #<COMMAND> <optional
COMMAND_ARGUMENTS>
```

Any text between the issue key and the smart commit command is ignored.

There are three smart commit commands you can use in your commit messages:

- [comment](#)
- [time](#)
- [transition](#)

Comment

Description	Adds a comment to a JIRA Software issue.
Syntax	<code><ignored text> ISSUE_KEY <ignored text> #comment <comment_string></code>
Example	JRA-34 #comment corrected indent issue
Notes	• The committer's email address must match the email address of a single JIRA Software user with permission to comment on issues in that particular project.

Time

Description	Records time tracking information against an issue.
Syntax	<ignored text> ISSUE_KEY <ignored text> #time <value>w <value>d <value>h <value>m <comment_string>
Example	JRA-34 #time 1w 2d 4h 30m Total work logged
Notes	This example records 1 week, 2 days, 4hours and 30 minutes against the issue, and adds the comment 'Total work logged' in the Work Log tab of the issue. - Each value for w, d, h and m can be a decimal number. - The committer's email address must match the email address of a single JIRA Software user with permission to log work on an issue. - Your system administrator must have enabled time tracking on your JIRA Software instance.

Workflow transitions

Description	Transitions a JIRA Software issue to a particular workflow state.
Syntax	<ignored text> ISSUE_KEY <ignored text> #<transition_name> <comment_string>
Example	JRA-090 #close Fixed this today
Notes	This example executes the close issue workflow transition for the issue and adds the comment 'Fixed this today' to the issue. You can see the custom commands available for use with smart commits by visiting the JIRA Software issue and seeing its available workflow transitions: 1. Open an issue in the project. 2. Click View Workflow (near the issue's Status). The smart commit only considers the part of a transition name before the first space. So, for a transition name such as <code>finish work</code>, then specifying <code>#finish</code> is sufficient. You must use hyphens to replace spaces when ambiguity can arise over transition names, for example: <code>#finish-work</code>. If a workflow has two valid transitions, such as: • Start Progress • Start Review A smart commit with the action <code>#start</code> is ambiguous because it could mean either of the two transitions. To specify one of these two transitions, fully qualify the transition you want by using either <code>#start-review</code> or <code>#start-progress</code>. • When you resolve an issue with the <code>#resolve</code> command, you cannot set the Resolution field with smart commits. • If you want to add a comment during the transition, the transition must have a screen associated with it. • The committer's email address must match the email address of a single JIRA Software user with the appropriate project permissions to transition issues.

Advanced examples

Multiple commands over multiple lines on a single issue

Syntax	<code><ISSUE_KEY> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS></code>
Commit message	JRA-123 #comment Imagine that this is a really, and I mean really, long comment #time 2d 5h
Result	Adds the comment 'This is a really, and I' (but drops the rest of the comment) and logs 2 days and 5 hours of work against issue JRA-123.

Multiple commands on a single issue

Syntax	<code><ISSUE_KEY> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS></code>
Commit message	JRA-123 #time 2d 5h #comment Task completed ahead of schedule #resolve
Result	Logs 2 days and 5 hours of work against issue JRA-123, adds the comment 'Task completed ahead of schedule', and resolves the issue.

A single command on multiple issues

Syntax	<code><ISSUE_KEY1> <ISSUE_KEY2> <ISSUE_KEY3> #<COMMAND> <optional COMMAND_ARGUMENTS> etc</code>
Commit message	JRA-123 JRA-234 JRA-345 #resolve
Result	Resolves issues JRA-123, JRA-234 and JRA-345. Multiple issue keys must be separated by whitespace or commas.

Multiple commands on multiple issues

Syntax	<code><ISSUE_KEY1> <ISSUE_KEY2> ... <ISSUE_KEYn> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS></code>
Commit message	JRA-123 JRA-234 JRA-345 #resolve #time 2d 5h #comment Task completed ahead of schedule
Result	Logs 2 days and 5 hours of work against issues JRA-123, JRA-234 and JRA-345, adds the comment 'Task completed ahead of schedule' to all three issues, and resolves all three issues. Multiple issue keys must be separated by whitespace or commas.

Get Smart Commits working

Smart Commits work with Bitbucket Server 4.2+ and JIRA Software 7.1+.

To get Smart Commits working for JIRA Software and Bitbucket Server:

1. Create an application link between JIRA Software and Bitbucket Server. See [Linking Bitbucket Server](#)

with JIRA.

2. Enable smart commits in JIRA Software. See [Enabling DVCS Smart Commits](#).

Some limitations of Smart Commits:

- Smart Commits only support the default JIRA Software issue key format (that is, two or more uppercase letters, followed by a hyphen and the issue number, for example BAM-123).
- Smart Commits don't provide for field-level updates in JIRA Software issues.

HipChat notifications

Bitbucket Server can send a notifications to a HipChat room for activities performed in a Bitbucket Server project or repository, and it works with HipChat Server or Cloud. [Learn more about HipChat](#).

What kind of notifications can I get?

The HipChat integration lets Bitbucket Server send the following notifications to your HipChat rooms:

- Pull requests—when they are created, commented, merged, and declined
- Commits—when they are pushed and commented

Here's an example of what you might see (from an Atlassian HipChat room):

Bitbucket	🔗 Branch bundle-HipChat-integration created by William in Bitbucket/bitbucket-server
Bitbucket	🔗 One commit pushed to branch bundle-HipChat-integration by William in Bitbucket/bitbucket - BSERV-7845 Add tests for HipChat integration · 3fa9dcfb558
Bitbucket	🔗 Pull request Bundle HipChat integration created by William in Bitbucket/bitbucket-server
Bitbucket	🔗 Pull request Bundle HipChat integration commented on by Alana in Bitbucket/bitbucket-server Looks good to me! See more
Bitbucket	🔗 Pull request Bundle HipChat integration merged by William in Bitbucket/bitbucket-server

On this page

- [What kind of notifications can I get?](#)
- [Install and configure the HipChat for Bitbucket Server integration](#)
- [Enable HipChat notifications for a repository](#)
- [I'm already using the old HipChat hook](#)

Install and configure the HipChat for Bitbucket Server integration

To enable Bitbucket Server to send notifications in HipChat you must connect Bitbucket Server to your HipChat instance. In order to do this you need administration privileges for your Bitbucket Server instance and for HipChat.

To integrate Bitbucket Server and HipChat

1. Select **Administration Settings**



> **HipChat integration**

2. If using *HipChat Cloud*, click the **Connect HipChat** button, which takes you to HipChat.com sign up page.
If using *HipChat Server*, click the link below the Connect HipChat button, enter your HipChat server URL, then click **Connect HipChat**.
3. Log in to HipChat with an account that has admin rights.
4. Click **Install** to finish installing the Bitbucket Server HipChat Addon.
5. Select the repository that you want to send notifications from, and the HipChat room where you want the notifications to appear.
 - You can choose multiple rooms to receive notifications from a repository but must add each connection separately.

- Repeat this process for all the repositories where you want to send notifications.

In HipChat, you will see a notification in your room telling you that notifications were enabled.

Bitbucket  Notifications from the repository **Bitbucket/bitbucket-server** will now appear in this room

Enable HipChat notifications for a repository

Once HipChat and Bitbucket Server are [integrated](#), you can enable and disable HipChat notifications for a particular repo by going to **Settings > HipChat**. You must have admin privileges for the project or repository you want to enable notifications for.

Initially, you will need to click the **Log in to HipChat** button to connect your Bitbucket Server and HipChat user accounts. If you don't see the **Log in to HipChat** button you will need to talk to your Bitbucket Server administrator and get them to install the integration first.

Now you can add room links between your repository and rooms in HipChat. Search for the room and press **Add** to get things going.

From repository: bitbucket-server



Development

Add

You can change the notification types by unticking some of the checkboxes.

From repository: bitbucket-server → Select HipChat room Add

Repository	Room	
bitbucket-server	Development	Done ×

Notify room with activity in these areas:

Pull requests	☒ Created	Commits	☒ Pushed
	☒ Commented		☒ Commented
	☒ Merged		
	☒ Declined		

I'm already using the old HipChat hook

There is a previous integration for Bitbucket Server and HipChat which has been deprecated. It will continue working until Bitbucket Server version 4.0. However, you are encouraged to upgrade to the latest version.

Once you are ready to setup the new integration make sure you disable all your configured old integrations.

1. Go to each repository that has the old hook configured.
2. Click **Disabled** to disable the old integration.

Post receive - perform actions after commits are processed

 **HipChat Push Notification (deprecated)** 

Sends a notice to the specified HipChat room whenever someone pushes to the repository.

Disabled Enabled

3. Setup the new integration by following the steps on this page

Bamboo integration

When you integrate [Bitbucket Server](#) with Atlassian's [Bamboo](#) build and deployment server, commit, branch, build and deployment information is shared for users of both applications.

On this page:

- [Benefits of integration](#)
- [Configuration](#)

Benefits of integration

When Bamboo (versions 5.6 and later) and Bitbucket Server are integrated, you and your team get all the following advantages:

Bitbucket Server tells Bamboo when to build

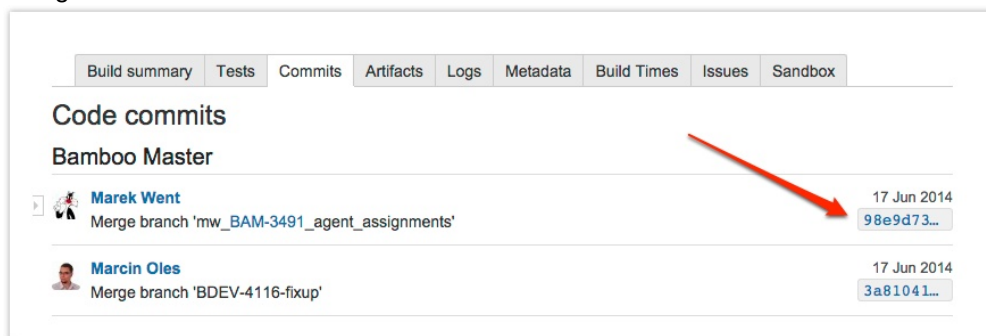
- When a developer pushes to a repository the build is automatically started.

Bitbucket Server tells Bamboo when to update plan branches to match changes in repository branches

- When a developer pushes a new branch to a repository a branch plan is automatically created.
- When a developer deletes a branch in a repository, the branch plan is automatically deleted or disabled.

Bitbucket Server commits are displayed in the relevant Bamboo builds

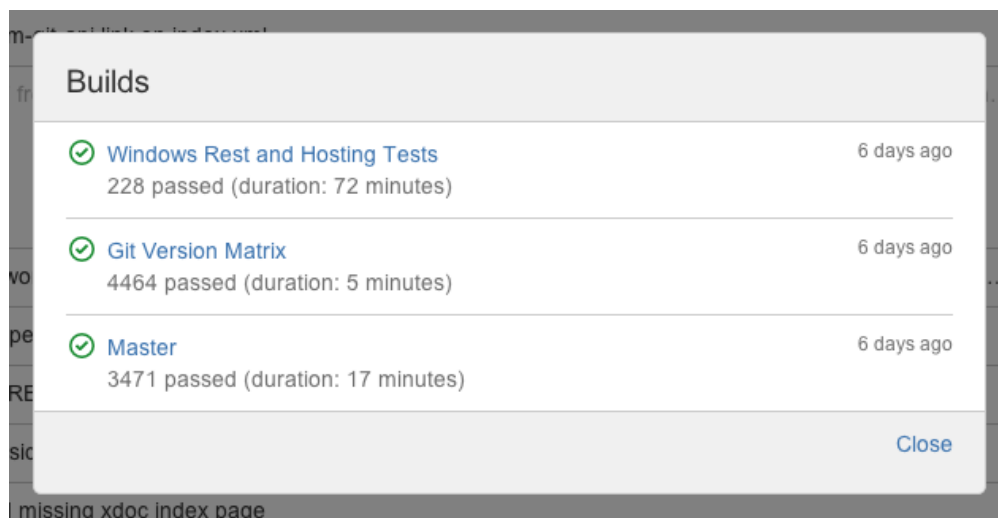
- In Bamboo, you can view all of the commits involved in the build, allowing you to accurately track changes:



- Simply click on a changeset to go to Bitbucket Server, where you can see the commit diff for all of the files that are part of the build.

Bamboo notifies Bitbucket Server automatically about build results

- Build notifications are automatically enabled when you link a build plan to a Bitbucket Server repository.
- Notifications are sent to all linked Bitbucket servers.
- You see the build results status for a commit when viewing any commit or pull request in Bitbucket Server, so you can easily check the build status of a branch when deciding whether to merge changes.
- Click a build status icon in Bitbucket Server to see further details:



Bitbucket Server displays the overall status of the build results. The status is 'passed' if all the different builds (for example, unit tests, functional tests, deploy to staging) have succeeded, and 'failed' if at least one run failed for any of those.

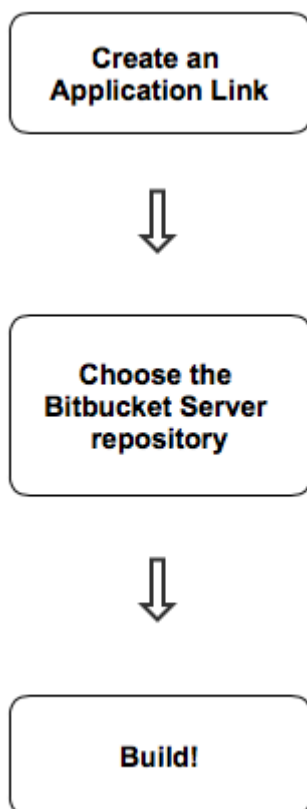
For example, when viewing the Commits tab for a Bitbucket Server project, you will see icons that indicate the status of the latest build results. The red 'fail' icon is displayed if there is at least one failed build run for the commit.

Note that the legacy Bitbucket Server notification type is deprecated – it is still available in Bamboo 5.6 but will be removed in Bamboo 5.7.

Configuration

There are just a few simple configuration steps to get the integrations described above with Bamboo (versions 5.6 and later) and Bitbucket Server.

Bamboo will be automatically configured to respond to repository events published by Bitbucket Server, and to notify Bitbucket Server about build results – you don't have to configure repository polling for new commits anymore in Bamboo, or set up dedicated web hooks in your Bitbucket Server instance.



1. Create an Application Link

You only need to do this once for each pair of Bitbucket Server and Bamboo instances.

See [Linking to another application](#).

Once linked, all the Bitbucket Server repositories are available to your plans in Bamboo.

2. Choose the Bitbucket Server repository for the Bamboo plan

Create a build plan (if necessary) and specify the repository in the plan (or job) configuration.

To connect to a Bitbucket Server repository, select **Bitbucket Server / Stash** and provide the Bitbucket Server details.

You must [enable the SSH access](#) on Bitbucket Server, otherwise the integration features won't work and you will have to provide an alternative HTTP repository type to connect to the Bitbucket Server repository.

BAM-15464 - Provide HTTP(S) authentication method option for Stash type repository
[OPEN](#)

See [Bitbucket Server](#) for more information about using Bitbucket Server source repositories in Bamboo.

3. Build!

You can also use the Bitbucket Server Rest API to automatically publish build status from Bamboo, Jenkins or any other build tool to Bitbucket Server. See the Bitbucket Server developer documentation to do with [updating build status](#).

Set the default time zone in Bitbucket Server

You can adjust the default time zone in Bitbucket Server individually or for all users. Setting time zones gives remote users and distributed teams accurate, local timestamps in the application and notification emails. However, setting the default time zone for all users won't override individual user's time zone settings.

To set the default time zone for all users

1. Log in to Bitbucket Server as an admin.
2. From the administration area, select **Server settings** (under 'Settings').
3. In the **Time Zone** field, specify the appropriate time zone.

To set the default time zone for a single user (yourself)

1. Click your profile picture (in the upper-right), then select **Manage account**.
2. In the **Time Zone** field, specify the appropriate time zone.

Use repository tags

Git supports annotated or lightweight tagging. Git stores annotated tags as full objects in the Git repository. As objects, annotated tags include the name, email, date, and have a message. You can sign an annotated tag and Git verifies it with GNU Privacy Guard (GPG). Lightweight tags don't include all this additional information. [You can also create tags from within the UI of Bitbucket Server.](#)

Create tags from the command line

To create an annotated tag, enter the following:

```
$ git tag -a 1.0 -m 'the initial release'
```

To create a lightweight tag, enter the following:

```
$ git tag 'tempTag'
```

A regular push command won't push a tag, to push all your tags:

```
git push origin --tags
```

To push a single tag:

```
git push origin : tempTag
```

To list the tags in a repo, enter the following:

```
$ git tag
1.0
tempTag
```

Create tags within Bitbucket Server

You can also create tags within Bitbucket Server. While viewing a single commit, select + (in the upper-right) next to the tag icon, then add the basic information for the new tag.

Administering Bitbucket Server

Administration actions that can be performed from the Bitbucket Server Administration user interface (click the 'cog' icon in the Bitbucket Server header):

- [Supported platforms](#)
- [Users and groups](#)
- [External user directories](#)
 - [Connecting Bitbucket Server to an existing LDAP directory](#)
 - [Connecting Bitbucket Server to JIRA for user management](#)
 - [Delegating Bitbucket Server authentication to an LDAP directory](#)
 - [Connecting Bitbucket Server to Crowd](#)
- [Global permissions](#)
- [Setting up your mail server](#)
- [Linking Bitbucket Server with JIRA](#)
- [Connecting Bitbucket Server to an external database](#)
- [Migrating Bitbucket Server to another server](#)
- [Specifying the base URL for Bitbucket Server](#)
- [Configuring the application navigator](#)
- [Managing add-ons](#)
- [Audit logging in Bitbucket Server](#)
- [Updating your Bitbucket Server license details](#)

System administration advanced actions that can be performed from outside of the Bitbucket Server user interface:

- [Running the Bitbucket Server installer](#)
- [Automated setup for Bitbucket Server](#)
- [Starting and stopping Bitbucket Server](#)
- [Install Bitbucket Server from an archive file](#)
- [Install and configure a remote Elasticsearch instance](#)
- [Running Bitbucket Server as a Linux service](#)
- [Running Bitbucket Server as a Windows service](#)
- [Bitbucket Server config properties](#)
- [Proxying and securing Bitbucket Server](#)
- [Enabling SSH access to Git repositories in Bitbucket Server](#)
- [Using diff transcoding in Bitbucket Server](#)
- [Changing the port that Bitbucket Server listens on](#)
- [Moving Bitbucket Server to a different context path](#)
- [Running Bitbucket Server with a dedicated user](#)
- [Bitbucket Server debug logging](#)
- [Data recovery and backups](#)
- [Lockout recovery process](#)
- [Scaling Bitbucket Server](#)
- [High availability for Bitbucket Server](#)
- [Clustering with Bitbucket Data Center](#)
- [Enabling JMX counters for performance monitoring](#)
- [Getting started with Bitbucket Server and AWS](#)
- [Disabling HTTP\(S\) access to Git repositories in Bitbucket Server](#)
- [Smart Mirroring](#)
- [Git Large File Storage](#)
- [Copy of Bitbucket Server config properties](#)

Users and groups

Bitbucket Server comes with an internal user directory already built-in that is enabled by default at installation. When you create the first administrator during the setup procedure, that administrator's username and other details are stored in the internal directory.

Bitbucket Server Admins and Sys Admins can manage users and groups in Bitbucket Server as described on this page. You can also set up Bitbucket Server to [use external user directories](#).

Note that:

- Even after users have been added to the Bitbucket Server user directory, they will not be able to log in to Bitbucket Server until they have been given [global access permissions](#).
- Permissions can also be applied separately at the level of [projects](#), [repositories](#) and [branches](#).

On this page

- [Creating a user](#)
- [Creating a group](#)
- [Adding users to groups](#)
 - [From the user account page](#)
 - [From the group page](#)
- [Changing usernames](#)
- [Deleting users and groups](#)

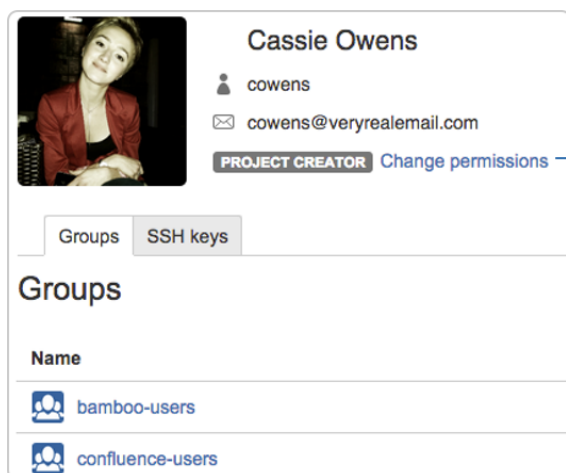
Related pages:

- [Getting started with Bitbucket Server](#)
- [External user directories](#)

Creating a user

To create a user:

1. In the administration area, click **Users** (under 'Accounts') and then **Create user** (on the 'Users' screen).
2. Complete the form. You can either set the user's password now, or have Bitbucket Server email the user with a link that they can use to set the password themselves:
3. Once you've created the user, click **Change permissions** to set up their access permissions. Note that a user doesn't have access to Bitbucket Server until global access permissions have been set.



Set up user permissions

See [Global permissions](#) for more information.

Creating a group

To create a group, from the administration area:

1. Click **Groups** (under 'Accounts') and then **Create group**.
2. Enter the name for the new group, and click **Create group** (again):

Create group

Group name ^{*}

- Now you can add users to your new group (see the next section).

Adding users to groups

You can add users to groups in two ways:

- add a particular user to multiple groups, [from the user's account page](#) in the admin area.
- add multiple users to a particular group, [from the group's page](#).

From the user account page

To add a user to a group from the user's account page,

- Click **Users** in the Administration section, and then use the filter to find the user:

Users

Name	Username	Email
Admin Istrator	admin	tis
Alana Grant	agrant	ag
Cassie Owens	cowens	co
Emma Paris	eparis	ep

User search
Filter users by name or email as you type.

- On the account page for the user, use the filter to find a group to which you want to add the user.
- Click **Add** for each group in turn.

From the group page

To add a user to a group from the group's page,

- Click **Groups** (under "Accounts") in the administration area, and use the filter to find the group:
- On the page for the group, use the filter to find a user to add to the group.
- Click **Add** for each user you select, to make them a member of the group.

Changing usernames

You can change the username for a user account that is hosted in Bitbucket Server's internal user directory.

To change a user's username:

- Go to **Users** in the Administration section, use the filter to find the user.
- On the account page for the user, click **Rename**.

Deleting users and groups

You can delete a user or group from Bitbucket Server's internal user directory, or the external directory from which Bitbucket Server sources users, such as an LDAP, Crowd or JIRA Software.

When a user or group is deleted from such a directory, Bitbucket Server checks to see if that user still exists in another directory:

- If the user or group *does* exist in another directory, Bitbucket Server assumes the administrator intended to *migrate* the user or group between directories and we leave their data intact.
- If the user or group *does not* exist in another directory, Bitbucket Server assumes the intent was to permanently delete them, and we delete the users permissions, SSH keys and 'rememberme' tokens.

Notes

- If an entire directory is deleted Bitbucket Server *always* assumes it is a migration and does nothing to clean up after users and groups.
- Content which might be of historical interest (comments, pull requests, etc.) is not deleted when a user or group is. Only authentication, authorization and data which serves no purpose to a user who can no longer log in is removed.
- In some situations, reordering the directories will change the directory that the current user comes from, if a user with the same username happens to exist in both. This behavior can be used in some cases to create a copy of the existing configuration, move it to the top, then remove the old one. Note, however, that duplicate usernames are not a supported configuration.
- You can enable or disable a directory at any time. If you disable a directory, your configuration details will remain but Bitbucket Server will not recognize the users and groups in that directory.

Limitations

- You cannot edit, disable or delete the directory that your own user account belongs to. This prevents administrators from locking themselves out of Bitbucket Server, and applies to internal as well as external directories.
- You cannot remove the internal directory. This limitation aligns with the recommendation that you always keep an administrator or sysadmin account active in the Bitbucket Server internal directory, so that you can troubleshoot problems with your user directories.
- You have to disable a directory before you can remove it. Removing a directory will remove the details from the database.

External user directories

You can connect Bitbucket Server to external user directories. This allows you to use existing users and groups stored in an enterprise directory, and to manage those users and groups in one place.

User management functions include:

- **Authentication:** determining which user identity is sending a request to Bitbucket Server.
- **Authorization:** determining the access privileges for an authenticated user.
- **User management:** maintaining profile information in user's accounts.
- **Group membership:** storing and retrieving groups, and group membership.

It is important to understand that these are separate components of a user management system. You could use an external directory for any or all of the above tasks.

There are several approaches to consider when using external user directories with Bitbucket Server, described briefly below:

- [LDAP](#)
- [JIRA applications](#)

On this page

- [LDAP](#)
- [JIRA applications](#)
- [Crowd](#)
- [Multiple directories](#)

Related pages

- [Connecting Bitbucket Server to an existing LDAP directory](#)
- [Delegating Bitbucket Server authentication to an LDAP directory](#)
- [Connecting Bitbucket Server to Crowd](#)
- [Connecting Bitbucket Server to JIRA for user management](#)
- [Users and groups](#)
- [External directory lockout recovery](#)

- [Crowd](#)
- [Multiple directories](#)

- Bitbucket Server provides a "read-only" connection to external directories for user management. This means that users and groups, fetched from *any external directory*, can only be modified or updated in the external directory itself, rather than in Bitbucket Server.
- Connecting Atlassian Bitbucket Server to your external directory is not sufficient to allow your users to log in to Bitbucket Server. You must explicitly grant them access to Bitbucket Server in the [global permission screen](#).
- We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Bitbucket Server license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket Server will display a warning banner. See [this FAQ](#).
- Bitbucket Server comes with an internal user directory, already built-in, that is enabled by default at installation. When you create the first administrator during the setup procedure, that administrator's username and other details are stored in the internal directory.
- See also this [information about deleting users and groups](#) in Bitbucket Server.

LDAP

You should consider connecting to an LDAP directory server if your users and groups are stored in an enterprise directory.

There are two common ways of using an external LDAP directory with Bitbucket Server:

- For full user and group management, including for user authentication — see [Connecting Bitbucket Server to an existing LDAP directory](#) for instructions.
- For delegated user authentication only, while using Bitbucket Server's internal directory for user and group management — see [Delegating Bitbucket Server authentication to an LDAP directory](#) for instructions.

Bitbucket Server is able to connect to the following LDAP directory servers:

- Microsoft Active Directory
- Apache Directory Server (ApacheDS) 1.0.x and 1.5.x
- Apple Open Directory (Read-Only)
- Fedora Directory Server (Read-Only Posix Schema)
- Novell eDirectory Server
- OpenDS
- OpenLDAP
- OpenLDAP (Read-Only Posix Schema)
- Generic Posix/RFC2307 Directory (Read-Only)
- Sun Directory Server Enterprise Edition (DSEE)
- Any generic LDAP directory server

JIRA applications

You can delegate Bitbucket Server user and group management, as well as user authentication, to a JIRA application. This is a good option if you already use a JIRA application in your organization. Note that Bitbucket Server can only connect to a JIRA application server running JIRA 4.3 or later.

You should consider using [Atlassian Crowd](#) for more complex configurations with a large number of users.

See [Connecting Bitbucket Server to JIRA for user management](#) for configuration instructions.

Crowd

You can connect Bitbucket Server to [Atlassian Crowd](#) for user and group management, as well as for user authentication.

Crowd is an application security framework that handles authentication and authorization for your web-based

applications. With Crowd you can integrate multiple web applications with multiple user directories, with support for single sign-on (SSO) and centralized identity management. See the [Crowd Administration Guide](#).

You should consider connecting to Crowd if you want to use Crowd to manage existing users and groups in multiple directory types, or if you have users of other web-based applications.

See [Connecting Bitbucket Server to Crowd](#) for configuration instructions.

Multiple directories

When Bitbucket Server is connected directly to multiple user directories, where duplicate user names and group names are used across those directories, the effective group memberships that Bitbucket Server uses for authorization can be determined using either of these two schemes:

- 'aggregating membership'
- 'non-aggregating membership'.

See [Effective memberships with multiple directories](#) for more information about these two schemes.

Note that:

- Aggregating membership is used by default for new installations of Bitbucket Server.
- Authentication, for when Bitbucket Server is connected to multiple directories, only depends on the mapped groups in those directories – the aggregation scheme is not involved at all.
- For inactive users, Bitbucket Server only checks if the user is active in the first (highest priority) directory in which they are found for the purpose of determining authentication. Whether a user is active or inactive does not affect how their memberships are determined.
- When a user is added to a group, they are only added to the first writeable directory available, in priority order.
- When a user is removed from a group, they are only removed from the group in the first directory the user appears in, when non-aggregating membership is used. With aggregating membership, they are removed from the group in *all* directories the user exists in.

A Bitbucket Server admin can change the membership scheme used by Bitbucket Server using the following commands:

- To change to *aggregating membership*, substitute your own values for <username>, <password> and <base-url> in this command:

```
curl -H 'Content-type: application/json' -X PUT -d
'{"membershipAggregationEnabled":true}' -u <username>:<password>
<base-url>/rest/crowd/latest/application
```

- To change to *non-aggregating membership*, substitute your own values for <username>, <password> and <base-url> in this command:

```
curl -H 'Content-type: application/json' -X PUT -d
'{"membershipAggregationEnabled":false}' -u
<username>:<password> <base-url>/rest/crowd/latest/application
```

Note that these operations are different from how you make these changes in Crowd. Note also that changing the aggregation scheme can affect the authorization permissions for your Bitbucket Server users, and how directory update operations are performed.

Connecting Bitbucket Server to an existing LDAP directory

You can connect Bitbucket Server to an existing LDAP user directory, so that your existing users and groups in an enterprise directory can be used in Bitbucket Server. The LDAP directory is used for both user authentication and account management.

Bitbucket Server is able to connect to the following LDAP directory servers:

- Microsoft Active Directory
- Apache Directory Server (ApacheDS) 1.0.x and 1.5.x
- Apple Open Directory (Read-Only)
- Fedora Directory Server (Read-Only Posix Schema)
- Novell eDirectory Server
- OpenDS
- OpenLDAP
- OpenLDAP (Read-Only Posix Schema)
- Generic Posix/RFC2307 Directory (Read-Only)
- Sun Directory Server Enterprise Edition (DSEE)
- Any generic LDAP directory server

On this page

- [License considerations](#)
- [Synchronization when Bitbucket Server is first connected to the LDAP directory](#)
- [Authentication when a user attempts to log in](#)
- [Connecting Bitbucket Server](#)
- [Server settings](#)
- [LDAP schema](#)
- [LDAP permission](#)
- [Advanced settings](#)
- [User schema settings](#)
- [Group schema settings](#)
- [Membership schema settings](#)

Related pages

- [Deleting Users and Groups](#)

Connecting Atlassian Bitbucket Server to your external directory is not sufficient to allow your users to log in to Bitbucket Server. You must explicitly grant them access to Bitbucket Server in the [global permission screen](#).

We recommend that you use groups instead of individual accounts when granting permissions.

License considerations

When connecting Bitbucket Server to an external directory, be careful not to allow access to Bitbucket Server by more users than your Bitbucket Server license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket Server will display a warning banner. See [this FAQ](#).

Synchronization when Bitbucket Server is first connected to the LDAP directory

When you first connect Bitbucket Server to an existing LDAP directory, the Bitbucket Server internal directory is synchronized with the LDAP directory. User information, including groups and group memberships, is copied across to the Bitbucket Server directory.

When we performed internal testing of synchronization with an Active Directory server on our local network with 10 000 users, 1000 groups and 200 000 memberships, we found that the initial synchronization took about 5 minutes. Subsequent synchronizations with 100 modifications on the AD server took a couple of seconds to complete. See the [option](#) below.

Note that when Bitbucket Server is connected to an LDAP directory, you cannot update user details in Bitbucket Server. Updates must be done directly on the LDAP directory, perhaps using a LDAP browser tool such as [Apache Directory Studio](#).

Option - Use LDAP filters to restrict the number of users and groups that are synchronized

You can use LDAP filters to restrict the users and groups that are synchronized with the Bitbucket Server internal directory. You may wish to do this in order to limit the users or groups that can access Bitbucket Server, or if you are concerned that synchronization performance may be poor.

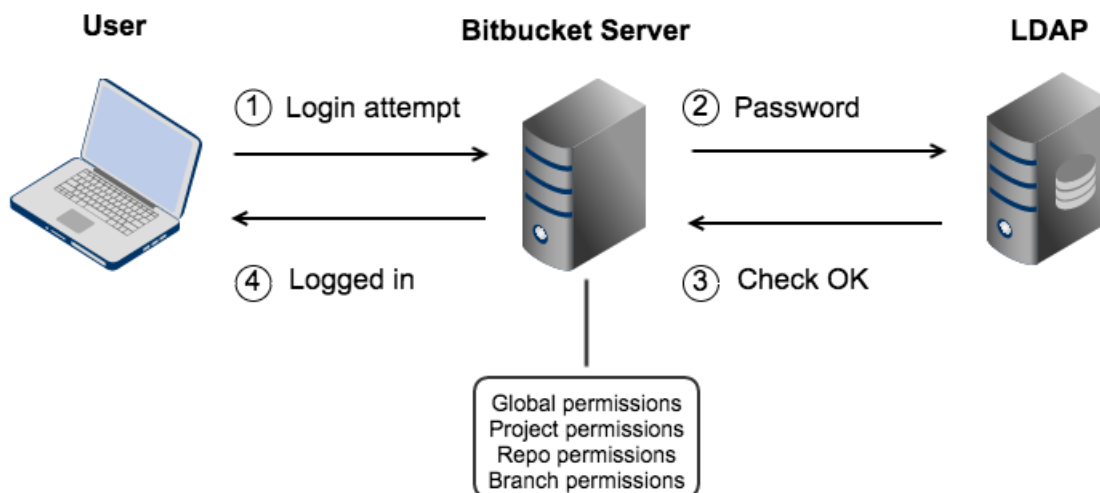
For example, to limit synchronization to just the groups named "bitbucket_user" or "red_team", enter the following into the **Group Object Filter** field (see [Group Schema Settings](#) below):

```
(&(objectClass=group)(|(cn=bitbucket_user)(cn=red_team)))
```

For further discussion about filters, with examples, please see [How to write LDAP search filters](#). Note that you need to know the names for the various containers, attributes and object classes in your particular directory tree, rather than simply copying these examples. You can discover these container names by using a tool such as [Apache Directory Studio](#).

Authentication when a user attempts to log in

When a user attempts to log in to Bitbucket Server, once synchronization has completed, Bitbucket Server confirms that the user exists in its internal directory and then passes the user's password to the LDAP directory for confirmation. If the password matches that stored for the user, LDAP passes a confirmation back to Bitbucket Server, and Bitbucket Server logs in the user. During the user's session, all authorizations (i.e. access to Bitbucket Server resources such as repositories, pull requests and administration screens) are handled by Bitbucket Server, based on permissions maintained by Bitbucket Server in its internal directory.



Connecting Bitbucket Server

To connect Bitbucket Server to an LDAP directory:

1. Log in as a user with 'Admin' permission.
2. In the Bitbucket Server administration area, click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select either **Microsoft Active Directory** or **LDAP** as the directory type.
4. Configure the directory settings, as described in the tables below.
5. Save the directory settings.
6. Define the directory order by clicking the arrows next to each directory on the 'User Directories' screen. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Server settings

Setting	Description
Name	Enter a meaningful name to help you identify the LDAP directory server. Examples: • Example Company Staff Directory • Example Company Corporate LDAP

Directory Type	Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for many of the options on the rest of screen. Examples: • Microsoft Active Directory • OpenDS • And more.
Hostname	The host name of your directory server. Examples: • <code>ad.example.com</code> • <code>ldap.example.com</code> • <code>opens.example.com</code>
Port	The port on which your directory server is listening. Examples: • 389 • 10389 • 636 (for example, for SSL)
Use SSL	Check this if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.
Username	The distinguished name of the user that the application will use when connecting to the directory server. Examples: • <code>cn=administrator,cn=users,dc=ad,dc=example,dc=com</code> • <code>cn=user,dc=domain,dc=name</code> • <code>user@domain.name</code> Ensure that this is an administrator user for the LDAP engine. For example, in Active Directory the user will need to be a member of the built-in Administrators group. The specific privileges for the LDAP user that is used to connect to LDAP are 'bind' and 'read' (user info, group info, group membership, update sequence number, deleted objects). Admin privileges are required because a normal user can't access the <code>uSNC</code> attribute and deleted objects container, causing incremental sync to fail silently. This has been reported as CWD-3093.
Password	The password of the user specified above. Note: Connecting to an LDAP server requires that this application log in to the server with the username and password configured here. As a result, this password cannot be one-way hashed - it must be recoverable in the context of this application. The password is currently stored in the database in plain text without obfuscation. To guarantee its security, you need to ensure that other processes do not have OS-level read permissions for this application's database or configuration files.

LDAP schema

Setting	Description
Base DN	The root distinguished name (DN) to use when running queries against the directory server. Examples: • <code>o=example,c=com</code> • <code>cn=users,dc=ad,dc=example,dc=com</code> • For Microsoft Active Directory, specify the base DN in the following format: <code>dc=domain1,dc=local</code>. You will need to replace the <code>domain1</code> and <code>local</code> for your specific configuration. Microsoft Server provides a tool called <code>ldp.exe</code> which is useful for finding out and configuring the the LDAP structure of your server.

Additional User DN	This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example: ou=Users
Additional Group DN	This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example: ou=Groups

If no value is supplied for **Additional User DN** or **Additional Group DN** this will cause the subtree search to start from the base DN and, in case of huge directory structure, could cause performance issues for login and operations that rely on login to be performed.

LDAP permission

Setting	Description
Read Only	LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens.
Read Only, with Local Groups	LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens. However, you can add groups to the internal directory and add LDAP users to those groups.

Advanced settings

The **Manage User Status Locally** option, described below, will not work within Bitbucket Server. Do not enable this option.

BSERV-5129 - Disable "Manage User Status Locally" in Bitbucket Server

[OPEN](#)

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called 'nested groups'. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Manage User Status Locally	If true, you can activate and deactivate users in Crowd independent of their status in the directory server.
Filter out expired users	If true, user accounts marked as expired in ActiveDirectory will be automatically removed. For cached directories, the removal of a user will occur during the first synchronization after the account's expiration date.
Use Paged Results	Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size – that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results.

Follow Referrals	Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup <code>java.naming.referral</code>) configuration setting. It is generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.
Naive DN Matching	If your directory server will always return a consistent string representation of a DN, you can enable naive DN matching. Using naive DN matching will result in a significant performance improvement, so we recommend enabling it where possible. This setting determines how your application will compare DN's to determine if they are equal. • If this checkbox is selected, the application will do a direct, case-insensitive, string comparison. This is the default and recommended setting for Active Directory, because Active Directory guarantees the format of DN's. • If this checkbox is not selected, the application will parse the DN and then check the parsed version.
Enable Incremental Synchronization	Enable incremental synchronization if you only want changes since the last synchronization to be queried when synchronizing a directory.  Please be aware that when using this option, the user account configured for synchronization must have read access to: • The <code>uSNChanged</code> attribute of all users and groups in the directory that need to be synchronized. • The objects and attributes in the Active Directory deleted objects container (see Microsoft's Knowledge Base Article No. 892806 for details). If at least one of these conditions is not met, you may end up with users who are added to (or deleted from) the Active Directory not being respectively added (or deleted) in the application. This setting is only available if the directory type is set to "Microsoft Active Directory".
Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.
Read Timeout (seconds)	The time, in seconds, to wait for a response to be received. If there is no response within the specified time period, the read attempt will be aborted. A value of 0 (zero) means there is no limit. The default value is 120 seconds.
Search Timeout (seconds)	The time, in seconds, to wait for a response from a search operation. A value of 0 (zero) means there is no limit. The default value is 60 seconds.
Connection Timeout (seconds)	This setting affects two actions. The default value is 0. • The time to wait when getting a connection from the connection pool. A value of 0 (zero) means there is no limit, so wait indefinitely. • The time, in seconds, to wait when opening new server connections. A value of 0 (zero) means that the TCP network timeout will be used, which may be several minutes.

User schema settings

Setting	Description

User Object Class	This is the name of the class used for the LDAP user object. Example: • user
User Object Filter	The filter to use when searching user objects. Example: • (&(objectCategory=Person)(sAMAccountName=*)) More examples can be found here and here.
User Name Attribute	The attribute field to use when loading the username. Examples: • cn • sAMAccountName NB: In Active Directory, the 'sAMAccountName' is the 'User Logon Name (pre-Windows 2000)' field. The User Logon Name field is referenced by 'cn'.
User Name RDN Attribute	The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example: • cn
User First Name Attribute	The attribute field to use when loading the user's first name. Example: • givenName
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: • sn
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: • displayName
User Email Attribute	The attribute field to use when loading the user's email address. Example: • mail
User Password Attribute	The attribute field to use when loading a user's password. Example: • unicodePwd
User Unique ID Attribute	The attribute used as a unique immutable identifier for user objects. This is used to track username changes and is optional. If this attribute is not set (or is set to an invalid value), user renames will not be detected — they will be interpreted as a user deletion then a new user addition. This should normally point to a UUID value. Standards-compliant LDAP servers will implement this as 'entryUUID' according to RFC 4530. This setting exists because it is known under different names on some servers, e.g. 'objectGUID' in Microsoft Active Directory.

Group schema settings

Setting	Description

Group Object Class	This is the name of the class used for the LDAP group object. Examples: • groupOfUniqueNames • group
Group Object Filter	The filter to use when searching group objects. Example: • (&(objectClass=group)(cn=*))
Group Name Attribute	The attribute field to use when loading the group's name. Example: • cn
Group Description Attribute	The attribute field to use when loading the group's description. Example: • description

Membership schema settings

Setting	Description
Group Members Attribute	The attribute field to use when loading the group's members. Example: • member
User Membership Attribute	The attribute field to use when loading the user's groups. Example: • memberOf
Use the User Membership Attribute, when finding the user's group membership	Check this if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.) • If this checkbox is selected, your application will use the group membership attribute on the user when retrieving the list of groups to which a given user belongs. This will result in a more efficient retrieval. • If this checkbox is not selected, your application will use the members attribute on the group ('member' by default) for the search. • If the Enable Nested Groups checkbox is selected, your application will ignore the Use the User Membership Attribute option and will use the members attribute on the group for the search.
Use the User Membership Attribute, when finding the members of a group	Check this if your directory server supports the user membership attribute on the group. (By default, this is the 'member' attribute.) • If this checkbox is selected, your application will use the group membership attribute on the user when retrieving the members of a given group. This will result in a more efficient search. • If this checkbox is not selected, your application will use the members attribute on the group ('member' by default) for the search.

Connecting Bitbucket Server to JIRA for user management

 *This page does not apply to JIRA Software Cloud; you can't use JIRA Software Cloud to manage your Bitbucket Server users.*

You can connect Bitbucket Server to an existing Atlassian JIRA Software instance to delegate Bitbucket Server user and group management, and authentication. Bitbucket Server provides a "read-only" connection to JIRA Software for user management. This means that users and groups, fetched from JIRA Software, can only be modified or updated in that JIRA Software server, rather than in Bitbucket Server.

Choose this option, as an alternative to Atlassian Crowd, for simple configurations with a limited number of users. Note that Bitbucket Server can only connect to an instance running JIRA Software 4.3 or later.

Connecting Bitbucket Server and JIRA Software is a 3-step process:

1. Set up JIRA Software to allow connections from Bitbucket Server
2. Set up Bitbucket Server to connect to JIRA Software
3. Set up Bitbucket Server users and groups in JIRA Software

Also on this page:

- [Server settings](#)
- [JIRA Software server permissions](#)
- [Advanced settings](#)

 You need to be an administrator in JIRA Software and a system administrator in Bitbucket Server to perform the following tasks.

1. Setup JIRA Software to allow connections from Bitbucket Server

1. Log in as a user with the 'JIRA Software Administrators' global permission.
2. For JIRA 4.3.x, select **Other Application** from the 'Users, Groups & Roles' section of the 'Administration' menu.
For later versions, choose **Administration > Users > JIRA User Server**.
3. Click **Add Application**.
4. Enter the **application name** (case-sensitive) and **password** that Bitbucket Server will use when accessing JIRA Software.
5. Enter the **IP address** of your Bitbucket Server instance. Valid values are:
 - A full IP address, e.g. 192.168.10.12.
 - A wildcard IP range, using CIDR notation, e.g. 192.168.10.1/16. For more information, see the introduction to [CIDR notation on Wikipedia](#) and [RFC 4632](#).
6. Click **Save**.
7. Define the directory order, on the 'User Directories' screen, by clicking the blue up- and down-arrows next to each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

2. Setup Bitbucket Server to connect to JIRA Software

1. Log in to Bitbucket Server as a user with 'Admin' permission.
2. In the Bitbucket Server administration area click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Atlassian JIRA**.
4. Enter settings, as described below.
5. Test and save the directory settings.
6. Define the directory order, on the 'User Directories' screen, by clicking the arrows for each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

3. Set up Bitbucket Server users and groups in JIRA Software

In order to use Bitbucket Server, users must be a member of the `Bitbucket Server-users` group or have Bitbucket Server global permissions. Follow these steps to configure your Bitbucket Server groups in JIRA Software:

1. Add the `bitbucket-users` and `bitbucket-administrators` groups in JIRA Software.
2. Add your own username as a member of both of the above groups.
3. Choose one of the following methods to give your existing JIRA Software users access to Bitbucket Server:

- Option 1: In JIRA Software, find the groups that the relevant users belong to. Add those groups as members of one or both of the above Bitbucket Server groups.
- Option 2: Log in to Bitbucket Server using your JIRA Software account and go to the administration area. Click **Global permissions** (under 'Accounts'). Assign the appropriate permissions to the relevant JIRA Software groups. See [Global permissions](#).

Connecting Atlassian Bitbucket Server to JIRA Software for user management is not sufficient, by itself, to allow your users to log in to Bitbucket Server. You must also grant them access to Bitbucket Server by using one of the above 2 options.

We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Bitbucket Server license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket Server will display a warning banner. See [this FAQ](#).

See also this [information about deleting users and groups](#) in Bitbucket Server.

Server settings

Setting	Description
Name	A meaningful name that will help you to identify this JIRA server in the list of directory servers. Examples: • JIRA Service Desk Server • My Company JIRA
Server URL	The web address of your JIRA server. Examples: • http://www.example.com:8080 • http://jira.example.com
Application Name	The name used by your application when accessing the JIRA server that acts as user manager. Note that you will also need to define your application to that JIRA server, via the ' Other Applications ' option in the 'Users, Groups & Roles' section of the 'Administration' menu.
Application Password	The password used by your application when accessing the JIRA server that acts as user manager.

JIRA Software server permissions

Setting	Description
Read Only	The users, groups and memberships in this directory are retrieved from the JIRA server that is acting as user manager. They can only be modified via that JIRA server.

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Before enabling nested groups, please check to see if nested groups are enabled on the JIRA server that is acting as user manager. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.

Enable Incremental Synchronization	Enable or disable incremental synchronization. Only changes since the last synchronization will be retrieved when synchronizing a directory..
Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.

Delegating Bitbucket Server authentication to an LDAP directory

You can configure Bitbucket Server to use an LDAP directory for delegated user authentication while still using Bitbucket Server for user and group management.

You can either create new user accounts manually in the LDAP directory, or use the option to automatically create a user account when the user attempts to log in, as described in the [Copy users on login](#) section below.

See also this [information about deleting users and groups](#) in Bitbucket Server.

To connect Bitbucket Server to an LDAP directory for delegated authentication:

1. Log in to Bitbucket Server as a user with 'Admin' permission.
2. Go to the Bitbucket Server administration area and click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Internal with LDAP Authentication** as the directory type.
4. Configure the directory settings, as described in the tables below.
5. Save the directory settings.
6. Define the directory order by clicking the arrows for each directory on the 'User Directories' screen. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Connecting Atlassian Bitbucket Server to your external directory is not sufficient to allow your users to log in to Bitbucket Server. You must explicitly grant them access to Bitbucket Server in the [global permission screen](#).

We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Bitbucket Server license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket Server will display a warning banner. See [this FAQ](#).

On this page:

- [Server settings](#)
- [Manually creating users](#)
- [Copying users on login](#)
- [LDAP schema](#)
- [Advanced settings](#)
- [User schema settings](#)
- [Group schema settings](#)
- [Membership schema settings](#)

Server settings

Setting	Description

Name	A descriptive name that will help you to identify the directory. Examples: • Internal directory with LDAP Authentication • Corporate LDAP for Authentication Only
Directory Type	Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for some of the options on the rest of screen. Examples: • Microsoft Active Directory • OpenDS • And more.
Hostname	The host name of your directory server. Examples: • ad.example.com • ldap.example.com • opens.example.com
Port	The port on which your directory server is listening. Examples: • 389 • 10389 • 636 (for example, for SSL)
Use SSL	Check this box if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.
Username	The distinguished name of the user that the application will use when connecting to the directory server. Examples: • cn=admin, cn=users, dc=ad, dc=example, dc=com • cn=user, dc=domain, dc=name • user@domain.name
Password	The password of the user specified above.

Manually creating users

Move the delegated authentication directory to the top of the User Directories list and create the user manually (go to **Administration > Users > Create user**). Using this manual method you must currently create a temporary password when creating users. There is an improvement request to address this:

BSERV-3424 - Disable "Change password" field from admin and user page when delegated authentication is used

CLOSED

If you intend to *change* the authentication directory of your users from Bitbucket Server Internal Directory to Delegated LDAP Authentication you must select the option to "Copy User on Login" since you can't create a new user that has the same username as another user in another directory.

Copying users on login

The settings described in the table below relate to when a user attempts to authenticate with Bitbucket Server. This authentication attempt can occur either:

- when using the Bitbucket Server login screen.
- when issuing a Git clone or push command at the command line, for a repository managed by Bitbucket Server.

Setting	Description

Copy User on Login	This option affects what will happen when a user attempts to log in. If this box is checked, the user will be created automatically in the internal directory that is using LDAP for authentication when the user first logs in and their details will be synchronized on each subsequent log in. If this box is not checked, the user's login will fail if the user wasn't already manually created in the directory. If you check this box the following additional fields will appear on the screen, which are described in more detail below: • Default Group Memberships • Synchronize Group Memberships • User Schema Settings (described in a separate section below)
Default Group Memberships	This field appears if you check the Copy User on Login box. If you would like users to be automatically added to a group or groups, enter the group name(s) here. To specify more than one group, separate the group names with commas. Each time a user logs in, their group memberships will be checked. If the user does not belong to the specified group(s), their username will be added to the group(s). If a group does not yet exist, it will be added to the internal directory that is using LDAP for authentication. Please note that there is no validation of the group names. If you mis-type the group name, authorization failures will result – users will not be able to access the applications or functionality based on the intended group name. Examples: • confluence-users • bamboo-users,jira-administrators,jira-core-users
Synchronize Group Memberships	This field appears if you select the Copy User on Login checkbox. If this box is checked, group memberships specified on your LDAP server will be synchronized with the internal directory each time the user logs in. If you check this box the following additional fields will appear on the screen, both described in more detail below: • Group Schema Settings (described in a separate section below) • Membership Schema Settings (described in a separate section below)

LDAP schema

Setting	Description
Base DN	The root distinguished name (DN) to use when running queries against the directory server. Examples: • o=example,c=com • cn=users,dc=ad,dc=example,dc=com • For Microsoft Active Directory, specify the base DN in the following format: dc=domain1,dc=local. You will need to replace the domain1 and local for your specific configuration. Microsoft Server provides a tool called <code>ldp.exe</code> which is useful for finding out and configuring the the LDAP structure of your server.
User Name Attribute	The attribute field to use when loading the username. Examples: • cn • sAMAccountName

Advanced settings

Setting	Description

Enable Nested Groups	Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called 'nested groups'. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Use Paged Results	Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size – that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results.
Follow Referrals	Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup <code>java.naming.referral</code>) configuration setting. It is generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.

User schema settings

Note: this section is only visible when **Copy User on Login** is enabled.

Setting	Description
Additional User DN	This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example: ou=Users
User Object Class	This is the name of the class used for the LDAP user object. Example: user
User Object Filter	The filter to use when searching user objects. Example: (& (objectCategory=Person) (sAMAccountName=*))
User Name RDN Attribute	The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example: cn
User First Name Attribute	The attribute field to use when loading the user's first name. Example: givenName
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: sn
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: displayName
User Email Attribute	The attribute field to use when loading the user's email address. Example: mail

Group schema settings

Note: this section is only visible when both **Copy User on Login** and **Synchronize Group Memberships** are enabled.

Setting	Description
Additional Group DN	This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example: ou=Groups
Group Object Class	This is the name of the class used for the LDAP group object. Examples: - groupOfUniqueNames - group
Group Object Filter	The filter to use when searching group objects. Example: (objectCategory=Group)
Group Name Attribute	The attribute field to use when loading the group's name. Example: cn
Group Description Attribute	The attribute field to use when loading the group's description. Example: description

Membership schema settings

Note: this section is only visible when both **Copy User on Login** and **Synchronize Group Memberships** are enabled.

Setting	Description
Group Members Attribute	The attribute field to use when loading the group's members. Example: member
User Membership Attribute	The attribute field to use when loading the user's groups. Example: memberOf
Use the User Membership Attribute, when finding the user's group membership	Check this box if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.) - If this box is checked, your application will use the group membership attribute on the user when retrieving the members of a given group. This will result in a more efficient retrieval. - If this box is not checked, your application will use the members attribute on the group ('member' by default) for the search.

Connecting Bitbucket Server to Crowd

You can configure Bitbucket Server to use Atlassian Crowd for user and group management, and for authentication and authorization.

Atlassian Crowd is an application security framework that handles authentication and authorization for your web-based applications. With Crowd you can integrate multiple web applications and user directories, with support for single sign-on (SSO) and centralized identity management. See the [Crowd Administration Guide](#).

Connect to Crowd if you want to use Crowd to manage existing users and groups in multiple directory types, or if you have users of other web-based applications.

See also this [information about deleting users and groups](#) in Bitbucket Server.

Connecting Atlassian Bitbucket Server to your external directory is not sufficient to allow your users to log in to Bitbucket Server. You must explicitly grant them access to Bitbucket Server in the [global permission screen](#).

We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Bitbucket Server license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket Server will display a warning banner. See [this FAQ](#).

On this page:

- [Server settings](#)
- [Crowd permissions](#)
- [Advanced settings](#)
- [Single sign-on \(SSO\) with Crowd](#)
- [Using multiple directories](#)

To connect Bitbucket Server to Crowd:

1. Log in as a user with 'Admin' permission.
2. In the Bitbucket Server administration area, click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Atlassian Crowd**.
4. Enter settings, as described below.
5. Test and save the directory settings.
6. Define the directory order, on the **Directories** tab, by clicking the blue up- and down-arrows next to each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Server settings

Setting	Description
Name	A meaningful name that will help you to identify this Crowd server amongst your list of directory servers. Examples: • Crowd Server • Example Company Crowd
Server URL	The web address of your Crowd console server. Examples: • http://www.example.com:8095/crowd/ • http://crowd.example.com
Application Name	The name of your application, as recognized by your Crowd server. Note that you will need to define the application in Crowd too, using the Crowd administration Console. See the Crowd documentation on adding an application .
Application Password	The password which the application will use when it authenticates against the Crowd framework as a client. This must be the same as the password you have registered in Crowd for this application. See the Crowd documentation on adding an application .

Crowd permissions

Bitbucket Server offers **Read Only** permissions for Crowd directories. The users, groups and memberships in Crowd directories are retrieved from Crowd and can only be modified from Crowd. You cannot modify Crowd users, groups or memberships using the Bitbucket Server administration screens.

For local Bitbucket Server directories, **Read Only** and **Read/Write** permissions are available.

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Before enabling nested groups, please check to see if the user directory or directories in Crowd support nested groups. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.

Single sign-on (SSO) with Crowd

Once the Crowd directory has been set up, you can enable Crowd SSO integration by adding the following setting to `shared/bitbucket.properties` in the [Bitbucket Server home directory](#) (create this file if it doesn't exist yet):

```

bitbucket.properties
# Whether SSO support should be enabled or not. Regardless of this
# setting SSO authentication
# will only be activated when a Crowd directory is configured in
# Bitbucket Server that is configured
# for SSO.
plugin.auth-crowd.sso.enabled=true

```

Please note that you will need to correctly set up the domains of the applications involved in SSO. See [Crowd SSO Domain examples](#).

In addition to this property, Crowd SSO integration can be tuned using the system properties described on [Bitbucket Server config properties](#).

Using multiple directories

When Bitbucket Server is connected to Crowd you can map Bitbucket Server to multiple user directories in Crowd.

For Crowd 2.8, and later versions, there are two different membership schemes that Crowd can use when multiple directories are mapped to an integrated application, and duplicate user names and group names are used across those directories. The schemes are called 'aggregating membership' and 'non-aggregating membership' and are used to determine the effective group memberships that Bitbucket Server uses for *authorization*. See [Effective memberships with multiple directories](#) for more information about these two schemes in Crowd.

Note that:

- *Authentication*, for when Bitbucket Server is mapped to multiple directories in Crowd, only depends on the mapped groups in those directories – the aggregation scheme is not involved at all.
- For inactive users, Bitbucket Server only checks if the user is active in the first (highest priority) directory in which they are found to determine *authentication*. The membership schemes described above are not used when Crowd determines if a user should have access to Bitbucket Server.
- When a user is added to a group, they are only added to the first writeable directory available, in priority order.
- When a user is removed from a group, they are only removed from the group in the first directory the user appears in, when non-aggregating membership is used. With aggregating membership, they are removed

from the group in *all* directories the user exists in.

An administrator can set the aggregation scheme that Bitbucket Server uses when integrated with Crowd. Go to the **Directories** tab for the Bitbucket Server instance in Crowd, and check **Aggregate group memberships across directories** to use the 'aggregating membership' scheme. When the checkbox is clear 'non-aggregating membership' is used.

Note that changing the aggregation scheme can affect the authorization permissions for your Bitbucket Server users, and how directory update operations are performed.

Global permissions

Bitbucket Server uses four levels of account permissions to control user and group access to Bitbucket Server projects and to the Bitbucket Server instance configuration.

Related pages:

- [Users and groups](#)
- [Using project permission](#)

User accounts that have not been assigned "Bitbucket Server User" permission or higher, either directly or through group membership, will not be able to log in to Bitbucket Server. These users are considered unlicensed and do not count towards your Bitbucket Server license limit.

You can also [apply access permissions to projects](#).

A user's permission level is displayed on the user's page seen from the admin area.



Admin Istrator

 admin

 tisadmin@veryrealemail.com

SYS. ADMIN [Change permissions](#)

	Login / Browse	Create projects	Manage users / groups	Manage global permissions	Edit application settings	Edit server config
Bitbucket User	✓	✗	✗	✗	✗	✗
Project Creator	✓	✓	✗	✗	✗	✗
Admin	✓	✓	✓	✓	✓	✗
System Admin	✓	✓	✓	✓	✓	✓

To edit the account permissions for an existing Bitbucket Server user or group:

1. Click the 'cog' menu in the header, to go to the admin area.
2. Click **Global permissions** (under 'Accounts').
3. Select, or clear, the permission checkboxes as required.
4. Click in the **Add Users** or **Add Groups** field to set permissions for additional users or groups.

You can remove all permissions for a user or group by clicking the X at the right-hand end of the row (when you hover there). This will remove that user or group.

Global Permissions

User access

Name	System Admin	Admin	Project Creator	Bitbucket User
				System Admin Add
Admin Istrator	☒	☒	☒	☒

Group access

Name	System Admin	Admin	Project Creator	Bitbucket User
Add Groups				Admin Add
bitbucket-admins	☐	☒	☒	☒
bitbucket-users	☐	☐	☐	☒
jira-administrators	☒	☒	☒	☒

Setting up your mail server

Setting up Bitbucket Server to use your SMTP mail server:

- allows Bitbucket Server to send [notifications](#) about events to do with pull requests. See [Using pull requests in Bitbucket Server](#). Note that if the mail server fails, notifications will be dropped.
- allows Bitbucket Server to email a link to a newly created user, which the user can use to generate their own password.
- allows a user to reset his or her password if they forget it.

To configure a mail server for Bitbucket Server, go to the administration area and click **Mail server** (under 'Settings'). See [Supported platforms](#) for the mail clients supported by Bitbucket Server.

Complete the form and click **Save**.

Hostname	The hostname of the mail server (for example "localhost" or "192.168.1.15").
Port	The port of the mail server (if unspecified, the port 25 will be used).
Username	The username to use to connect to the mail server.
Password	The password to use to connect to the mail server.
Protocol	Use either SMTP or SMTPS when connecting to the mail server. When using SMTP, you can specify that: • SSL/TLS is used if supported by the mail server, otherwise mail is sent in plaintext. • mail should only be sent if the mail server supports SSL/TLS. See Securing email notifications below.
Use SSL/TLS if available	If the SMTP server supports the STARTTLS extension this will be used to encrypt mail with SSL/TLS otherwise plaintext will be used. SMTPS servers always support SSL/TLS
Always use SSL/TLS	If the SMTP server does not support the STARTTLS extension mail will not be sent. SMTPS servers always support SSL/TLS
Email from	Specifies the 'From' header in notification emails (for example: noreply@yourcompany.com).

Send a test email

Enter an email address to send a test email to check that the mail server is configured correctly.

Anonymous user

If you wish to set up the outgoing mail server as an anonymous user, simply leave the username and password fields empty. However, in Chrome, these fields may be auto-populated, leading to an error – as a workaround, try using a different browser.

Mail server configuration

Mail settings

Hostname*
The hostname of the mail server (for example "localhost" or "192.168.1.15")

Port
The port of the mail server (if unspecified, the port 25 will be used). Typically port 25 or 587 for SMTP and 465 for SMTPS

Username
The username to use to connect to the mail server

Password
The password to use to connect to the mail server

Protocol
Select the protocol to use when connecting to the mail server

☒ Use SSL/TLS if available
If the SMTP server supports the STARTTLS extension this will be used to encrypt mail with SSL/TLS otherwise plaintext will be used. SMTPS servers always support SSL/TLS

☐ Always use SSL/TLS
If the SMTP server does not support the STARTTLS extension mail will not be sent. SMTPS servers always support SSL/TLS

Email from*
Specifies the From: header in notification emails (for example: noreply@yourcompany.com)

Send a test email

Recipient
The email address to send the test message to

Securing email notifications

Bitbucket Server 3.6 and later versions support the following protocols:

- SMTP, where mail is not encrypted.
- SMTP encrypted by SSL/TLS using the STARTTLS extension, where the protocol conversation is upgraded only if SSL/TLS is supported by the mail server, but otherwise remains as plaintext.
- SMTP, where STARTTLS support is required on the mail server, otherwise mail is not sent.

- SMTPS (where the whole protocol conversation uses SSL/TLS).

Note that if you use either SMTP with STARTTLS, or SMTPS, and connect to a self-signed mail server, you may need to import the server's certificate and set up a custom cacerts file for Bitbucket Server (just as you do for any outbound SSL/TLS connection to a self-signed server). See this [Bitbucket Server knowledge base article](#) for information about how to do that.

Configuring the mail server to use Gmail

If you wish to connect to a Gmail account for email notifications in Bitbucket Server, refer to the [Configuring the Mail Server to Use Gmail](#) guide.

In particular, note that Gmail won't show images in the email because of the way that Google loads images on their servers. For Google Apps, a Bitbucket Server administrator can solve the problem by adding the Bitbucket Server domain name to a whitelist – see <https://support.google.com/a/answer/3299041?hl=en> for more information.

Linking Bitbucket Server with JIRA

See [JIRA integration](#) for a description of all the integrations you get when Bitbucket Server is linked with JIRA Software.

You can also use JIRA Software for delegated user management. See [External user directories](#).

This page describes how to link Bitbucket Server to JIRA Software.

Link Bitbucket Server with JIRA Software

On this page

- [Link Bitbucket Server with JIRA Software](#)
- [Update an existing link to use OAuth](#)
- [Restrictions for JIRA Software integration](#)
- [Link Bitbucket Server with JIRA Software Cloud](#)
- [Troubleshoot integration with JIRA Software](#)

You can integrate Bitbucket Server with one or more instances of JIRA Software by means of 'application links'. You set up application links either:

- during the Bitbucket Server install process, using the [Setup Wizard](#), or
- at any time after installation, as described below.

To link Bitbucket Server to a JIRA Software server:

1. Click **Application Links** (under 'Settings') in the Bitbucket Server admin area.
2. Enter the URL for the JIRA Software instance you want to link to and click **Create new link**.
3. Complete the application link wizard to connect Bitbucket Server to your JIRA Software server. You must make use of the automatic link-back from JIRA Software to Bitbucket Server to get full integration (you'll need system administrator global permission for that).

Note that:

- Atlassian only recommends using OAuth authentication for application links, because of the greater security inherent with that protocol. We no longer recommend the Trusted Applications and Basic Access authentication types.
- When Bitbucket Server 4.0 or later is linked with JIRA Software 6.2 or later, you won't see the Source tab at the bottom of the View Issue screen any more.
- Bitbucket Server only begins scanning commit messages for issue keys on the first push after you created the application link to JIRA Software – the scan may take a short time.
- The following [system plugins](#) must be enabled in Bitbucket Server. These are bundled and enabled by default in Bitbucket Server 4.0 (and later):
 - Atlassian Navigation Links Plugin (com.atlassian.plugins.atlassian-nav-links-plugin)
 - Bitbucket Server Dev Summary Plugin (bitbucket-jira-development-integration-plugin).

See [Link Atlassian applications to work together](#) for more details.

Update an existing link to use OAuth

You may need to update an existing application link to use OAuth authentication when:

- you upgrade an Atlassian application to a version that uses version 5.2, or later, of application links. See the [Application links version matrix](#).
- the existing link uses Trusted Applications authentication, but your team can't see summary information from a developer tool such as Bitbucket Server in the Development panel in JIRA Software issues.
- an existing application link uses OAuth, but your team can't see the details dialogs for the Development panel in JIRA Software issues.
- you use a plugin that requires the OAuth authentication type.

Here's how to do that in JIRA Software, but the process is much the same for other Atlassian server products:

Begin in the local application

Go to the 'Configure Application Links' page in the admin area of the local application.

You may see a **DEPRECATED** lozenge beside links that need to be updated.

Click the pencil icon on the right to edit the configuration for the link you are updating.

In the 'Edit' dialog, set the local authentication for the link under 'Connections':

Edit - Confluence **DEPRECATED**

Troubleshoot

Remote application

Application name: Confluence

Application URL: http://teamsinspace.com:8090

Display URL: http://teamsinspace.com:8090

Connections

Direction	Local authentication	Remote authentication
Outgoing	OAuth (impersonation)	Confluence OAuth (impersonation) Troubleshoot
Incoming	OAuth (impersonation)	Confluence OAuth (impersonation) Troubleshoot

Save changes Cancel

Choose either:

- **OAuth** can be used where the applications have different user bases.
- **OAuth (impersonation)** if your users have the same user account and use the same password on each application.

See [OAuth security for application links](#) for more information.

Make sure that the authentication matches for the local and remote ends of both the incoming and outgoing directions.

Click **Save changes**.

Now, in the remote application

Go to the 'Configure Application Links' page in the admin area of the remote application. Choose the instructions column here that matches the UI you see (they both achieve the same result):

Click the pencil icon on the right to edit the configuration for the link you are updating.

In the 'Edit' dialog, set the local authentication for the link under 'Connections':

Edit - Confluence DEPRECATED

Troubleshoot

Remote application

Application name: Confluence

Application URL: http://teamsinspace.com:8090

Display URL: http://teamsinspace.com:8090

Connections

Direction	Local authentication	Remote authentication
Outgoing	OAuth (impersonation)	→ Confluence OAuth (impersonation) Troubleshoot
Incoming	OAuth (impersonation)	← Confluence OAuth (impersonation) Troubleshoot

Save changes Cancel

Choose either:

- **OAuth** can be used where the applications have different user bases.
- **OAuth (impersonation)** if your users have the same user account and use the same password on each application.

See [OAuth security for application links](#) for more information.

Make sure that the authentication matches for the local and remote ends of both the incoming and outgoing directions.

Click **Save changes**.

Click **Edit** for the application link you are updating.

In the 'Configure' dialog, click **Outgoing Authentication** and then the **OAuth** tab:

Configure Stash

Application Details
Outgoing Authentication
Incoming Authentication

This application Bitbucket (Bitbucket Server) can be configured to make requests to Stash (Bitbucket Server) using these authentication methods:

Trusted Applications OAuth Basic Access

Status **Configured**

Disable

☒ Enable outgoing 2-Legged OAuth requests ?

☐ Enable outgoing 2-Legged OAuth with Impersonation requests ?

Update

Close

Now, select **Enable 2-Legged OAuth**, assuming that the applications have different user bases, so that integration features will work.

Optionally, select **Enable 2-Legged OAuth with impersonation**, if your users have the same user account and use the same password on each application.

Click **Update**.

Now, click **Incoming Authentication** and then the **OAuth** tab:

Configure Stash

Application Details
Outgoing Authentication
Incoming Authentication

This application Bitbucket (Bitbucket Server) can be configured to allow incoming requests from Stash (Bitbucket Server) using these authentication methods:

Trusted Applications OAuth Basic Access

Status **Configured**

Disable

☒ Allow 2-Legged OAuth ?

Execute as

☐ Allow user impersonation through 2-Legged OAuth ?

This option allows 2-Legged OAuth requests to impersonate any user. It should allowed only in links with fully trusted application.

Update

Close

Now, select **Enable 2-Legged OAuth**, assuming that the applications have different user bases, so that integration features will work.

Optionally, select **Enable 2-Legged OAuth with impersonation**, if your users have the same user account and use the same password on each application.

Click **Update**.

Note that:

- Users who can see summarized data in the JIRA Software Development panel may not have permission to see all the information that contributed to those summaries and that is visible in the

details dialogs (for example, for branches, commits and pull requests). That is, the details dialogs respect the access permissions that users have in the connected applications.

- Your team members must have the 'View Development Tools' permission in JIRA Software to see the Development panel for an issue.
- If you run an application on port 443, you must use a valid SSL certificate (which is not self-signed) to get the full functionality available.

See [OAuth security for application links](#) for more details.

Restrictions for JIRA Software integration

- The display of [details for JIRA Software issues](#), for example when viewing a pull request, relies on the JIRA 5.0 REST API. Issue details are not displayed when Bitbucket Server is integrated with JIRA Software versions earlier than 5.0.
- Transitioning issues requires OAuth authentication. If only Basic Access authentication is used for the application link, users will be able to view issue details, but will not be able to transition issues.
- JIRA Software permissions are respected, so a user who is not permitted to transition an issue will not see the transition buttons in Bitbucket Server.
- If Bitbucket Server is linked with multiple JIRA Software instances and the projects happen to have the same key, only the issue from the instance marked as **PRIMARY** will be displayed. See [Making a primary link for links to the same application type](#).

Link Bitbucket Server with JIRA Software Cloud

There are port restrictions, and other considerations, when linking Bitbucket Server with JIRA Software Cloud.

Your local server must use a valid SSL certificate, and it must be accessible on port 80 or 443. For more information, see [this Atlassian Cloud documentation](#).

If you have a internet-facing firewall, make sure to allow the IP range used by Atlassian to reach your internal network. For up-to-date information on that, see [Database and IP information](#).

Troubleshoot integration with JIRA Software

There are a few situations where the integration of Bitbucket Server with JIRA Software can produce an error or may not function as expected:

Unable to see the Development panel within an issue

You must have the 'View Development Tools' permission in JIRA Software to see the Development panel. See [Managing Global Permissions](#).

You don't have permission to access the project

If you don't have permission to access the project within JIRA Software then Bitbucket Server will be unable to display issues.

The JIRA Software server is of an unsupported version

Bitbucket Server can integrate with JIRA 4.3.x, or later. Some features require higher versions of JIRA Software to function properly. See [Integrating Bitbucket Server with Atlassian applications](#) for details.

The issue key is invalid

Bitbucket Server doesn't check for invalid issue keys, such as 'UTF-8'. An error will result if Bitbucket Server tries to connect to an issue that doesn't exist. See this issue:

BSERV-2470 - JIRA Integration: Check for issue validity before linking issues

[OPEN](#)

The issue keys are of a custom format

Bitbucket Server assumes that issue keys are of the default format (that is, two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example TEST-123). By default, Bitbucket Server will not recognize custom issue key formats. See [Using custom JIRA issue keys with Bitbucket Server](#) for details.

The application link is created with OAuth only without the option to create a link using Trusted Applications

Bitbucket Server allows a user with global permissions of "Administrator" to create an OAuth only application link. You need to log in with a user having "System Administrator" privileges to create an application link using Trusted Applications authentication.

Still having problems?

See [Troubleshooting JIRA Software Integration](#) for more information specifically related to JIRA Software and Bitbucket Server.

Having trouble integrating your Atlassian products with application links?

We've developed a [guide to troubleshooting application links](#), to help you out. Take a look at it if you need a hand getting around any errors or roadblocks with setting up application links.

Using custom JIRA issue keys with Bitbucket Server

Bitbucket Server assumes that JIRA Software issue keys are of the default format (that is, two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example TEST-123). By default, Bitbucket Server will not recognize custom issue key formats.

You can use custom issue key formats with Bitbucket Server, however note that integrations with JIRA Software can depend on using the default issue key format in both applications. See [Integrating using custom JIRA Software issue keys](#) for more details.

Configure Bitbucket Server to recognize custom issue key formats by editing <Bitbucket Server installation directory>/bin/setenv.sh (on Windows, edit <Bitbucket Server installation directory>/bin/setenv.bat instead).

To override the default issue key format, use the JVM_SUPPORT_RECOMMENDED_ARGS property, like this:

Bitbucket Server and Stash 2.8 or later:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-Dintegration.jira.key.pattern=\"(<Some
different regex>)\""
```

You'll need to restart Bitbucket Server.

For example, to use lowercase letters in issue keys, use a regex with the parameter like this:

Bitbucket Server and Stash 2.8 or later:

```
"-Dintegration.jira.key.pattern=\"((?!([a-z]{1,10})-?)[a-z]+-\d+)\""
```

See also [Reindex JIRA Software issue keys](#).

As always, please back up your home directory (and perhaps the database) before performing any manual operation on Bitbucket Server. Consider testing this change on another copy of Bitbucket Server before using it in production.

Connecting Bitbucket Server to an external database

This page provides information about using Bitbucket Server with an external database.

Bitbucket Server ships with an embedded database that it uses straight out-of-the-box, with no configuration required. This is great for evaluation purposes, but for production installations we recommend that you use one of the supported external databases.

Please refer to [Supported platforms](#) for the versions of external databases supported by Bitbucket Server.

If you just want to change the password for the external database, see [How do I change the external database password](#).

Instructions for connecting Bitbucket Server to the supported external databases:

- [Connecting Bitbucket Server to PostgreSQL](#)
- [Connecting Bitbucket Server to Oracle](#)
- [Connecting Bitbucket Server to SQL Server](#)
- [Connecting Bitbucket Server to MySQL](#)

MySQL is **not** supported for Bitbucket Data Center instances. MySQL is supported for Bitbucket Server (standalone) instances, but not recommended. See [Connecting Bitbucket Server to MySQL](#) for more information.

Why would I want to use an external database?

Bitbucket Server ships with an embedded database that is great for evaluation purposes, but for production installations we recommend that you make use of one of the [supported](#) external databases, for the following reasons:

- **Improved protection against data loss:** The Bitbucket Server built-in database, which runs [HSQLDB](#), is susceptible to data loss during system crashes. External databases are generally more resistant to data loss during a system crash. HSQLDB is not supported in production environments and should only be used for evaluation purposes.
- **Performance and scalability:** If you have a large number of users on your Bitbucket Server instance, running the database on the same server as Bitbucket Server may slow it down. We recommend that for large installations, Bitbucket Server and the DBMS are run on separate machines. When using the embedded database, the database will always be hosted and run on the same server as Bitbucket Server, which will limit performance.
- **Unified back-up:** Use your existing DBMS tools to back up your Bitbucket Server database alongside your organization's other databases.
- **Bitbucket Data Center support:** If you want to upgrade your instance to [Bitbucket Data Center](#), either now or in the future, to take advantage of the performance-at-scale and high availability benefits of running Bitbucket Server in clustered mode, then you **must** use an external database. HSQLDB is not supported in Bitbucket Data Center.

Using the Database Migration Wizard

The Database Migration Wizard is not supported in Bitbucket Data Center instances while more than one cluster node is running. To migrate databases for a Bitbucket Data Center instance, you should perform the migration before starting multiple cluster nodes.

You can use the Database Migration Wizard to migrate the Bitbucket Server data:

- from the embedded database to a [supported](#) external DBMS.
- to another instance of the same DBMS.
- from one DBMS to another supported DBMS (for example, from MySQL to PostgreSQL).

You need to have created the DBMS (such as PostgreSQL) that you wish to migrate the Bitbucket Server data to before running the Migration Wizard.

To run the Database Migration Wizard:

1. Log in to Bitbucket Server.
2. In the administration area, click **Database** (under 'Settings').

3. Click **Migrate database** and follow the instructions for running the migration.

Notes about database migration

- **Back up the database and Bitbucket home directory:**
Before starting the database migration process you should back up your [Bitbucket Server home directory](#). If you intend to migrate from one external database to another, you should also backup the existing database before proceeding. See [Data recovery and backups](#) for more information.
- **Bitbucket Server will be unavailable during the migration:**
Bitbucket Server will not be available to users during the database migration operation. In addition, running the migration when people are using Bitbucket Server can sometimes cause the migration to time out waiting for all activity in Bitbucket Server that uses the database to complete. For these reasons we recommend that you run the database migration outside of normal usage periods.
- **Migration will usually take less than 30 minutes:**
The duration of the migration process depends on the amount of data in the Bitbucket Server database being migrated. For new installations of Bitbucket Server, containing very little data, the migration process typically takes just a few seconds. If you have been using Bitbucket Server for some time, its database will contain more data, and the migration process will therefore take longer. If Bitbucket Server has been linked to a JIRA Software instance, and there are hundreds of thousands of commits in Bitbucket Server with issue keys in the commit messages, the migration may take tens of minutes.
- **We strongly recommend using a new clean database for the new Bitbucket Server database:**
In case of a migration failure, Bitbucket Server may have partially populated the target database. If the target database is new (therefore empty) and set aside for Bitbucket Server's exclusive use, it's very easy to clean up after a failed migration; just drop the target database and use a clean target database instance for the next attempt.
- **Ensure your [Bitbucket Server home directory](#) is secured against unauthorized access:**
 - After the migration, the connection details (including the username and password) for the database are stored in the `bitbucket.properties` file.
 - Migration will create a dump file of the contents of your database in the [Bitbucket Server home export](#) directory. This is used during the migration and is kept for diagnostic purposes in the case of an error. You may remove this after migration but it may reduce Atlassian Support's ability to help you in the case of migration issues.
 - You can [edit the database password](#) if needed after migration.

Connecting Bitbucket Server to MySQL

This page describes how to connect Bitbucket Server to a MySQL or MariaDB database. The procedure for MySQL and MariaDB is the same, except where noted below. See [Connecting Bitbucket Server to an external database](#) for general information.

MySQL / MariaDB performance issues

MySQL and MariaDB, while supported by Bitbucket Server, are currently **not** recommended especially in larger instances, due to inherent performance and deadlock issues that occur in this database engine under heavy load.

Affected systems may experience slow response times, deadlock errors and in extreme cases errors due to running out of database connections. These issues are intrinsic to MySQL and MariaDB (no other database engine in Bitbucket Server's [Supported platforms](#) shares this behavior) and are due to the way MySQL and MariaDB perform row-level locking in transactions. See <http://dev.mysql.com/doc/refman/5.0/en/innodb-deadlocks.html> for some general information on this.

Bitbucket Server does its best to work around the MySQL / MariaDB behavior - see issues [STASH-4517](#), [STASH-4701](#) and others for example. However, under very heavy load you will generally get better performance with any of the other database engines supported by Bitbucket Server (such as PostgreSQL which is also freely available) than you will with MySQL or MariaDB. Please see [Connecting Bitbucket Server to an external database](#) for instructions on migrating your data to one of these other engines.

MySQL and MariaDB are not supported in Bitbucket Data Center

Bitbucket Data Center does not support any version of MySQL or MariaDB. With Bitbucket Data Center you must use one of the other database engines [supported](#) by Bitbucket Server (such as PostgreSQL which is also freely available). Please see [Connecting Bitbucket Server to an external database](#) for instructions on migrating your data to one of these other engines.

MySQL 5.6.x compatibility

Note that Bitbucket Server is not compatible at all with versions of MySQL 5.6 earlier than 5.6.16 because of bugs in its query optimizer ([#68424](#), [#69005](#)). Please watch [STASH-3164](#) for further updates on this. Bitbucket Server does support versions of MySQL 5.6 from 5.6.16 on.

See [Supported platforms](#) for the versions of MySQL and MariaDB supported by Bitbucket Server.

The overall process for using a MySQL or MariaDB database with Bitbucket Server is:

1. Install MySQL or MariaDB where it is accessible to Bitbucket Server. It is assumed here that you already have MySQL or MariaDB installed and running. See the MySQL documentation at <http://dev.mysql.com/doc/>.
2. [Create the database and user](#) on the MySQL / MariaDB server for Bitbucket Server to use.
3. [Download and install the JDBC driver](#).
4. [Migrate Bitbucket Server to the MySQL / MariaDB database](#).

Create the Bitbucket Server database

Before you can use Bitbucket Server with MySQL or MariaDB, you must set up the MySQL or MariaDB server as follows:

Step	Notes
Create database	Create a database on MySQL or MariaDB for Bitbucket Server to use.
Create database user	Create a Bitbucket Server user on the database.
Character encoding	Configure the database to use utf8 character set encoding. Note that Bitbucket Server on MySQL and MariaDB does not support 4 byte UTF-8 characters .
Collation	Configure the database to use utf8_bin collation (to ensure case sensitivity).
Logging format	If MySQL or MariaDB is using binary logging, configure the database to use a binary logging format of either MIXED or ROW. Refer to the MySQL documentation. Note that Bitbucket Server sets the MySQL / MariaDB transaction isolation level to READ-COMMITTED when it connects to the database. Packages of MySQL or MariaDB in some Linux distributions may be configured with <code>binlog_fmt=statement</code> by default. Before using such packages with Bitbucket Server you must change this to either <code>mixed</code> or <code>row</code>. See this KB article for more information.
Connection timeout	Bitbucket Server requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be seemingly random connection errors.

Here is an example of how to do that. When Bitbucket Server and MySQL / MariaDB run on the same physical computer (accessible through `localhost`), run the following commands (replacing `bitbucketuser` and `password` with your own values):

```
mysql> CREATE DATABASE bitbucket CHARACTER SET utf8 COLLATE utf8_bin;
mysql> GRANT ALL PRIVILEGES ON bitbucket.* TO
'bitbucketuser'@'localhost' IDENTIFIED BY 'password';
mysql> FLUSH PRIVILEGES;
mysql> QUIT
```

This creates an empty MySQL / MariaDB database with the name `bitbucket`, and a user that can log in from the host that Bitbucket Server is running on who has full access to the newly created database. In particular, the user should be allowed to create and drop tables, indexes and other constraints.

If the MySQL / MariaDB database and Bitbucket Server instances 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).

If the MySQL / MariaDB database and Bitbucket Server instances are on different computers, just replace the `localhost` part of the `GRANT ALL` statement above with the hostname of the machine that Bitbucket Server is running on. See the documentation at <http://dev.mysql.com/doc/refman/5.1/en/account-names.html>.

Note that Bitbucket Server will generally require about 25–30 connections to the database. The maximum number of connections is a configurable system property – see [Database pool](#).

Download and install the JDBC driver

The JDBC drivers for MySQL / MariaDB are *not* bundled with Bitbucket Server (due to licensing restrictions). You need to download and install the driver yourself, after you have installed Bitbucket Server.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).

The MariaDB Java Client is not compatible with Bitbucket Server

The MySQL Connector/J must be used for both MySQL and MariaDB. The MariaDB Java Client is *not* compatible with Bitbucket Server and is not supported.

2. Expand the downloaded zip/tar.gz file.
3. Copy the `mysql-connector-java-5.1.XX-bin.jar` file from the extracted directory to your `<Bitbucket home directory>/lib` directory (for Bitbucket Server 2.1 or later).
4. Stop, and then restart Bitbucket Server. See [Starting and stopping Bitbucket Server](#).

Migrate Bitbucket Server to the MySQL / MariaDB database

You can migrate Bitbucket Server to the MySQL or MariaDB database created above, either from the embedded database or from another external database.

The migration process makes a backup of your existing Bitbucket Server database in `exports` under the Bitbucket Server [home directory](#). See [Data recovery and backups](#) for further information about backing up Bitbucket Server.

Run the migration as follows:

1. In the administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **MySQL** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

See [these notes](#) about database migration.

Migrate Database

Bitbucket will be unavailable to users while a migration is in progress. See our [documentation](#) for more information

Database Type

PostgreSQL

Hostname*

Hostname or IP address of the database server

Port*

5432

TCP port number for the database server

Database name*

Database username*

Database password

Start migration

Test

Cancel

Hostname	The host name or IP address of the computer running the database server.
Port	The TCP port with which Bitbucket Server can connect to the database server. The default value is the default port that MySQL or MariaDB runs against. You can change that if you know the port that your MySQL or MariaDB instance is using.
Database name	The name of the database that Bitbucket Server should connect to.
Database username	The username that Bitbucket Server should use to access the database.
Database password	The password that Bitbucket Server should use to access the database.

Connecting Bitbucket Server to Oracle

This page describes how to connect Bitbucket Server to a Oracle database.

The overall process for using a Oracle database with Bitbucket Server is:

- Install Oracle where it is accessible to Bitbucket Server.
- Create a database and user on the Oracle server for Bitbucket Server to use.
- Install Bitbucket Server on Windows, or on Linux or Mac. See [Getting started](#).
- Either:
 - at Bitbucket Server install time, run the Setup Wizard to connect Bitbucket Server to the Oracle database, or
 - at a later time, migrate Bitbucket Server to the Oracle database. See [Using the Database Migration Wizard](#).

It is assumed here that you already have Oracle installed and running. For information about installing Oracle and creating Oracle databases, see the [Oracle documentation pages](#). For the versions of Oracle supported by Bitbucket Server see [Supported platforms](#).

On this page:

[Prerequisites](#)
[Connect Bitbucket Server to the Oracle database](#)

Related pages:

- [Connecting Bitbucket Server to an external database](#)
- [Connecting Bitbucket Server to MySQL](#)
- [Connecting Bitbucket Server to PostgreSQL](#)
- [Connecting Bitbucket Server to SQL Server](#)

Prerequisites**Backup**

If you are migrating your data from the internal Bitbucket Server database, back up the [Bitbucket Server home directory](#).

If you are migrating your Bitbucket Server data from a different external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See [Data recovery and backups](#).

Create the Bitbucket Server database

Before you can use Bitbucket Server with Oracle, you must set up Oracle as follows:

- Ensure that you have a database instance available for Bitbucket Server (either create a new one or use an existing one)
The character set of the database must be set to either AL32UTF8 or UTF8, to support storage of Unicode data as per the [Oracle documentation](#).
Note that it is important to the proper operation of Bitbucket Server that the database store its data in a case-sensitive manner. By changing the values of the [NLS_COMP](#) and/or [NLS_SORT](#) variables, it is possible to cause Oracle to perform its searches in a case-insensitive manner. We therefore strongly recommend that those variables be left at their default values.
- Create a user that Bitbucket Server will connect as (e.g. **bitbucket**).
 Remember the database user name; it will be used to configure Bitbucket Server's connection to the database in subsequent steps.
 When you create a user in Oracle, a schema is automatically created.
It is strongly recommended that you create a new database user for use by Bitbucket Server rather than sharing one that is used by other applications or people.
- Grant the Bitbucket Server user connect and resource roles only. The connect role is required to set up a connection, while resource role is required to allow the user to create objects in its own schema.
- Create a local `all_objects` view to the user's schema, so that there is no possibility that a table with the same name as one of the Bitbucket Server tables in another schema will cause any conflicts.
- Note that Bitbucket Server requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be [seemingly random connection errors](#).

The format of the command to create a user in Oracle is:

```
CREATE USER <user>
  IDENTIFIED BY <password>
  DEFAULT TABLESPACE USERS
  QUOTA UNLIMITED ON USERS;
GRANT CONNECT, RESOURCE to <user>;
CREATE VIEW <user>.all_objects AS
  SELECT *
  FROM sys.all_objects
  WHERE owner = upper('<user>');
```

Here is a simple example, using SQL*Plus, of how one might create a user called **bitbucket** with password **jdHyd6Sn21** in tablespace **users**, and grant the user a minimal set of privileges. When you run the command on your machine, remember to replace the user name, password and tablespace names with your own values.

```
CREATE USER bitbucket
  IDENTIFIED BY jdHyd6Sn21
  DEFAULT TABLESPACE USERS
  QUOTA UNLIMITED ON USERS;
GRANT CONNECT, RESOURCE to bitbucket;
CREATE VIEW bitbucket.all_objects AS
  SELECT *
  FROM sys.all_objects
  WHERE owner = upper('bitbucket');
```

This creates an empty Oracle schema with the name **bitbucket**, and a user that can log in from the host that Bitbucket Server is running on and who has full access to the newly created schema. In particular, the user is allowed to create sessions and tables.

Bitbucket Server will generally require about 25–30 connections to the database. The maximum number of connections is a configurable system property – see [Database pool](#).

Connect Bitbucket Server to the Oracle database

You can now connect Bitbucket Server to the Oracle database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate to Oracle, either from the embedded Bitbucket Server database or from another external database.

When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **Oracle** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow the instructions in the Bitbucket Server Setup Wizard.

When migrating to Oracle

1. In the Bitbucket Server administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **Oracle** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Hostname	The host name or IP address of the computer running the Oracle server.
Port	The TCP port with which Bitbucket Server can connect to the database server. The default value is the default port that Oracle runs against. You can change that if you know the port that your Oracle instance is using.

Database name	The system identifier of the Oracle instance that Bitbucket Server should connect to. Bitbucket Server does not support connecting to Oracle servers which are using SIDs or TNS Alias to identify themselves; it requires the fully qualified Service Name instead.
Database username	The username that Bitbucket Server should use to access the database.
Database password	The password that Bitbucket Server should use to access the database.

Migrate Database

Stash will be unavailable to users while a migration is in progress. See our [documentation](#) for more information.

Database Type:

Hostname*:

Hostname or IP address of the database server

Port*:

TCP port number for the database server

Database name*:

Database username*:

Database password:

Connecting Bitbucket Server to PostgreSQL

This page describes how to connect Bitbucket Server to a PostgreSQL database.

The overall process for using a PostgreSQL database with Bitbucket Server is:

- Install PostgreSQL where it is accessible to Bitbucket Server.
- Create a database and user on the PostgreSQL server for Bitbucket Server to use.
- Install Bitbucket Server on Windows, or on Linux or Mac. See [Getting started](#).
- Either:
 - at Bitbucket Server install time, run the Setup Wizard to connect Bitbucket Server to the PostgreSQL database, or
 - at a later time, migrate Bitbucket Server to the PostgreSQL database. See [Using the Database Migration Wizard](#).

It is assumed here that you already have PostgreSQL installed and running. For more information about PostgreSQL installation and operation, refer to the [PostgreSQL documentation](#). For additional information review this page on [tuning](#).

PostgreSQL has the idea of schemas. When you create a PostgreSQL database, a 'public' schema is created and set as the default for that database. It is possible to create a different schema (e.g. 'bitbucket') and set that as the default schema. Bitbucket Server will use whatever schema is set as the default for the logged-in user. Bitbucket Server does not provide a way for a user to nominate the schema to use; it uses schema that is set as the PostgreSQL default.

See [Supported platforms](#) for the versions of PostgreSQL supported by Bitbucket Server.

On this page:

[Prerequisites](#)
[Connect Bitbucket Server to the PostgreSQL database](#)

Related pages:

- [Connecting Bitbucket Server to an external database](#)
- [Connecting Bitbucket Server to MySQL](#)
- [Connecting Bitbucket Server to Oracle](#)
- [Connecting Bitbucket Server to SQL Server](#)

Prerequisites**Backup**

If you are migrating your Bitbucket Server data from the HSQL internal database, back up the [Bitbucket Server home directory](#).

If you are migrating your Bitbucket Server data from another external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See [Data recovery and backups](#).

Create the Bitbucket Server database

Before you can use Bitbucket Server with PostgreSQL, you must:

- Create a role for Bitbucket Server to use when it connects to the database. We strongly recommend that this role be established for Bitbucket Server's use exclusively; it should not be shared by other applications or people.
- Create a database in which Bitbucket Server can store its data. The database must be configured to use the UTF-8 character set. During normal operation, Bitbucket Server will acquire 25–30 connections to the database. The maximum number of connections is a configurable system property – see [Database pool](#).
- Note that Bitbucket Server requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be [seemingly random connection errors](#).

Here is an example of how to create a user called **bitbucketuser** with password **jellyfish** , and a database called **bitbucket** , which is configured for use by bitbucketuser. Using a PostgreSQL client application like [psql](#) or [pgAdmin](#), run the following commands, replacing the user name, password, and database name with your own values.

```
CREATE ROLE bitbucketuser WITH LOGIN PASSWORD 'jellyfish' VALID UNTIL
'infinity';

CREATE DATABASE bitbucket WITH ENCODING='UTF8' OWNER=bitbucketuser
CONNECTION LIMIT=-1;
```

If the server that is hosting the PostgreSQL database is not the same server as Bitbucket Server, then please ensure that the Bitbucket Server server can connect to the database server. Please also refer to the [PostgreSQL documentation on how to set up pg_hba.conf](#). If the `pg_hba.conf` file is not set properly, remote communication to the PostgreSQL server will fail.

Connect Bitbucket Server to the PostgreSQL database

You can now connect Bitbucket Server to the PostgreSQL database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate Bitbucket Server to PostgreSQL, either from the embedded HSQL database or from another external database.

When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **PostgreSQL** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow the instructions in the Bitbucket Server Setup Wizard.

When migrating to PostgreSQL

1. In the Bitbucket Server administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **PostgreSQL** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Hostname	The host name or IP address of the computer running the database server.
Port	The TCP port with which Bitbucket Server can connect to the database server. The default value is the default port that PostgreSQL runs against. You can change that if you know the port that your PostgreSQL instance is using.
Database name	The name of the database that Bitbucket Server should connect to.
Database username	The username that Bitbucket Server should use to access the database.
Database password	The password that Bitbucket Server should use to access the database.

Migrate Database

Stash will be unavailable to users while a migration is in progress. See our [documentation](#) for more information.

Database Type: PostgreSQL

Hostname*:

Port*: 5432

Database name*:

Database username*:

Database password:

Start Migration Test Cancel

Connecting Bitbucket Server to SQL Server

This page describes how to connect Bitbucket Server to a Microsoft SQL Server database.

The overall process for using a SQL Server database with Bitbucket Server is:

- Install SQL Server where it is accessible to Bitbucket Server.
- Create a database and user on the SQL

- Server instance for Bitbucket Server to use.
- Install Bitbucket Server on Windows, or on Linux or Mac. See [Getting started](#).
- Either:
 - at Bitbucket Server install time, run the Setup Wizard to connect Bitbucket Server to the SQL Server database, or
 - at a later time, migrate Bitbucket Server to the SQL Server database. See [Using the Database Migration Wizard](#).

On this page

- [Prerequisites](#)
- [Connect Bitbucket Server to the SQL Server database](#)
- [Use Integrated Authentication or 'Windows Authentication Mode' \(Optional\)](#)
- [Install the JDBC driver](#)

Related pages

- [Transitioning from jTDS to Microsoft's JDBC driver](#)
- [Connecting Bitbucket Server to an external database](#)
- [Connecting Bitbucket Server to MySQL](#)
- [Connecting Bitbucket Server to Oracle](#)
- [Connecting Bitbucket Server to PostgreSQL](#)

It is assumed here that you already have SQL Server installed and running.

- SQL Server documentation is available at <http://msdn.microsoft.com/en-us/library/bb545450.aspx>.
- JDBC documentation is available at <http://msdn.microsoft.com/en-us/library/ms378672.aspx>.

See [Supported platforms](#) for the versions of SQL Server supported by Bitbucket Server.

Prerequisites

Back up your current database

If you are migrating your data from the internal Bitbucket Server database, back up the [Bitbucket Server home directory](#).

If you are migrating your Bitbucket Server data from a different external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See [Data recovery and backups](#).

Create the Bitbucket Server database

Before you can use Bitbucket Server with SQL Server, you must set up SQL Server as follows:

Step	Notes
Create a database	e.g. bitbucket. Remember this database name for the connection step below.
Set the collation type	This should be case-sensitive, for example, 'SQL_Latin1_General_CP1_CS_AS' (CS = Case Sensitive).
Set the isolation level	Configure the database to use the isolation level, Read Committed with Row Versioning.
Create a database user	e.g. bitbucketuser. This database user should not be the database owner, but should be in the db_owner role. It needs to be in this role during setup <i>and</i> at all points when Bitbucket Server is running due to the way Bitbucket Server interacts with the database. See SQL Server Startup Errors . Remember this database user name for the connection step below.

Set database user permissions	The Bitbucket Server database user has permission to connect to the database, and to create and drop tables, indexes and other constraints, and insert and delete data, in the newly-created database.
Enable TCP/IP	Ensure that TCP/IP is enabled on SQL Server and that SQL Server is listening on the correct port (which is 1433 for a default SQL Server installation). Remember this port number for the connection step below.
Check the authentication mode	Ensure that SQL Server is operating in the appropriate authentication mode. By default, SQL Server operates in 'Windows Authentication Mode'. However, if your user is not associated with a trusted SQL connection, 'Microsoft SQL Server, Error: 18452' is received during Bitbucket Server startup, and you will need to change the authentication mode to 'Mixed Authentication Mode'. <i><u>Bitbucket Server instances running on Windows</u> are also able to support SQL Server databases running in 'Windows Authentication Mode'. This is described at the bottom of this page and it has to be manually configured: Connecting Bitbucket Server to SQL Server - Use Integrated Authentication (Optional)</i>
Check that SET NOCOUNT is off	Ensure that the SET NOCOUNT option is turned off. You can do that in SQL Server Management Studio as follows: 1. Navigate to Tools > Options > Query Execution > SQL Server > Advanced. Ensure that the SET NOCOUNT option is cleared. 2. Now, go to the Server > Properties > Connections > Default Connections properties box and clear the no count option.

Note that Bitbucket Server will generally require about 25–30 connections to the database.

Note also that Bitbucket Server requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be [seemingly random connection errors](#).

Here is an example of how to create and configure the SQL Server database from the command line. When Bitbucket Server and SQL Server run on the same physical computer (accessible through localhost), run the following commands (replacing bitbucketuser and password with your own values):

```
SQL Server> CREATE DATABASE bitbucket
SQL Server> GO
SQL Server> USE bitbucket
SQL Server> GO
SQL Server> ALTER DATABASE bitbucket SET ALLOW_SNAPSHOT_ISOLATION ON
SQL Server> GO
SQL Server> ALTER DATABASE bitbucket SET READ_COMMITTED_SNAPSHOT ON
SQL Server> GO
SQL Server> ALTER DATABASE bitbucket COLLATE
SQL_Latin1_General_CP1_CS_AS
SQL Server> GO
SQL Server> SET NOCOUNT OFF
SQL Server> GO
SQL Server> USE master
SQL Server> GO
SQL Server> CREATE LOGIN bitbucketuser WITH PASSWORD=N'password',
DEFAULT_DATABASE=bitbucket, CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
SQL Server> GO
SQL Server> ALTER AUTHORIZATION ON DATABASE::bitbucket TO
bitbucketuser
SQL Server> GO
```


This creates an empty SQL Server database with the name `bitbucket`, and a user that can log in from the host that Bitbucket Server is running on who has full access to the newly created database. In particular, the user should be allowed to create and drop tables, indexes and other constraints.

Connect Bitbucket Server to the SQL Server database

You can now connect Bitbucket Server to the SQL Server database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate to SQL Server, either from the embedded database or from another external database.

When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **SQL Server** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow the instructions in the Bitbucket Server Setup Wizard.

When migrating to SQL Server

1. In the Bitbucket Server administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **SQL Server** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Hostname	The host name or IP address of the computer running the database server.
Port	The TCP port with which Bitbucket Server can connect to the database server. The default value of 1433 is the default port that SQL Server runs against. You can change that if you know the port that your SQL Server instance is using.
Database name	The name of the database that Bitbucket Server should connect to.
Database username	The username that Bitbucket Server should use to access the database.
Database password	The password that Bitbucket Server should use to access the database.

Migrate Database

Bitbucket will be unavailable to users while a migration is in progress. See our [documentation](#) for more information

Database Type:

Hostname:

Port:

Database name:

Database username:

Database password:

Named Instances

If you have a named instance on your server, you will need to manually edit the `bitbucket.properties` file as described on the [Connecting to named instances in SQL Server from Bitbucket Server](#) Knowledge Base article.

Use Integrated Authentication or 'Windows Authentication Mode' (Optional)

Windows authentication is only available for Bitbucket Server instances running on Windows. It cannot be used on Linux because Microsoft does not provide shared objects for it. You will either need to run Bitbucket Server on Windows, allowing you to use Windows security, or you will need to enable mixed-mode authentication for SQL Server if you are running Bitbucket Server on Linux. Unfortunately, there are no other options at this time.

Integrated authentication uses a native DLL to access the credentials of the logged-in user to authenticate with SQL Server. The native DLLs for both 32- and 64-bit systems are included in the distribution; there is no need to download the entire package from Microsoft.

Bitbucket Server does not currently support configuring the system to use integrated authentication from the

BSERV-3035 - Add support for integrated authentication for Microsoft SQL Server

UI (Vote for it! [OPEN](#))

). This means

you can't currently migrate to SQL Server with integrated authentication, nor can you configure Bitbucket Server to use SQL Server with integrated authentication during initial setup. However, if Bitbucket Server has already been configured to use SQL Server (for example, when the Setup Wizard was run at first use), you can enable integrated authentication by directly modifying Bitbucket Server's configuration, as follows:

1. Based on the JVM being used to run Bitbucket Server, rename either the x64 or x86 DLL to `sqljdbc_auth.dll` in `lib/native`. Note that running on Windows x64 does *not* require the use of the x64 DLL; you should only use the x64 DLL if you are also using a 64-bit JVM.
2. In `setenv.bat`, a `JVM_LIBRARY_PATH` variable has already been defined. Simply remove the leading `rem`. Note that if you are putting the native DLL in an alternative location, you may need to change the value to point to your own path. The value of the `JVM_LIBRARY_PATH` variable will automatically be included in the command line when Tomcat is run using `start-bitbucket.bat`.
3. Edit the `%BITBUCKET_HOME%\shared\bitbucket.properties` file to include `;integratedSecurity=true` in the `jdbc.url` line. Note that `jdbc.user` and `jdbc.password` will no longer be used to supply credentials *but they must still be defined* – Bitbucket Server will fail to start if these properties are removed.
4. Ensure the Bitbucket Server process or service is running as the correct user to access SQL Server. (Note that this user is generally a Windows Domain User Account, but should not be a member of any administrators groups, that is local, domain, or enterprise.)

It is also possible to configure integrated authentication over Kerberos, rather than using the native DLLs. Details for that are included in the [JDBC documentation](#).

Install the JDBC driver

This section is only relevant to some distributions of Bitbucket Server, for example if you are running Bitbucket Server via the Atlassian Plugin SDK, or have built Bitbucket Server from source.

If the SQL Server JDBC driver is *not* bundled with Bitbucket Server, you will need to download and install the driver yourself.

1. Download the appropriate JDBC driver from the Microsoft [download site](#).
2. Install the driver file to your <Bitbucket home directory> /lib directory (for Bitbucket Server 2.1 or later).
3. Stop, then restart, Bitbucket Server. See [Starting and stopping Bitbucket Server](#).

If Bitbucket Server was configured to use Microsoft SQL Server by manually entering a JDBC URL, please refer to [this guide](#).

Transitioning from jTDS to Microsoft's JDBC driver

This page describes how to change from using jTDS to using the Microsoft SQL Server JDBC driver to access Microsoft SQL Server.

What do I have to do?

If Bitbucket Server was configured to use Microsoft SQL Server by following the steps outlined in [Connecting Bitbucket Server to SQL Server](#), *no change is necessary*. However, If Bitbucket Server was configured to use Microsoft SQL Server by **manually entering a JDBC URL**, the system will lock on startup if the driver class and URL are not manually updated.

How to proceed

In the [Bitbucket Server home directory](#), `bitbucket.properties` must be edited to change the JDBC driver and URL. The existing configuration should look similar to this:

```
jdbc.driver=net.sourceforge.jtds.jdbc.Driver
jdbc.url=jdbc:jtds:sqlserver://localhost:1433;databaseName=stash;
jdbc.user=stashuser
jdbc.password=secretpassword
```

 The JDBC URL above is in the format constructed by Bitbucket Server when Connecting Bitbucket Server to SQL Server and will automatically be updated to a URL compatible with Microsoft's driver, with no change required on the administrator's part. If the URL contains additional properties, such as `domain=`, it will need to be manually updated.

To use Microsoft's SQL Server driver, the settings above would be updated to this:

```
jdbc.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
jdbc.url=jdbc:sqlserver://localhost:1433;databaseName=stash;
jdbc.user=stashuser
jdbc.password=secretpassword
```

The exact values to use in the new URL are beyond the scope of this documentation; they must be chosen based on the jTDS settings they are replacing.

Additional Information for the curious

The new JDBC driver class is: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

The JDBC URL format for the jTDS driver is documented on SourceForge at <http://jtds.sourceforge.net/faq.html#urlFormat>.

The JDBC URL format for Microsoft's SQL Server driver is documented on MSDN at <http://msdn.microsoft.com/en-us/library/ms378428.aspx>, with documentation for additional properties at <http://msdn.microsoft.com/en-us/library/ms378988.aspx>.

Why change drivers?

▼ [Click here to find all the technical details...](#)

Recent releases of Hibernate, which Bitbucket Server uses to simplify its persistence layer, have introduced a requirement that the JDBC drivers and connection pools used be JDBC4-compliant. JDBC4 was introduced with Java 6.

The jTDS driver used by releases prior to Bitbucket Server 2.1 is a JDBC3 driver, compatible with Java 1.3, and therefore cannot be used with newer versions of Hibernate. While jTDS 1.3.0 implements JDBC4, and JDBC4.1 which is provided by Java 7, it *requires* a Java 7 runtime environment. Upgrading Bitbucket Server to that version was a non-starter, as it would require raising the minimum Java version for the product to Java 7.

Instead, the decision was made to replace jTDS with Microsoft's own SQL Server driver. Microsoft's driver is actively maintained, where jTDS is only recently seeing its first updates in over 3 years, and supports all the features of SQL Server, including SQL Server 2012.

Bitbucket Server attempts to automatically update jTDS JDBC URLs to values compatible with Microsoft's JDBC driver. However, for installations using custom JDBC URLs—for example, to use domain authentication—such automatic updating is not possible; the URL, which was manually entered, must be manually updated.

Migrating Bitbucket Server to another server

This page describes how to move your Bitbucket Server installation from one physical machine to a different machine. For most scenarios, the overall procedure involves the following 4 steps, although your situation may not require all of these:

1. Prepare for the migration.
2. Move the Bitbucket Server data.
3. Move the Bitbucket Server installation to the new location, and update the value of the BITBUCKET_HOME environment variable.
4. Update the Bitbucket Server bitbucket.properties file. This will be necessary if you were unable to use the Migration Wizard in Step 2.

See also the [Bitbucket Server upgrade guide](#). You can upgrade Bitbucket Server either before or after you migrate Bitbucket Server. This page *does not* describe any aspect of the upgrade procedure.

On this page:

- [1. Prepare for the migration](#)
- [2. Move the Bitbucket Server data to a different machine](#)
- [3. Move Bitbucket Server to a different machine](#)
- [4. Update the Bitbucket Server configuration](#)

Related pages:

- [Supported platforms](#)
- [Connecting Bitbucket Server to an external database](#)
- [Bitbucket Server upgrade guide](#)

1. Prepare for the migration

In preparation for migrating Bitbucket Server to another server, check that you have done the following:

- Confirm that the operating system, database, other applicable platforms and hardware on the new machine will comply with the [requirements](#) for Bitbucket Server.
- Check for any known migration issues in the [Bitbucket Server Knowledge Base](#).
- Alert users to the forthcoming Bitbucket Server service outage.
- Ensure that users will not be able to update existing Bitbucket Server data during the migration. You can do this by temporarily changing the access permissions for Bitbucket Server.
- Make sure you have [created a user in Bitbucket Server](#) (not in your external user directory) that has System Administrator [global permissions](#) so as to avoid being locked out of Bitbucket Server in case the new server does not have access to your external user directory.

2. Move the Bitbucket Server data to a different machine

This section gives a brief overview of how to move the Bitbucket Server data to a different machine. You do not need to do anything in this section if you will continue to use the embedded database - the Bitbucket Server data is moved when you move the Bitbucket Server installation.

The Bitbucket Server data includes the data directories (including the Git repositories), log files, installed plugins, temporary files and caches.

You can move the Bitbucket Server data:

- from the embedded database to a [supported](#) external DBMS.
- to another instance of the same DBMS.
- from one DBMS to another supported DBMS (for example, from MySQL to PostgreSQL).

You can also move the actual DBMS. Atlassian recommends that for large installations, Bitbucket Server and the DBMS run on separate machines.

There are 2 steps:

1. Create and configure the DBMS in the new location. Please refer to [Connecting Bitbucket Server to an external database](#), and the relevant child page, for more information.
2. Either:
 - If the new location is currently visible to Bitbucket Server, use the Bitbucket Server Database Migration Wizard. Please refer to [Connecting Bitbucket Server to an external database](#), and the relevant child page, for more information.
 - If the new location is not currently visible to Bitbucket Server (perhaps because you are moving to a new hosting provider), you need to perform a database export and then import the backup to the new DBMS. Please refer to the vendor documentation for your DBMS for detailed information. You will also need to update the `bitbucket.properties` file in the `<Bitbucket home directory>` as described below.

3. Move Bitbucket Server to a different machine

This section describes moving the Bitbucket Server installation to a different machine.

1. Stop Bitbucket Server. See [Starting and stopping Bitbucket Server](#).
2. Make an archive (such as a zip file) of the Bitbucket home directory. The home directory contains data directories (including the Git repositories), log files, installed plugins, SSH fingerprints, temporary files and caches. The home directory location is defined:
 - on Windows, by the `BITBUCKET_HOME` environment variable, or by the `BITBUCKET_HOME` line of `<Bitbucket Server installation directory>/bin/setenv.bat`.
 - on Linux and Mac, by the `BITBUCKET_HOME` line of `<Bitbucket Server installation directory>/bin/setenv.sh`.
3. Copy the archive of the Bitbucket home directory to the new machine and unzip it to its new location there.
 - For production environments the Bitbucket Server home directory should be secured against unauthorized access. See [Bitbucket home directory](#).
 - When moving the Bitbucket Server home directory from Windows to Linux or Mac, make sure that the files within `<Bitbucket home directory>/git-hooks` and `<Bitbucket home directory>/shared/data/repositories/<repoID>/hooks` directories have the executable file permission set.
4. Set up an instance of Bitbucket Server in the new location by doing one of the following:
 - Make an archive of the old Bitbucket Server installation directory and copy it across to the new machine.
 - Install the same version of Bitbucket Server from scratch on the new machine.
5. Redefine the value for `BITBUCKET_HOME`, mentioned in Step 2. above, in the new `<Bitbucket Server installation directory>`, using the new location for your copied home directory. See [Bitbucket Server home directory](#) for more information.
6. If you are continuing to use the Bitbucket Server embedded database, or you used the Migration Wizard to move the Bitbucket Server data, you should now be able to start Bitbucket Server on the new machine

and have all your data available. See [Starting and stopping Bitbucket Server](#). Once you have confirmed that the new installation of Bitbucket Server is working correctly, revert the access permissions for Bitbucket Server to their original values.

7. If you moved the Bitbucket Server data by performing a database export and import, carry on to Step 4. below to update the `bitbucket.properties` file in the `<Bitbucket home directory>`.

4. Update the Bitbucket Server configuration

If you moved the Bitbucket Server data by performing a database export, you must update the `bitbucket.properties` file within `<Bitbucket home directory>/shared` with the changed configuration parameters for the database connection.

The configuration parameters are described in [Bitbucket Server config properties](#).

Once the configuration parameters are updated, you should be able to start Bitbucket Server on the new machine and have all your data available. See [Starting and stopping Bitbucket Server](#). Once you have confirmed that the new installation of Bitbucket Server is working correctly, revert the access permissions for Bitbucket Server to their original values.

Specifying the base URL for Bitbucket Server

This is the base URL for this installation of Bitbucket Server. All links *which are not from a web request* (for example in Bitbucket Server email notifications), will be prefixed by this URL. If you are experiencing trouble with setting an `https` base URL, please ensure you have configured [Tomcat with SSL](#) correctly.

To specify Bitbucket Server's base URL:

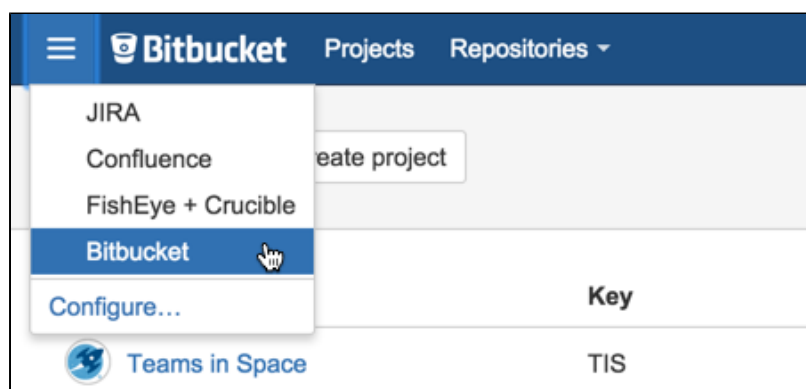
1. In the Bitbucket Server administration area, click **Server settings** (under 'Settings').
2. In the **Base URL** field, type the URL address of your Bitbucket Server instance (for example, "`https://bitbucket.mycompany.com`").
3. Click **Save**.

Related pages:

- [Administering Bitbucket Server](#)

Configuring the application navigator

The application navigator, on the left of the Bitbucket Server header, allows you to switch to your other applications, such as JIRA Software and Bamboo – or any other web application – all from the Bitbucket Server header:



Users only see the application navigator when links are set up – if there are no links, only administrators can see it.

Bitbucket Server administrators can configure which apps appear in the navigator – just click **Configure** in the application navigator, or go to the Bitbucket Server admin area and click **Application Navigator**:

- [Linked applications](#) are automatically configured in the application navigator, and can't be deleted. Click **Manage** to configure those in the source application.
- Specify new links, as required by your system, by entering a **Name** and **URL**.
- Restrict the visibility of links to particular user groups, or hide the link completely. Click in a row, under the

Groups column header, to edit those properties for existing rows.

- Use the 'handles' at the left to change the link order when seen in Bitbucket Server.

Application Navigator

The application navigator appears in the top left corner of the header so users can quickly access other applications. Linked applications are automatically added to the application navigator for you, and you can't delete them. Add your own links for users by providing the name and URL of the destination below.

Name	URL	Hide	Restricted to Groups	
		☐		Add
JIRA	http://jira.teamsinspace.com:8080/	Manage		
Confluence	http://confluence.teamsinspace.com:8090/	Manage		
FishEye + Crucible	http://fecru.teamsinspace.com:8060/	Location		
Bitbucket	https://bitbucket.dev.com:7990/			Delete

Managing add-ons

An add-on is an installable component that supplements or enhances the functionality of Bitbucket Server in some way. For example, the [Custom Navigation Plugin](#) enables you to configure custom navigation tabs specific to a repository. Other add-ons are available for adding graphs to Bitbucket Server, importing SVN source control projects into Bitbucket Server, and accessing Atlassian support from Bitbucket Server.

Bitbucket Server comes with many pre-installed add-ons (called system add-ons). You can install more add-ons, either by acquiring the add-on from the [Atlassian Marketplace](#) or by uploading it from your file system. This means that you can install add-ons that you have developed yourself. For information about developing your own add-ons for Bitbucket Server, see the [Bitbucket Server Developer Documentation](#).

On this page

- [About the Universal Plugin Manager \(UPM\)](#)
- [Administering add-ons in Bitbucket Server](#)
- [Add-ons for Bitbucket Data Center](#)

About the Universal Plugin Manager (UPM)

You administer add-ons for Bitbucket Server using the Universal Plugin Manager (UPM). The UPM is itself an add-on that exposes add-on administration pages in the Bitbucket Server Administration Console. UPM works across Atlassian applications, providing a consistent interface for administering add-ons in Bitbucket Server, Crucible, Confluence, FishEye, JIRA applications, and Bamboo.

UPM comes pre-installed in recent versions of all Atlassian applications, so you do not normally need to install it yourself. However, like other add-ons, the UPM software is subject to regular software updates. Before administering add-ons in Bitbucket Server, therefore, you should [verify your version](#) of the UPM and update it if needed.

Administering add-ons in Bitbucket Server

You can update UPM, or any add-on, from the UPM's own add-on administration pages. Additionally, you can perform these tasks from the UPM administration pages:

- Install or remove add-ons
- Configure add-on settings
- Discover and install new add-ons from the [Atlassian Marketplace](#)
- Enable or disable add-ons and their component modules

It shows only those plugins that are supported in your version of the product, so that you do not install incompatible plugins.

If the add-on request feature is enabled in your Atlassian application, non-administrative users can also discover add-ons on the Atlassian Marketplace. Instead of installing the add-ons, however, these users have the option of requesting the add-ons from you, the administrator of the Atlassian application.

For more information on administering the add-on request feature or performing other common add-on administration tasks, see the [Universal Plugin Manager documentation](#). For an end-user's view of requesting add-ons in Bitbucket Server, see [Requesting add-ons](#).

Add-ons for Bitbucket Data Center

Installing, and managing, add-ons for Bitbucket Data Center is done in the same way as for Bitbucket Server, as described above. The only requirement is that the add-on is Data Center-compatible – see [Bitbucket Data Center Add-ons](#) for compatibility information.

You can install an add-on from any cluster node. The add-on is stored on the [shared file system](#) for the Bitbucket Data Center, and made available to all nodes in the cluster.

POST service webhook for Bitbucket Server

Repository administrators can add a POST service to a repository. Bitbucket Server POSTs to the service URL you specify.

You can use an URL with the following format:

```
https://server:port/path/
```

The service receives a POST whenever the user pushes to the repository.

The content type header of the POST has an `application/json` type. The content is a JSON payload that represents the repository push.

Setting up the POST service

You can either set up the POST service manually or you can write a service to automate this. You would write a service if you are integrating an application with Bitbucket Server.

Set up in the repository settings

1. Go to the repository's settings.
2. Click **Hooks** in the left-hand navigation.
3. Click **Enable** for the 'Post-Receive Webhooks' item. You can add up to 5 URLs for where Bitbucket Server should send its update messages.
4. Press **Save**.

POST data

When a user pushes to a repository, Bitbucket Server POSTs to the URL you provided. The body of the POST request contains information about the repository where the change originated, a list of recent commits, and the name of the user that made the push.

Example of payload

This is an example of a push that contains one commit that changes 2 files (*pom.xml*) in folders *iridium-common* and *iridium-magma*.

```
JSON Payload
```

```

{
  "repository":{
    "slug":"iridium-parent",
    "id":11,
    "name":"iridium-parent",
    "scmId":"git",
    "state":"AVAILABLE",
    "statusMessage":"Available",
    "forkable":true,
    "project":{
      "key":"IR",
      "id":21,
      "name":"Iridium",
      "public":false,
      "type":"NORMAL",
      "isPersonal":false
    },
    "public":false
  },
  "refChanges":[
    {
      "refId":"refs/heads/master",
      "fromHash":"2c847c4e9c2421d038fff26ba82bc859ae6ebe20",
      "toHash":"f259e9032cdeb1e28d073e8a79a1fd6f9587f233",
      "type":"UPDATE"
    }
  ],
  "changesets":{
    "size":1,
    "limit":100,
    "isLastPage":true,
    "values":[
      {
        "fromCommit":{
          "id":"2c847c4e9c2421d038fff26ba82bc859ae6ebe20",
          "displayId":"2c847c4"
        },
        "toCommit":{
          "id":"f259e9032cdeb1e28d073e8a79a1fd6f9587f233",
          "displayId":"f259e90",
          "author":{
            "name":"jhocman",
            "emailAddress":"jhocman@atlassian.com"
          },
          "authorTimestamp":1374663446000,
          "message":"Updating poms ...",
          "parents":[
            {
              "id":"2c847c4e9c2421d038fff26ba82bc859ae6ebe20",
              "displayId":"2c847c4"
            }
          ]
        }
      ]
    },
    "changes":{
      "size":2,
      "limit":500,
      "isLastPage":true,

```

```

        "values": [
            {
                "contentId": "2f259b79aa7e263f5829bb6e98096e7ec976d998",
                "path": {
                    "components": [
                        "iridium-common",
                        "pom.xml"
                    ],
                    "parent": "iridium-common",
                    "name": "pom.xml",
                    "extension": ".xml",
                    "toString": "iridium-common/pom.xml"
                },
                "executable": false,
                "percentUnchanged": -1,
                "type": "MODIFY",
                "nodeType": "FILE",
                "srcExecutable": false,
                "link": {
                    "url": "/projects/IR/repos/iridium-parent/commits/f259e9032cdeb1e28d073e8a79a1fd6f9587f233#iridium-common/pom.xml",
                    "rel": "self"
                }
            },
            {
                "contentId": "2f259b79aa7e263f5829bb6e98096e7ec976d998",
                "path": {
                    "components": [
                        "iridium-magma",
                        "pom.xml"
                    ],
                    "parent": "iridium-magma",
                    "name": "pom.xml",
                    "extension": ".xml",
                    "toString": "iridium-magma/pom.xml"
                },
                "executable": false,
                "percentUnchanged": -1,
                "type": "MODIFY",
                "nodeType": "FILE",
                "srcExecutable": false,
                "link": {
                    "url": "/projects/IR/repos/iridium-parent/commits/f259e9032cdeb1e28d073e8a79a1fd6f9587f233#iridium-magma/pom.xml",
                    "rel": "self"
                }
            }
        ],
        "start": 0,
        "filter": null
    },
    "link": {

```

```
"url":"/projects/IR/repos/iridium-parent/commits/f259e9032cdeb1e28d073e8
a79a1fd6f9587f233#iridium-magma/pom.xml",
    "rel":"self"
  }
},
"start":0,
```

```

    "filter":null
  }
}

```

Properties

Some of the system-wide properties for the Webhook Plugin can be overridden in the Bitbucket Server configuration file. The available properties are listed in [Bitbucket Server config properties](#).

Audit logging in Bitbucket Server

Bitbucket Server comes with an internal audit system enabled by default at installation. The audit system is intended to give administrators an insight into the way Bitbucket Server is being used. The audit system could be used to identify authorized and unauthorized changes, or suspicious activity over a period of time.

Viewing recent events

Bitbucket Server administrators and system administrators can see a list of recent events for each project and repository in the 'Audit log' view. This is found in the 'Settings' for a project or repository, and shows only the most important audit events.

Audit log		
Shows the most important audit events affecting this repository (up to a maximum of the 500 most recent events)		
User	Action	Details
 Admin Istrator	RestrictedRefAddedEvent	{"id":1,"value":"auipugin-","users":["admin","eparis","hjennings","jevans","rlee"]}
 Admin Istrator	RepositoryPermissionGrantedEvent	{"permission":"REPO_WRITE","group":"stash-users"}
 Admin Istrator	RepositoryModifiedEvent	{"old.name":"booking engine","new.name":"Apollo UI"}
 Admin Istrator	RepositoryModifiedEvent	{"old.name":"TeamsInSpace","new.name":"booking engine"}
 Admin Istrator	RepositoryCreatedEvent	{"project":"AIS","repository":"teamsinspace"}

The audit log displays a subset of the events recorded in the log file and is kept to a configurable maximum size (the default is 500 events). See [Audit events in Bitbucket Server](#) for more details.

Accessing the audit log file

The full audit log file records a wide range of events in Bitbucket Server. See [Audit events in Bitbucket Server](#) for a list of these.

The volume of events that are logged is coarsely configurable by changing a Bitbucket Server instance setting. See [Bitbucket Server config properties](#) for more details.

You can find the log file in the `<Bitbucket Server home directory>/log/audit` directory.

The log file will roll daily and also when it grows past a maximum size of 25 MB. There is a limit (currently 100) to the number of rolled files that Bitbucket Server will keep. When the limit is reached, the oldest file is deleted each day.

Note that you will need to backup the log files before they are removed, if your organization needs to keep copies of those.

Configuring audit logging

There are [various system properties](#) that can be used to configure audit logging in Bitbucket Server.

Audit events in Bitbucket Server

The auditing component of Bitbucket Server will log many different events that occur when Bitbucket Server is being used. The events have been assigned priorities based on how important they are – these priorities can be used to control how much information is added to the audit log file. For example, if you have a server under high load and no need for auditing, you may wish to turn audit logging off by setting it to `NONE` - see the [audit log config properties](#).

On this page:

- [Server level events](#)
- [User management events](#)
- [Permission events](#)
- [Project events](#)
- [Repository events](#)
- [Pull request events](#)
- [Plugin events](#)
- [SSH key events](#)

Server level events

Event	Description	Priority
ApplicationConfigurationChangedEvent	The server configuration has changed e.g. the display name or the base url .	HIGH
BackupEvent	Audited at the beginning and the end of a system backup .	HIGH
LicenseChangedEvent	The server license has changed.	HIGH
MailHostConfigurationChangedEvent	The servers mail host has changed (used to send email notifications).	HIGH
MigrationEvent	Audited at the beginning and the end of a database migration .	HIGH
ServerEmailAddressChangedEvent	The server email address has changed (used in email notifications).	HIGH
TicketRejectedEvent	Certain resources (e.g. the Git processes) are throttled, when tickets are rejected (e.g. too many Git processes are in use) this event is fired.	LOW

User management events

Event	Description	Priority
DirectoryCreatedEvent	Occurs when a new directory is created.	HIGH
DirectoryDeletedEvent	Occurs when a new directory is deleted.	HIGH
GroupCreatedEvent	Occurs when a new group is created in the internal directory.	HIGH
GroupUpdatedEvent	Occurs when a new group is updated (not when membership changes) in the internal directory.	HIGH

GroupDeletedEvent	Occurs when a new group is deleted from the internal directory.	HIGH
GroupMembershipCreatedEvent	Occurs when a user is added to a group in the internal directory.	HIGH
GroupMembershipDeletedEvent	Occurs when a user is removed from a group in the internal directory.	HIGH
UserAuthenticatedEvent	Occurs when a user is successfully authenticated (logged in).	LOW
UserAuthenticationFailedInvalidAuthenticationEvent	Occurs whenever a user fails to authenticate. Note that this can occur frequently in Bitbucket Server whenever a command line CLI is used as the initial URL provided to Bitbucket Server contains a username but no password, which is rejected by Crowd.	MEDIUM
UserCreatedEvent	Occurs when a user is created in the internal directory.	HIGH
UserCredentialUpdatedEvent	Occurs when a user changes password in the internal directory.	HIGH
UserDeletedEvent	Occurs when a user is deleted from the internal directory.	HIGH
UserRenamedEvent	Occurs when the username of a user is changed in the internal directory.	HIGH

Permission events

 in the table below indicates that the event is visible in the recent audit log screen for the project or repository.

Event	Description	Priority
GlobalPermissionGrantedEvent	Occurs when a user or group is granted a global permission (e.g. create project).	HIGH
GlobalPermissionRevokedEvent	Occurs when a user or group has a global permission revoked.	HIGH
ProjectPermissionGrantedEvent	Occurs when a user or group is granted a permission for a specific project. 	HIGH
ProjectPermissionRevokedEvent	Occurs when a user or group has a permission for a specific project revoked. 	HIGH
RepositoryPermissionEvent	Occurs when a user or group has a permission for a specific repository altered. 	HIGH
RestrictedRefEvent	Children of this event are fired when a restricted ref is altered. 	HIGH

Project events

 in the table below indicates that the event is visible in the recent audit log screen for the project.

Event	Description	Priority
ProjectAvatarUpdatedEvent	Raised when a project avatar has been successfully updated.	LOW
ProjectCreatedEvent	Raised when a project is created. 	HIGH
ProjectCreationRequestedEvent	Raised just before a project is created; can be canceled.	LOW
ProjectModifiedEvent	Raised when a project has been successfully updated (e.g. the project name). 	HIGH
ProjectModificationRequestedEvent	Raised just before a project is updated; can be canceled.	LOW
ProjectDeletedEvent	Raised when a project is deleted.	HIGH
ProjectDeletionRequestedEvent	Raised just before a project is deleted; can be canceled.	LOW

Repository events

 in the table below indicates that the event is visible in the recent audit log screen for the project or repository.

Event	Description	Priority
RepositoryAccessedEvent	Raised when a repository is accessed by a user. Bitbucket Server currently only fires this event selectively - when users hit a repository page.	LOW
RepositoryCreatedEvent	Raised when a repository is created. 	MEDIUM
RepositoryCreationFailedEvent	Raised when an attempt to create a repository fails.	LOW
RepositoryCreationRequestedEvent	Raised just before a repository is created; can be canceled.	LOW
RepositoryForkedEvent	Raised when a repository is forked successfully. 	MEDIUM
RepositoryForkFailedEvent	Raised when an attempt to fork a repository fails.	LOW
RepositoryForkRequestedEvent	Raised just before a repository is forked; can be canceled.	LOW
RepositoryDefaultBranchModifiedEvent	Raised when the default branch of a repository is reconfigured (typically through repository settings).	LOW
RepositoryDeletedEvent	Raised when a repository is deleted. 	HIGH
RepositoryDeletionRequestedEvent	Raised just before a repository is deleted; can be canceled.	LOW

RepositoryOtherReadEvent	Raised when the server uploads a pack file to the client via HTTP.	LOW
RepositoryOtherWriteEvent	Raised when the server receives a pack file from the client via HTTP.	LOW
RepositoryPullEvent	Raised when a Git client pulls from a repository (only when new content is sent to the client).	LOW
RepositoryPushEvent	Raised when a Git client pushed to a repository.	LOW

Pull request events

Event	Description	Priority
PullRequestEvent	Fired at different points in the pull request lifecycle (declined, merged, opened, reopened, rescoped [code updated], updated, approved, unapproved, participants updated).	LOW

Plugin events

See this [plugin documentation](#) for details of when these events below are triggered.

Event	Description	Priority
PluginDisabledEvent	Occurs when a plugin has been disabled, either by the system or a user.	MEDIUM
PluginEnabledEvent	Occurs when a plugin has been enabled, either by the system or a user.	MEDIUM
PluginModuleDisabledEvent	Occurs when a plugin module has been disabled, either by the system or a user.	MEDIUM
PluginModuleEnabledEvent	Occurs when a plugin module has been enabled, either by the system or a user.	MEDIUM
PluginModuleUnavailableEvent	Signifies a plugin module is now unavailable outside the usual installation process.	MEDIUM
PluginUninstalledEvent	Occurs when a plugin is explicitly uninstalled (as opposed to as part of an upgrade).	MEDIUM
PluginUpgradedEvent	Signifies that a plugin has been upgraded at runtime.	MEDIUM
PluginContainerUnavailableEvent	Occurs when the container of a plugin is being shutdown, usually as a result of the server being stopped.	LOW
PluginModuleAvailableEvent	Signifies that a plugin module is now available outside the usual installation process.	LOW
PluginFrameworkStartedEvent	Signifies that the plugin framework has been started and initialized.	LOW

SSH key events

 in the table below indicates that the event is visible in the recent audit log screen for the project or

repository.

Event	Description	Priority
SshKeyCreatedEvent	Occurs when: - an SSH key is added for a user, or - an access key is added to a project or repository and the key has not yet been used on any other projects or repositories.	HIGH
SshKeyDeletedEvent	Occurs when: - an SSH key is removed from a user, or - an access key is removed from a project or repository and it is no longer being used by any other projects or repositories.	HIGH
SshKeyAccessGrantedEvent	Occurs when an access key is given access to a project or repository. 	HIGH
SshKeyAccessRevokedEvent	Occurs when an access key is removed from a project or repository. 	HIGH

Advanced actions

This section describes the administrative actions that can be performed from outside of the Bitbucket Server Administration user interface.

In this section:

- [Running the Bitbucket Server installer](#)
- [Automated setup for Bitbucket Server](#)
- [Starting and stopping Bitbucket Server](#)
- [Install Bitbucket Server from an archive file](#)
- [Install and configure a remote Elasticsearch instance](#)
- [Running Bitbucket Server as a Linux service](#)
- [Running Bitbucket Server as a Windows service](#)
- [Bitbucket Server config properties](#)
- [Proxying and securing Bitbucket Server](#)
- [Enabling SSH access to Git repositories in Bitbucket Server](#)
- [Using diff transcoding in Bitbucket Server](#)
- [Changing the port that Bitbucket Server listens on](#)
- [Moving Bitbucket Server to a different context path](#)
- [Running Bitbucket Server with a dedicated user](#)
- [Bitbucket Server debug logging](#)
- [Data recovery and backups](#)
- [Lockout recovery process](#)
- [Scaling Bitbucket Server](#)
- [High availability for Bitbucket Server](#)
- [Clustering with Bitbucket Data Center](#)
- [Enabling JMX counters for performance monitoring](#)
- [Getting started with Bitbucket Server and AWS](#)
- [Disabling HTTP\(S\) access to Git repositories in Bitbucket Server](#)
- [Smart Mirroring](#)
- [Git Large File Storage](#)
- [Copy of Bitbucket Server config properties](#)

Related pages:

- [Administering Bitbucket Server](#)
- [Supported platforms](#)
- [Bitbucket Server FAQ](#)

Running the Bitbucket Server installer

This page provides information about running the Bitbucket Server with search installer. For high-level information about installing and using Bitbucket Server see [Getting started](#).

Installers are available for Linux, Mac OS X and Windows operating systems.

The installer will:

- Install Bitbucket Server into a fresh directory, even if you have an earlier version of Bitbucket Server installed.
- Install a supported version of the Java JRE, which is only available to Bitbucket Server, if necessary.
- Install an bundled, local instance of Elasticsearch.
- Launch Bitbucket Server when it finishes.

Additional services provided by the installer, and described on this page, are:

- [Installing Bitbucket Server as a service](#)
- [Running the installer in console and unattended modes](#)

You can also automate the Bitbucket Server Setup Wizard so that a Bitbucket Server instance can be completely provisioned automatically – see [Automated setup for Bitbucket Server](#).

Running the installer

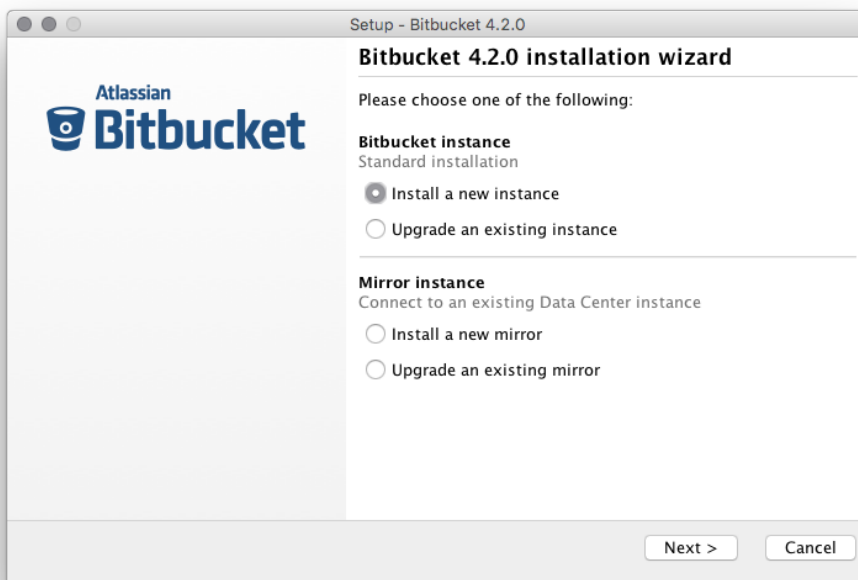
Download the Bitbucket Server installer from the Atlassian download site.

▼ [For Linux...](#)

On Linux, you need to set the executable flag on the installer file before running it:

```
chmod +x atlassian-bitbucket-x.x.x-x64.bin
```

Run the installer, and follow the installation wizard.



Install Bitbucket Server as a service

On Linux and Windows systems, the installer can install Bitbucket Server as a service (although not when upgrading an existing instance of Bitbucket Server).

A service account named 'atlbitbucket' will be created.

On Linux

- The 'atlbitbucket' account will be a locked account (it cannot be used to log in to the system).
- The `init.d` script will be linked to run levels 2, 3, 4 and 5. If you wish to change this, you will need to configure it [manually](#).

On Windows

- The installer generates a password for the service account. As a Windows administrator, you can update the account password if you wish to own the account. You'll also need to update the log on credentials for the service.
- The 'atlbitbucket' account will be configured with `SeServiceLogonRight` so that it can be used by the service. It will also be configured with `SeDenyBatchLogonRight`, `SeDenyInteractiveLogonRight`, `SeDenyNetworkLogonRight`, and `SeDenyRemoteInteractiveLogonRight` so that it cannot be used to log into the machine.
- For Windows services created using the Bitbucket Server installer, the [Bitbucket Server home directory](#) location (defined by the `BITBUCKET_HOME` variable) is configured as a Tomcat Service JVM option. To change it see [Change BITBUCKET_HOME when installed as a Windows service](#).

Console and unattended mode

The Bitbucket Server installer has three modes:

- GUI mode: the default mode for the installer is to display a GUI installer.
- Console mode: if the installer is invoked with the `-c` argument, the interaction with the user is performed in the terminal from which the installer was invoked.
- Unattended mode: if the installer is invoked with the `-q` argument, there is no interaction with the user and the installation is performed automatically with the default values.

Unattended mode also allows you to supply a response file with a `-var file` option, to supply answers for all questions that are used instead of the defaults. An example response file is:

Example response file

```
// Should Bitbucket Server be installed as a Service? Must be ADMIN
// (default: true if the process is running with administrator rights,
// false otherwise). If false, the home and installation directories must
// be specified to point to directories owned by the user
app.install.service$Boolean=true

// The ports Bitbucket Server should bind to (defaults:
// portChoice=default, httpPort=7990, serverPort=8006)
portChoice=custom
httpPort=7990
serverPort=8006

// Path to the Bitbucket Server HOME directory (default:
// /var/atlassian/application-data/bitbucket if the process is running with
// administrator rights, ~/atlassian/application-data/bitbucket otherwise)
app.bitbucketHome=/var/atlassian/application-data/bitbucket

// The target installation directory (default:
// /opt/atlassian/bitbucket/<VERSION> if the process is running with
// administrator rights, ~/atlassian/bitbucket/<VERSION> otherwise)
app.defaultInstallDir=/opt/atlassian/bitbucket/<VERSION>
```

On Windows, you must tell CMD/PowerShell to wait for the install4j process to use console/unattended mode:

```
start /wait installer.exe -c
```

On Mac OS X, mount the disk image, then run the Java stub in the installer using this command:

```
/Volumes/Bitbucket Server/Bitbucket Server\
X.X.X\Installer.app/Contents/MacOS/JavaApplicationStub -options
```

where X.X.X is the version of Bitbucket Server, and -options can include -c or -q, and -varfile followed by the path to the response file.

For more information see the [install4j documentation](#).

Further reading

- [Using Bitbucket Server in the enterprise](#)

Automated setup for Bitbucket Server

This page describes how the Bitbucket Server Setup Wizard can be completed automatically, that is without the need for manual interaction in the browser. See [Getting started](#) for an outline of the tasks that the Setup Wizard assists with when setting up Bitbucket Server manually.

You might want to configure this when automating the provisioning of Bitbucket Data Center, for example. Of

course, you'll need to configure provisioning tools such as Puppet or Vagrant yourself.

Note that you can also automate the 'install and launch' phase of provisioning Bitbucket Server – see [Running the Bitbucket Server installer](#) for information about how to run the installer in console and unattended modes.

1. Get a license for Bitbucket Server

You can get a new Bitbucket Server license by doing one of:

- Logging in to your [My.Atlassian.com](#) account.
- Contacting Atlassian, if you need a Bitbucket Data Center license.

If you already have a licensed instance of Bitbucket Server, you can find the license in the Bitbucket Server admin area.

2. Set the configuration properties

After installing Bitbucket Server, but before you start Bitbucket Server for the first time, edit the `bitbucket.properties` file to add the properties in the table below. Use the standard format for Java properties files.

Note that the `bitbucket.properties` file is created automatically in the `shared` folder of your [Bitbucket home directory](#) when you perform a [database migration](#). Create the file yourself if it does not yet exist. See [Bitbucket Server config properties](#) for information about the properties file.

Add these properties to the `bitbucket.properties` file:

Property	Description
<code>setup.displayName=displayName</code>	The display name for the Bitbucket Server application.
<code>setup.baseUrl= https://bitbucket.yourcompany.com</code>	The base URL.
<code>setup.license=AAAB...</code>	The Bitbucket Server license. Use the <code>\</code> character at the end of each line if you wish to break the license string over multiple lines.
<code>setup.sysadmin.username=username</code>	Credentials for the system admin account.
<code>setup.sysadmin.password=password</code>	
<code>setup.sysadmin.displayName=John Doe</code>	The display name for the system admin account. An empty property is ignored.
<code>setup.sysadmin.emailAddress=sysadmin@yourcompany.com</code>	The email address for the system admin account.
<code>jdbc.driver=org.postgresql.Driver</code>	JDBC connection parameters. Use the appropriate values for your own JDBC connection.
<code>jdbc.url=jdbc:postgresql://localhost:5432/bitbucket</code>	
<code>jdbc.user=bitbucket</code>	
<code>jdbc.password=bitbucket</code>	


```
plugin.mirroring.upstream.url=https://bitbucket.com
pany.com
```

(Smart Mirroring only) On the mirror, specifies the base URL of the primary Bitbucket Data Center instance that the new mirror will be mirroring from. See [Set up a mirror#setup](#) for more information.

You should specify the JDBC properties so that Bitbucket Server can connect to the external database.

If any of the following required properties are not provided in the properties file, when you start Bitbucket Server the Setup Wizard will launch at the appropriate screen so that you can enter values for those properties.

3. Start Bitbucket Server

Start Bitbucket Server as usual. See [Starting and stopping Bitbucket Server](#).

Bitbucket Server reads the `bitbucket.properties` file and applies the setup properties automatically.

When you now visit Bitbucket Server in the browser, you see the welcome page.

Troubleshooting

The Setup Wizard launches in the browser

The Setup Wizard will run if there are missing configuration properties, such as the license string, in the `bitbucket.properties` file. Check the properties file and compare with the table in Step 2 above. Alternatively, the set up can be completed using the web UI.

Write access for the `config.properties` file

Once the automated setup process completes, the relevant properties in the `bitbucket.properties` file are commented out. This requires that the system user has write permission on the properties file.

Bitbucket Server fails to start with a 'Could not acquire change log lock.' error

If Bitbucket Server is forced to quit while modifying the `config.properties` file, you may not be able to restart Bitbucket Server, and `atlassian-bitbucket.log` contains the above error.

See this KB article for information about how to resolve this: [Bitbucket Server Does Not Start - Could not acquire change log lock](#)

Starting and stopping Bitbucket Server

There are a few ways that you can start and stop Bitbucket Server:

- [At install time](#)
- [When Bitbucket Server runs as a service](#)
- [Manually](#)
- [Start Bitbucket Server with a remote Elasticsearch instance](#)

At install time

The Bitbucket Server installer automatically starts Bitbucket Server.

On Windows and Linux systems you can choose to have Bitbucket Server installed as a service.

When Bitbucket Server runs as a service

If Bitbucket Server is installed as a service on Windows or Linux systems, it will be started automatically when the system boots.

Windows

Start and stop the Bitbucket Server service from the services console, on Windows.



For Bitbucket Server 4.6+

Ensure that you start both `atlbitbucket` and `atlbitbucket_search` services.

Linux

Manage the Bitbucket Server service using the following commands:

```
# service atlbitbucket status
# service atlbitbucket stop
# service atlbitbucket start
```

For Bitbucket Server 4.6+

Ensure that the Bitbucket Search service also starts. This should be started at the same time or before the `atlbitbucket` service.

```
# service atlbitbucket_search status
# service atlbitbucket_search start
# service atlbitbucket_search stop
```

Mac OS X

On Mac OS X, you will need to restart Bitbucket Server manually, as described below.

Manually

You can start and stop Bitbucket Server manually as follows:

Windows

Start and stop Bitbucket Server using the **Bitbucket Server** items in the Windows Start menu. Use the URL item there to visit Bitbucket Server in your default browser.

Alternatively, start Bitbucket Server from a command prompt, by changing directory to the `<Bitbucket Server installation directory>` and running the following command:

```
bin\start-bitbucket.bat
```

Stop Bitbucket Server manually by changing directory to the `<Bitbucket Server installation directory>` and running the following command:

```
bin\stop-bitbucket.bat
```

Linux

Start and stop Bitbucket Server manually using the scripts provided.

Start Bitbucket Server by changing directory in a terminal to the `<Bitbucket Server installation directory>` and running:

```
bin/start-bitbucket.sh
```

Stop Bitbucket Server by changing directory in a terminal to the `<Bitbucket Server installation directory>` and running:

directory> and running:

```
bin/stop-bitbucket.sh
```

Mac

Start and stop Bitbucket Server manually using the app icons (shown above) in the <Bitbucket Server installation directory>. These simply link to the `start-bitbucket.sh` and `stop-bitbucket.sh` scripts in <Bitbucket Server installation directory>/bin.

Use the URL icon to visit Bitbucket Server in your default browser.

Start Bitbucket Server with a remote Elasticsearch instance

When using a remote Elasticsearch instance, instead of the bundled Elasticsearch instance, start Bitbucket Server by running `start-webapp.sh` instead of `start-bitbucket.sh`. This starts Bitbucket Server alone without running the bundled Elasticsearch instance.

Install Bitbucket Server from an archive file

This page describes how to manually install Bitbucket Server from an archive file. However, we strongly recommend that you use the [Bitbucket Server installer](#) instead, for a quick and trouble-free install experience.

Related pages

- See [Getting started](#), and consider using the installer
- [Using Bitbucket Server in the enterprise](#)
- [Docker container image for Bitbucket Server](#)

1. Check supported platforms

Check the [Supported platforms](#) page for details of the application servers, databases, operating systems, web browsers and Java and Git versions that we have tested Bitbucket Server with and recommend.

Atlassian only officially supports Bitbucket Server running on x86 hardware and 64-bit derivatives of x86 hardware.

Cygwin Git is *not supported*. No internal testing is done on that platform, and many aspects of Bitbucket Server's functionality (pull requests and forks among them) have known issues. When running Bitbucket Server on Windows, *always* use msysGit.

2. Check your version of Java

In a terminal or command prompt, run this:

```
java -version
```

The version of Java should be **1.8.x**. You'll need a 64-bit version of Java if you have a 64-bit operating system.

▼ [On Linux, if you don't see a supported version, then get Java...](#)

Install Java

Download Java Server JRE from [Oracle's website](#), and install it.

Now try running 'java -version' again to check the installation. The version of Java should be **1.8.x**.

Check that the system can find Java

In a terminal, run this:

```
echo $JAVA_HOME
```

You should see a path like /usr/jdk/jdk1.8.0.

If you don't see a path, then set JAVA_HOME

Do one of the following:

- If JAVA_HOME is not set, log in with 'root' level permissions and run:

```
echo JAVA_HOME="path/to/JAVA_HOME" >> /etc/environment
```

where path/to/JAVA_HOME may be like: /usr/jdk/jdk1.8.0

- If JAVA_HOME needs to be changed, open the /etc/environment file in a text editor and modify the value for JAVA_HOME to:

```
JAVA_HOME="path/to/JAVA_HOME"
```

It should look like: /usr/jdk/jdk1.8.0

▼ On Mac OS X, if you don't see a supported version, then get Java...

Install Java

Download Java Server JRE from [Oracle's website](#), and install it.

Now try running 'java -version' again to check the installation. The version of Java should be **1.8.x**.

Check that the system can find Java

In a terminal, run this:

```
echo $JAVA_HOME
```

You should see a path like /System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/.

If you don't see a path, then set JAVA_HOME

Open your ~/.profile file in a text editor and insert:

```
JAVA_HOME="path/to/JAVA_HOME"
export JAVA_HOME
```

where path/to/JAVA_HOME may be like: /System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/

Refresh your ~/.profile in the terminal and confirm that JAVA_HOME is set:

```
source ~/.profile
$JAVA_HOME/bin/java -version
```

You should see a version of Java that is **1.8.x**, like this:

```
java version "1.8.0_1"
```

▼ On Windows, if you don't see a supported version, then get Java...

Install Java

Download Java Server JRE from [Oracle's website](#), and install it.

Now try running 'java -version' again to check the installation. The version of Java should be **1.8.x**.

Check that the system can find Java

Bitbucket Server uses the JAVA_HOME environment variable to find Java. To check that, in a command prompt, run:

```
echo %JAVA_HOME%
```

You should see a path to the root directory of the Java installation. When running Bitbucket Server on Windows, unlike Linux or Unix, JAVA_HOME paths with spaces are just fine.

If you don't see a path, then set JAVA_HOME

[Windows 7 ...](#)**Stage 1. Locate the JRE installation directory**

If you already know the installation path for the Java Runtime Environment, go to *Stage 2* below. Otherwise, find the installation path by following these instructions:

1. If you didn't change the installation path for the Java Runtime Environment during installation, it will be in a directory under C:\Program Files\Java. Using Explorer, open the directory C:\Program Files\Java.
2. Inside that path will be one or more subdirectories such as C:\Program Files\Java\jre8.

Stage 2. Set the JAVA_HOME variable

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "JAVA_HOME" as the **Variable name**, and the absolute path to where you installed Java as the **Variable value**. Don't use a trailing backslash, and don't wrap the value in quotes.

Now, in a *new command prompt*, try running '%JAVA_HOME%\bin\java -version'. You should see the same version of Java as you saw in 2. above.

▼ Windows Server 2003 R2 ...

Stage 1. Locate the JRE installation directory

If you already know the installation path for the Java Runtime Environment, go to *Stage 2* below. Otherwise, find the installation path by following these instructions:

1. If you didn't change the installation path for the Java Runtime Environment during installation, it will be in a directory under C:\Program Files\Java. Using Explorer, open the directory C:\Program Files\Java.
2. Inside that path will be one or more subdirectories such as C:\Program Files\Java\jre8.

Stage 2. Set the JAVA_HOME variable

Once you have identified the JRE installation path:

1. Right-click the **My Computer** icon on your desktop and select **Properties**.
2. Click the **Advanced** tab.
3. Click the **Environment Variables** button.
4. Under **System Variables**, click **New**.
5. Enter the **variable name** as JAVA_HOME.
6. Enter the **variable value** as the installation path for the Java Development Kit. Don't use a trailing backslash, and don't wrap the value in quotes.
 - If your Java installation directory has a space in its path name, you should use the shortened path name (e.g. C:\Progra~1\Java\jre7) in the environment variable instead.

Note for Windows users on 64-bit systems

Progra~1 = 'Program Files'
Progra~2 = 'Program Files(x86)'

7. Click **OK**.
8. Click **Apply Changes**.
9. Close any command window which was open before you made these changes, and open a new command window. There is no way to reload environment variables from an active command prompt. If the changes do not take effect even after reopening the command window, restart Windows.

Now, in a *new command prompt*, try running '%JAVA_HOME%\bin\java -version'. You should see the same version of Java as you saw in 2. above.

3. Check your versions of Git and Perl

In a terminal or command prompt, run:

```
git --version
perl --version
```

The version of Git should be **1.8.x** or higher. The version of Perl should be **5.8.8** or higher.

If you don't see supported versions of Git and Perl, either install or upgrade them – see [Installing and upgrading Git](#).

4. Now it's time to get Bitbucket Server

Download [Bitbucket Server](#) from the Atlassian download site.

Download latest version ➞

Looking for the [Bitbucket Server WAR file?](#)

Extract the downloaded file to an install location (without spaces in the path).

The path to the extracted directory is referred to as the `<Bitbucket Server installation directory>` in these instructions.

Never unzip the Bitbucket Server archive file over the top of an existing Bitbucket Server installation – each version of Bitbucket Server includes versioned jar files, such as `bitbucket-model-4.0.0.jar`. If you copy these, you end up with multiple versions of Bitbucket Server's jar files in the classpath, which leads to runtime corruption.

Note that you should use the same user account to both extract Bitbucket Server and to run Bitbucket Server (in Step 6.) to avoid possible permission issues at startup. For production installations, we recommend that you create a new dedicated user that will run Bitbucket Server on your system. See [Running Bitbucket Server with a dedicated user](#).

5. Tell Bitbucket Server where to store your data

The Bitbucket Server [home directory](#) is where your Bitbucket Server data is stored.

If you are upgrading Bitbucket Server, simply update the value of `BITBUCKET_HOME` in the `<Bitbucket Server installation directory>/bin/setenv` file so the *new* Bitbucket Server installation points to your *existing* Bitbucket Server [home directory](#) (if you use a `BITBUCKET_HOME` environment variable to specify the home directory location, no change is required).

Otherwise, for a new install, create your Bitbucket home directory (without spaces in the name), and then tell Bitbucket Server where you created it by editing the `<Bitbucket Server installation directory>/bin/set-bitbucket-home.sh` (or `set-bitbucket-home.bat`) file – uncomment the `BITBUCKET_HOME` line and add the absolute path to your home directory. Here's an example of what that could look like when you're done:

```
#
if ["x${BITBUCKET_HOME}" = "x"]; then
    export BITBUCKET_HOME="/home/username/bitbucket_home"
fi
```

You *should not* locate your Bitbucket home directory inside the `<Bitbucket Server installation directory>` — they should be entirely separate locations. If you do put the home directory in the `<Bitbucket Server installation directory>` it may be overwritten, and lost, when Bitbucket Server gets upgraded. And by the way, you'll need separate Bitbucket Server home directories if you want to run multiple instances of Bitbucket Server.

▼ [Click here for Windows notes...](#)

Tell Bitbucket Server where you created it by setting a `BITBUCKET_HOME` environment variable, for Windows 7, as follows:

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "`BITBUCKET_HOME`" as the **Variable name**, and the absolute path to your Bitbucket home directory as the **Variable value**. Don't use a trailing backslash.

There are a few things to know about setting up the Bitbucket home directory on Windows that will make life easier:

- Keep the path length to the Bitbucket home directory as short as possible. See [Bitbucket Server always shows incorrect Merge Conflict in PRs](#) for an explanation.
- Don't use spaces in the path to the Bitbucket home directory.

6. Move server.xml to your Bitbucket Server home shared directory

If this is a new installation, or you are already running Stash 3.8 or above, you can [skip to the next step](#).

If you are upgrading from Stash 3.7 or earlier and you made any changes to `<Bitbucket Server installation directory>/conf/server.xml` (for instance to [secure your server with SSL](#)):

1. In the `<BITBUCKET_HOME>` directory, make a new directory called `shared`.
2. Then, copy your modified `server.xml` file into `<BITBUCKET_HOME>/shared/`. Ensure the copied file is readable by the [user account that runs Bitbucket Server](#).

7. (Optional) Install and configure a remote Elasticsearch instance

Bitbucket Server 4.5+ comes with an bundled [Elasticsearch](#) instance, which runs as a separate process from the Bitbucket Server application, and does not require any extra configuration.

If you plan to use the bundled Elasticsearch instance, jump to the next step, [Start Bitbucket Server!](#)

However, you can also install Elasticsearch on a remote machine, which can provide some advantages allocating memory resources. Read the instructions for installing and configuring a remote Elasticsearch instance here: [Install and configure a remote Elasticsearch instance](#).

Elasticsearch ports

Bitbucket Server's bundled Elasticsearch instance requires ports 7992 and 7993 be available to provide code search functionality. This is not configurable, so ensure these ports are available.

8. Start Bitbucket Server!

There are a couple of ways in which you can start Bitbucket Server – see [Starting and stopping Bitbucket Server](#).

If you've setup a remote Elasticsearch instance you probably do not want to start the bundled Elasticsearch instance.

To start Bitbucket Server with a remote Elasticsearch instance

When using a remote Elasticsearch instance, instead of the bundled Elasticsearch instance, start Bitbucket Server by running `start-webapp.sh` instead of `start-bitbucket.sh`. This starts Bitbucket Server alone without running the bundled Elasticsearch instance.

Finish configuring Bitbucket Server

Now, in your browser, go to <http://localhost:7990> and run through the Setup Wizard. In the Setup Wizard:

- If you're evaluating Bitbucket Server, select **Internal** at the 'Database' step. Bitbucket Server will use its internal database, and you can easily migrate to external database later. See [Connecting Bitbucket Server to an external database](#).
- Enter your Bitbucket Server license key.
- [Set the base URL for Bitbucket Server](#).
- Set up an administrator account.
- You can set up JIRA Software integration, but you can do this later if you wish. See [Configuring JIRA integration in the Setup Wizard](#).

9. Set up your mail server

Configure your email server so users can receive a link from Bitbucket Server that lets them generate their own passwords. See [Setting up your mail server](#).

10. Add users and repositories

Now is the time to set up your users in Bitbucket Server, and to tell Bitbucket Server about any existing repositories you have. Please see the following pages for the details:

- [Getting started with Git and Bitbucket Server](#)
- [Importing code from an existing project](#)

Additional steps for production environments

For production or enterprise environments we recommend that you configure the additional aspects described on [Using Bitbucket Server in the enterprise](#). The aspects described there are not necessary when you are installing for evaluation purposes only.

If you wish to install Bitbucket Server as a service on Linux or Windows, see either of:

- [Running Bitbucket Server as a Linux service](#)
- [Running Bitbucket Server as a Windows service](#)

Stopping Bitbucket Server

See [Starting and stopping Bitbucket Server](#).

Uninstalling Bitbucket Server

To uninstall Bitbucket Server, stop Bitbucket Server as described above and then delete the <Bitbucket Server installation directory> and [Bitbucket Server home directory](#).

Install and configure a remote Elasticsearch instance

Bitbucket Server 4.6+ comes with an bundled [Elasticsearch](#) instance, which runs as a separate process from the Bitbucket Server application and doesn't require any extra configuration. You can also install a remote Elasticsearch instance on a remote machine, which can provide some advantages allocating memory resources.



This page describes

- How to [install and configure Elasticsearch on a remote machine](#).
- How to [migrate from the bundled Elasticsearch instance to a remote Elasticsearch instance](#).

Install Elasticsearch on a remote machine

We don't provide specific instructions for installing Elasticsearch, but a good place to start is the [Elasticsearch installation instructions](#). Elastic provides installation packages in several different formats on their [downloads page](#).

Elasticsearch ports

Bitbucket Server's bundled Elasticsearch instance requires ports 7992 and 7993 be available to provide code search functionality. This is not configurable, so ensure these ports are available.

Configure your remote Elasticsearch instance

The `elasticsearch.yml` file contains configuration details for your Elasticsearch instance.

To configure your remote Elasticsearch instance to be compatible with Bitbucket Server

1. Locate the `elasticsearch.yml` file within the [configuration path of your Elasticsearch](#) instance.
2. Add these parameters to your `elasticsearch.yml` file

Parameter	Value
<code>index.mapper.dynamic</code>	<code>false</code>
<code>action.auto_create_index</code>	<code>false</code>

Secure your remote Elasticsearch instance

After you've completed installing Bitbucket Server, you need to secure access to your remote Elasticsearch instance with a username and password. Atlassian recommends securing your remote Elasticsearch instance with a security plugin that requires anyone connecting to it provides authentication credentials. Atlassian provides a free plugin called Buckler for this purpose. Bitbucket Server also supports authentication to Elasticsearch through other plugins that provide basic authentication, like Elastic's [Shield plugin](#).

To secure your remote Elasticsearch instance with the Buckler plugin

1. Download the Buckler plugin on the remote machine: [buckler-plugin-0.2.2.zip](#).
2. Install the plugin onto your remote Elasticsearch instance using the plugin helper in the Elasticsearch `/bin` directory:

```
cd /path/to/elasticsearch/bin
./plugin install -b
https://maven.atlassian.com/content/groups/public/com/atlassian/
elasticsearch/buckler-plugin/0.2.2/buckler-plugin-0.2.2.zip
```

3. Configure basic authentication (at minimum) using the Buckler authentication plugin.
 - a. Within the `elasticsearch/config` directory, create a directory called `buckler`.
 - b. Create a file named `buckler.yml`. By default there is no configuration, which means all the features of the Buckler plugin are disabled.
 - c. Enable basic authentication by adding the following properties, modifying the parameters for your company's details.
 - ▼ [Here's an explanation of all the parameters you can configure...](#)

Parameter	Value	Description
<code>auth.basic.http.enabled:</code>	<code>true</code>	Enables basic authentication for HTTP.
<code>auth.basic.tcp.enabled:</code>	<code>true</code>	Enables basic authentication for TCP.
<code>auth.basic.username:</code>	<code><username></code>	Username to access Elasticsearch instance.
<code>auth.basic.password:</code>	<code><password></code>	Password to access Elasticsearch instance.
<code>tls.http.enabled:</code>	<code>true</code>	Enables TLS for HTTP.
<code>tls.tcp.enabled:</code>	<code>true</code>	Enables TLS for TCP.
<code>tls.keystore.path:</code>	<code><path/to/keystore></code>	Absolute filesystem path to the keystore.

<code>tls.keystore.password:</code>	<code><keystorepassword></code>	Password for accessing the keystore.

▼ [Here's an example config which enables HTTP authentication...](#)

```
auth.basic.http.enabled: true
auth.basic.username: admin
auth.basic.password: basicpassword
```

▼ [Here's an example config which enables every feature...](#)

```
auth.basic.http.enabled: true
auth.basic.tcp.enabled: true
auth.basic.username: admin
auth.basic.password: basicpassword
tls.http.enabled: true
tls.tcp.enabled: true
tls.keystore.path: /path/to/keystore
tls.keystore.password: keystorepassword
```

Connect your Elasticsearch instance to Bitbucket Server

Once you've configured your Elasticsearch instance you need to connect it to Bitbucket Server.

To connect your remote Elasticsearch instance to Bitbucket Server via the `bitbucket.properties` file

Another way to configure connection settings is via `bitbucket.properties` file (resides in `$BITBUCKET_HOME/shared`).

If a parameter is set in the properties file, *it cannot be edited later from the admin UI*. Any changes that need to be made to the Elasticsearch configuration must be made within the `bitbucket.properties` file.

1. In the `$BITBUCKET_HOME` directory, create a new directory called `shared`.
2. In the `$BITBUCKET_HOME/shared` directory, create a text file named `bitbucket.properties`.
3. Add these Elasticsearch properties to the file.

Property	Parameter name for properties file
URL	<code>plugin.search.elasticsearch.baseurl</code>
Username	<code>plugin.search.elasticsearch.username</code>
Password	<code>plugin.search.elasticsearch.password</code>

▼ [Here's a configuration example...](#)

```
plugin.search.elasticsearch.baseurl=http://localhost:9200/
plugin.search.elasticsearch.username=username
plugin.search.elasticsearch.password=password
```

4. Save and close the file.
5. [Start Bitbucket Server](#).

Configure details of your Elasticsearch instance within Bitbucket Server

To configure the details of your Elasticsearch instance within the Bitbucket Server UI, go to the Administration settings page, then click **Server Settings**. At the bottom of the page is where you can configure the details of your Elasticsearch instance.

If a parameter is set in the `bitbucket.properties` file, *it cannot be edited later from the admin UI*. Any changes that need to be made to the Elasticsearch configuration must be made within the `bitbucket.properties` file.

Search

Elasticsearch URL*	http://localhost:7992/
	Base URL of the Elasticsearch instance (defaults to <code>http://localhost:7992</code>)
Elasticsearch username	bitbucket
	Username for connecting to the Elasticsearch instance
Elasticsearch password	
	Password for connecting to the Elasticsearch instance

Migrate from using the bundled Elasticsearch instance to using a remote Elasticsearch instance

To begin using a remote instance of Elasticsearch after having previously installed a standard instance of Bitbucket Server, which uses the bundled Elasticsearch instance:

1. [Install the remote Elasticsearch instance.](#)
2. [Configure the remote Elasticsearch instance to work with Bitbucket Server.](#)
3. [Secure the remote Elasticsearch instance.](#)
4. [Configure the remote Elasticsearch instance details within Bitbucket Server.](#)

To start Bitbucket Server when using a remote Elasticsearch instance, use `start-webapp.sh`

Once your remote Elasticsearch instance is installed, configured, and secured, you will start Bitbucket Server by running `start-webapp.sh` instead of `start-bitbucket.sh`. This starts Bitbucket Server without running the bundled Elasticsearch instance.

Running Bitbucket Server as a Linux service

- The Bitbucket Server installer for Linux installs Bitbucket Server as a service – see [Getting started](#). The information on this page only applies if you are manually installing or upgrading Bitbucket Server.
- System administration tasks are [not supported by Atlassian](#). These instructions are only provided as a guide and may not be up to date with the latest version of your operating system.

For production use on a Linux server, Bitbucket Server should be configured to run as a Linux service, that is, as a daemon process. This has the following advantages:

- Bitbucket Server can be automatically restarted when the operating system restarts.
- Bitbucket Server can be automatically restarted if it stops for some reason.
- Bitbucket Server is less likely to be accidentally shut down, as can happen if the terminal Bitbucket Server was manually started in is closed.
- Logs from the Bitbucket Server JVM can be properly managed by the service.

This page describes the following approaches to running Bitbucket Server as a service on Linux:

- Use the [Java Service Wrapper](#), which allows a Java application to be run as a UNIX daemon.
- Use an [init.d script](#) to start Bitbucket Server at boot time - this doesn't restart Bitbucket Server if it stops for some reason.
- Use a [systemd unit file](#) to start Bitbucket Server at boot time - this doesn't restart Bitbucket Server if it stops for some reason.

Note that Bitbucket Server assumes that the external database is available when it starts; these approaches do not support service dependencies, and the startup scripts will not wait for the external database to become available.

On this page:

- [Using the Java Service Wrapper](#)
- [Using an init.d script](#)
 - [Running on system boot](#)
- [Using a systemd unit file](#)

Related pages:

- [Install Bitbucket Server from an archive file](#)

Using the Java Service Wrapper

Bitbucket Server can be run as a service on Linux using the [Java Service Wrapper](#). The Service Wrapper is [known to work with](#) Debian, Ubuntu, and Red Hat.

The Service Wrapper provides the following benefits:

- Allows Bitbucket Server, which is a Java application, to be run as a service.
- No need for a user to be logged on to the system at all times, or for a command prompt to be open and running on the desktop to be able to run Bitbucket Server.
- The ability to run Bitbucket Server in the background as a service, for improved convenience, system performance and security.
- Bitbucket Server is launched automatically on system startup and does not require that a user be logged in.
- Users are not able to stop, start, or otherwise tamper with Bitbucket Server unless they are an administrator.
- Can provide advanced failover, error recovery, and analysis features to make sure that Bitbucket Server has the maximum possible uptime.

Please see <http://wrapper.tanukisoftware.com/doc/english/launch-nix.html> for wrapper installation and configuration instructions.

The service wrapper supports the standard commands for SysV init scripts, so it should work if you just create a symlink to it from `/etc/init.d`.

Using an init.d script

The usual way on Linux to ensure that a process restarts at system restart is to use an init.d script. This approach does not restart Bitbucket Server if it stops by itself.

1. [Stop Bitbucket Server](#) .
2. Create a bitbucket user, set the permissions to that user, create a home directory for Bitbucket Server and create a symlink to make upgrades easier:


```
$> curl -OL
http://downloads.atlassian.com/software/bitbucket/downloads/atlassi
an-bitbucket-X.Y.Z.tar.gz
$> tar xz -C /opt -f atlassian-bitbucket-X.Y.Z.tar.gz
$> ln -s /opt/atlassian-bitbucket-X.Y.Z
/opt/atlassian-bitbucket-latest

# Create a home directory
$> mkdir /opt/bitbucket-home

# ! Update permissions and ownership accordingly
```

(Be sure to replace X.Y.Z in the above commands with the version number of Bitbucket Server.)

3. Create the [startup script](#) in /etc/init.d/bitbucket with the following contents (Ensure the script is executable by running `chmod 755 bitbucket`):

```
#!/bin/sh

#### BEGIN INIT INFO
# Provides:          bitbucket
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Initscript for Atlassian Bitbucket Server
# Description:       Automatically start Atlassian Bitbucket Server when
the system starts up.
#                   Provide commands for manually starting and stopping
Bitbucket Server.
#### END INIT INFO

# Adapt the following lines to your configuration
# RUNUSER: The user to run Bitbucket Server as.
RUNUSER=vagrant

# BITBUCKET_INSTALLDIR: The path to the Bitbucket Server
installation directory
BITBUCKET_INSTALLDIR="/opt/atlassian-bitbucket-X.Y.Z"

# BITBUCKET_HOME: Path to the Bitbucket home directory
BITBUCKET_HOME="/opt/bitbucket-home"

#
=====
=====
#
=====
=====
#
=====
=====

# PATH should only include /usr/* if it runs after the mountnfs.sh
script
```



```

PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="Atlassian Bitbucket Server"
NAME=bitbucket
PIDFILE=$BITBUCKET_INSTALLDIR/work/catalina.pid
SCRIPTNAME=/etc/init.d/$NAME

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.0-6) to ensure that this file is
present.
. /lib/lsb/init-functions

run_with_home() {
    if [ "$RUNUSER" != "$USER" ]; then
        su - "$RUNUSER" -c "export
BITBUCKET_HOME=${BITBUCKET_HOME};${BITBUCKET_INSTALLDIR}/bin/$1"
    else
        export
BITBUCKET_HOME=${BITBUCKET_HOME};${BITBUCKET_INSTALLDIR}/bin/$1
    fi
}

#
# Function that starts the daemon/service
#
do_start()
{
    run_with_home start-bitbucket.sh
}

#
# Function that stops the daemon/service
#
do_stop()
{
    if [ -e $PIDFILE ]; then
        run_with_home stop-bitbucket.sh
    else
        log_failure_msg "$NAME is not running."
    fi
}

case "$1" in
    start)
        [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
        do_start
        case "$?" in
            0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
            2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
    stop)
        [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"

```

```

do_stop
case "$?" in
    0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
    2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
esac
;;
status)
    if [ ! -e $PIDFILE ]; then
        log_failure_msg "$NAME is not running."
        return 1
    fi
    status_of_proc -p $PIDFILE "" $NAME && exit 0 || exit $?
    ;;
restart|force-reload)
    #
    # If the "reload" option is implemented then remove the
    # 'force-reload' alias
    #
    log_daemon_msg "Restarting $DESC" "$NAME"
    do_stop
    case "$?" in
        0|1)
            do_start
            case "$?" in
                0) log_end_msg 0 ;;
                1) log_end_msg 1 ;; # Old process is still running
                *) log_end_msg 1 ;; # Failed to start
            esac
            ;;
        *)
            # Failed to stop
            log_end_msg 1
            ;;
    esac
    ;;
*)
    echo "Usage: $SCRIPTNAME
{start|stop|status|restart|force-reload}" >&2

```

```
        exit 3
    ;;
esac
```

Running on system boot

1. To start on system boot, add the script to the start up process.
For Ubuntu (and other Debian derivatives) use:

```
update-rc.d bitbucket defaults
```

For RHEL (and derivatives) use:

```
chkconfig --add bitbucket --level 0356
```

Note: You may have to install the `redhat-lsb` package on RHEL (or derivatives) to provide the LSB functions used in the script.

2. Verify that the Bitbucket Server service comes back up after restarting the machine.

Using a systemd unit file

Thanks to [Patrick Nelson](#) for calling out this approach, which he set up for a Fedora system. It also works on other distributions that use systemd as the init system. This approach does not restart Bitbucket Server if it stops by itself.

1. Create a `bitbucket.service` file in your `/etc/systemd/system/` directory with the following lines:

```
[Unit]
Description=Atlassian Bitbucket Server Service
After=syslog.target network.target

[Service]
Type=forking
User=atlbitbucket
ExecStart=/opt/atlassian-bitbucket-X.Y.Z/bin/start-bitbucket.sh
ExecStop=/opt/atlassian-bitbucket-X.Y.Z/bin/stop-bitbucket.sh

[Install]
WantedBy=multi-user.target
```

The value for `User` should be adjusted to match the user that Bitbucket Server runs as. `ExecStart` and `ExecStop` should be adjusted to match the path to your <Bitbucket Server installation directory>.

2. Enable the service to start at boot time by running the following in a terminal:

```
systemctl enable bitbucket.service
```

3. [Stop Bitbucket Server](#), then restart the system, to check that Bitbucket Server starts as expected.
4. Use the following commands to manage the service:

Disable the service:

```
systemctl disable bitbucket.service
```

Check that the service is set to start at boot time:

```
if [ -f /etc/systemd/system/*.wants/bitbucket.service ]; then echo  
"On"; else echo "Off"; fi
```

Manually start and stop the service:

```
systemctl start bitbucket  
systemctl stop bitbucket
```

Check the status of Bitbucket Server:

```
systemctl status bitbucket
```

Running Bitbucket Server as a Windows service

This page only applies...
... if you are manually
installing or upgrading
Bitbucket Server [from an
archive file](#) .

If you're using the installer...
... you should read the
Bitbucket Server [Getting
started](#) page instead.

Related pages...

- [Running Bitbucket Server as a Linux service](#)
- [Using Bitbucket Server in the enterprise.](#)

We recommend that you use the Bitbucket Server installer to install Bitbucket Server as a service on Windows. It installs Bitbucket Server as a service and creates items in the Windows 'Start' menu for starting and stopping Bitbucket Server – see [Getting started](#).

The information on this page only applies if you are manually installing or upgrading Bitbucket Server from an archive file. See [Install Bitbucket Server from an archive file](#) .

For long-term use on a Windows server, Bitbucket Server should be configured to run as a Windows service. This has the following advantages:

- Bitbucket Server will be automatically restarted when the operating system restarts.
- Bitbucket Server is less likely to be accidentally shut down, as can happen if the console window Bitbucket Server was manually started in is closed.
- Bitbucket Server logs are properly managed by the Windows service.

System administration tasks are [not supported by Atlassian](#). These instructions are only provided as a guide.

Prerequisites

- If you are using a 64-bit version of Windows, first ensure that Bitbucket Server uses a 64-bit JVM (check by running `java -version` in a Command Prompt, and ensure the `JAVA_HOME` system environment variable points to the 64-bit JVM), and then replace the 32-bit Tomcat binaries with their 64-bit counterparts in the `<Bitbucket Server installation directory>/bin` directory:

```
cd <BITBUCKET-INST/bin>
rename tomcat8.exe tomcat8.exe.x86
rename tcnative-1.dll tcnative-1.dll.x86
rename tomcat8.exe.x64 tomcat8.exe
rename tcnative-1.dll.x64 tcnative-1.dll
```

- On any Windows operating system with User Account Control (UAC) such as Windows Vista or Windows 7, simply logging in to Windows with an administrator account will not be sufficient to execute the script in the procedure below. You must either disable UAC or run `cmd.exe` as an administrator (e.g. by right-clicking on **cmd.exe** then choose **Run as administrator**).
- Ensure the `JAVA_HOME` variable is set to the root of your Java platform's installation directory.

Your `JAVA_HOME` cannot contain spaces, so the default Java installation directory of `C:\Program Files\Java` won't work.

- Bitbucket Server should be run from a local [dedicated user account](#) that does not have admin privileges and that has read, write and execute access to the Bitbucket home directory and the `<Bitbucket Server installation directory>`. See [Git push operations extremely slow on Windows](#).
- When you run Bitbucket Server as a Windows service, all settings in `setenv.bat` are ignored. Ensure that you have set `BITBUCKET_HOME` as a *system* environment variable, before running the `service.bat` script.
- If you upgraded Bitbucket Server from version 1.x to 2.x and Bitbucket Server stopped running as a service you will need to reinstall the service according to instructions in the [Bitbucket Server upgrade guide](#).

Set up Bitbucket Server as a Windows service

The information in this section only applies if you are manually installing Bitbucket Server as a Windows service. Alternatively, you can use the Bitbucket Server installer for Windows to install Bitbucket Server as a service – see [Running the Bitbucket Server installer](#).

As of version 4.6, installing Bitbucket Server as a Windows service requires installing two services: the Bitbucket Server service, which provides application functionality, and the bundled Bitbucket Elasticsearch service, which provides content indexing and search functionality.

To install Bitbucket Server as a Windows service

- [Stop Bitbucket Server](#).
- Create a system environment variable with `BITBUCKET_HOME` as the **Variable name** and the absolute path to your Bitbucket Server [home directory](#) as the **Variable value**. Don't use a trailing backslash. Note that the Bitbucket home directory *should not* be located inside the `<Bitbucket Server installation directory>`. You must do this step *before* running the `service.bat` script in Step 5 below.
- Create these system environment variables required for the bundled Bitbucket Elasticsearch service:

Variable name	Variable value
LOG_DIR	%BITBUCKET_HOME%\log\search
DATA_DIR	%BITBUCKET_HOME%\shared\search\data

CONF_DIR	%BITBUCKET_HOME%\shared\search

4. Open a Command Prompt (as an administrator – see the [Prerequisites](#) section above).
5. Change directory to <Bitbucket Server installation directory>\bin. If a directory in the path has spaces (e.g. C:\Program Files\.), use its eight-character equivalent (e.g. C:\Progra~1\.).
6. Create a service with the name "BITBUCKET" and a display name of "Atlassian Bitbucket Server BITBUCKET" by running this command:

```
service.bat install
```

If you would like to customize the name you can instead run:

```
service.bat install MyName
```

This creates a service called "MyName" with a display name of "Atlassian Bitbucket Server MyName".

7. Increase the amount of memory that Bitbucket Server can use (the default is 768 Mb) by running this command:

```
tomcat8 //US//service_name --JvMx 1024
```

8. Change directory to <Bitbucket Server installation directory>\elasticsearch.
9. If you are installing a new instance of Bitbucket Server, or are upgrading from Bitbucket Server 4.5 or earlier, copy the template Elasticsearch config files over to your Bitbucket home directory by running the following command:

```
robocopy /S config-template %BITBUCKET_HOME%\shared
```

10. Change directory to <Bitbucket Server installation directory>\elasticsearch\bin.
11. Create a service with the name "BITBUCKET_SEARCH" and a display name of "Atlassian Bitbucket Elasticsearch (BITBUCKET_SEARCH)" by running this command:

```
service.bat install
```

If you would like to customise the name you can instead run:

```
service.bat install MyName
```

This creates the service "MyName" with a display name of "Atlassian Bitbucket Elasticsearch (MyName)".

12. Verify the Bitbucket Server and Bitbucket Elasticsearch services start after restarting the machine.

Here is an example of the output you can expect after successfully completed this procedure

```
C:\Atlassian\atlassian-bitbucket-4.6.3\bin>service.bat install
Installing the service 'BITBUCKET' ...
Using CATALINA_HOME:      "C:\Atlassian\atlassian-bitbucket-4.6.3"
Using CATALINA_BASE:      "C:\Atlassian\atlassian-bitbucket-4.6.3"
Using JAVA_HOME:          "c:\java\jdk1.8.0_92"
Using JRE_HOME:           "c:\java\jdk1.8.0_92\jre"
Using JVM:                 "c:\java\jdk1.8.0_92\jre\bin\server\jvm.dll"
The service 'BITBUCKET' has been installed.
```

```
C:\Atlassian\atlassian-bitbucket-4.6.3\bin>cd ..\elasticsearch\bin
C:\Atlassian\atlassian-bitbucket-4.6.3\elasticsearch\bin>service.bat
install
Installing service        : "BITBUCKET_SEARCH"
Using JAVA_HOME (64-bit): "c:\java\jdk1.8.0_92"
The service 'BITBUCKET_SEARCH' has been installed.
```

```
C:\Atlassian\atlassian-bitbucket-4.6.3\elasticsearch\bin>net start
BITBUCKET_SEARCH
The Atlassian Bitbucket Elasticsearch 2.3.1 (BITBUCKET_SEARCH)
service is starting.
The Atlassian Bitbucket Elasticsearch 2.3.1 (BITBUCKET_SEARCH)
service was started successfully.
```

```
C:\Atlassian\atlassian-bitbucket-4.6.3\elasticsearch\bin>net start
BITBUCKET
The Atlassian Bitbucket BITBUCKET service is starting.
The Atlassian Bitbucket BITBUCKET service was started successfully.
```

Troubleshooting

- If your service fails to start with code 4, ensure you ran `service.bat install` in a Command Prompt running as an Administrator.

Bitbucket Server config properties

This page describes the Bitbucket Server system properties that can be used to control aspects of the behavior in Bitbucket Server. Create the `bitbucket.properties` file, in the shared folder of your [Bitbucket Server home directory](#), and add the system properties you need, use the standard format for Java properties files.

Note that the `bitbucket.properties` file is created automatically when you perform a [database migration](#).

Bitbucket Server must be restarted for changes to become effective.

Default values for system properties, where applicable, are specified in the tables below.

On this page:

• Audit	• Executor	• Ref metadata
• Authentication	• Features	• Ref restrictions
• Avatars	• Hibernate	• Resource throttling
• Backup	• JIRA Applications	• SCM – Cache
• Changesets	• JMX	• SCM – Git
• Changeset indexing	• Liquibase	• Server busy banners
• Commit graph cache	• Logging	• Setup automation
• Database	• Notifications	• SMTP
• Database pool	• Paging	• SSH command execution
• Display	• Password reset	• SSH security

- [Downloads](#)
- [Elasticsearch](#)
- [Events](#)
- [Process execution](#)
- [Pull requests](#)
- [Readme parsing](#)
- [Syntax highlighting](#)
- [Webhooks](#)

Audit

Property	Description
<code>audit.highest.priority.to.log=HIGH</code>	Defines the lowest priority audit events that will be logged. Accepted values are: HIGH, MEDIUM, LOW and NONE. Setting the value to HIGH will result in only HIGH level events being logged. NONE will cause no events to be logged. MEDIUM will only allow events with a priority of MEDIUM and HIGH to be logged. Refer to the levels for the various events. This does not affect events displayed in the Audit log screens for projects and repositories.
<code>audit.details.max.length=1024</code>	Defines the number of characters that can be stored as details for a single audit entry.
<code>plugin.bitbucket-audit.max.entity.rows=500</code>	The maximum number of entries a project or repository can have in the audit tables. This does not affect the data stored in the logs.
<code>plugin.bitbucket-audit.cleanup.batch.size=1000</code>	When trimming the audit entries table this is the maximum number of rows that will be trimmed in one transaction. Reduce this size if you are having issues with long running transactions. This does not affect the data stored in the logs.
<code>plugin.bitbucket-audit.cleanup.run.interval=24</code>	How often the audit tables will be checked to see if they need to be trimmed (in hours). This does not affect the data stored in the logs.

Authentication

See also [Connecting Bitbucket Server to Crowd](#).

Property	Descrip

<code>plugin.auth-crowd.sso.enabled=false</code>	Whether support is enabled. Regardless of the setting, SSO authentication can only be used when a Crowd directory configuration in Bitbucket that is configured for SSO.
<code>plugin.auth-crowd.sso.session.lastvalidation=atl.crowd.sso.lastvalidation</code>	The session to use when storing a value of last authentication.
<code>plugin.auth-crowd.sso.session.tokenkey=atl.crowd.sso.tokenkey</code>	The session to use when storing a value of authentication token.
<code>plugin.auth-crowd.sso.session.validationinterval=3</code>	The number of minutes authentication validation session. If the value is 0, the SSO session is validated every Crowd server request.
<code>plugin.auth-crowd.sso.http.max.connections=20</code>	The maximum number of connections for communication with the server.
<code>plugin.auth-crowd.sso.http.proxy.host</code>	The name of the proxy server to connect to Crowd server.
<code>plugin.auth-crowd.sso.http.proxy.port</code>	The port of the proxy server (if specified) or the host is the

<code>plugin.auth-crowd.sso.http.proxy.username</code>	The user used to authenticate the proxy; the proxy requires authentication.
<code>plugin.auth-crowd.sso.http.proxy.password</code>	The password used to authenticate the proxy; the proxy requires authentication.
<code>plugin.auth-crowd.sso.http.timeout=5000</code>	The HTTP connection timeout in milliseconds for communication with the server. A zero indicates there is no connection timeout.
<code>plugin.auth-crowd.sso.socket.timeout=20000</code>	The socket timeout in milliseconds may wisely override default values if latency to the Crowd is high.

<code>auth.remember-me.enabled=optional</code>	Controls remember-me authentication. If disabled, users must perform a login checkbo checked login for 'Remember login' checkbox only displayed when selected. Possible values are: • <code>always</code>: always check remember-me cookie and generate success login • <code>optional</code>: check display login Remember cookie generate when checked • <code>never</code>: Remember authentication is disabled completely
<code>auth.remember-me.token.expiry=30</code>	How long remember-me tokens are valid. Note that the remember-me token is invalidated when a new remember-me token is generated. Value is

Avatars

Property	Description

<code>avatar.gravatar.default=mm</code>	The fallback URL for Gravatar avatar when a user does not have an acceptable avatar configured. This may be a URL resource, or a Gravatar provided default set. This configuration setting is DEPRECATED. It will be removed in Bitbucket Server 3.0. Use <code>avatar.url.default</code> instead.
<code>avatar.max.dimension=1024</code>	Controls the max height <i>and</i> width for avatar image. Even if the avatar is within the acceptable file size, if its dimensions exceed this value for height <i>or</i> width, it will be rejected. When an avatar is loaded by the server for processing, images with large dimensions may expand from as small as a few kilobytes on disk to consume a substantially larger amount of memory depending on how well the image data was compressed. Increasing this limit <i>substantially</i> increase the amount of memory used while processing avatars and may result in OutOfMemoryErrors. Value is in PIXELS.
<code>avatar.max.size=1048576</code>	Controls how large an avatar is allowed to be. Avatars larger than this are rejected and cannot be uploaded to the server, to prevent excessive disk usage. Value is in BYTES.
<code>avatar.temporary.cleanup.interval=1800000</code>	Controls how frequently temporary avatars are cleaned up. Any temporary avatars that have been uploaded are checked against their configured maximum age and removed from the file system if they are "too old". Value is in MILLISECONDS.
<code>avatar.temporary.max.age=30</code>	Controls how long a temporary avatar that has been uploaded is retained before it is automatically deleted. Value is in MINUTES.

<code>avatar.url.default=\${avatar.gravatar.default}</code>	Defines the fallback URL to be formatted into the <code>avatar.url.format.http</code> or <code>avatar.url.format.https</code> URL format for use when a user does not have an acceptable avatar configured. This value may be a URL or, if using Gravatar, it may be the identifier for one of Gravatar's default avatars. The default here falls back on the now-deprecated <code>avatar.gravatar.fallback</code> setting, which should ensure that the value, if set, continues to work until it is removed in Bitbucket Server 3.0. At that time, this default will become "mm".
<code>avatar.url.format.http=http://www.gravatar.com/avatar/%1\$s.jpg?s=%2\$d&d=%3\$s</code>	Defines the default URL format for retrieving user avatars over HTTP. The default uses any G-rated avatar provided by the Gravatar service [http://www.gravatar.com] The following format parameters are available: <code>%1\$s</code> – the user's e-mail address, MD5 hashed, or "00000000000000000000000000000000" if the user has no e-mail. <code>%2\$d</code> – the requested avatar size. <code>%3\$s</code> – the fallback URL, URL-encoded which may be defined using "avatar.url.default". <code>%4\$s</code> – the user's e-mail address, not hashed, or an empty string if the user has no e-mail.
<code>avatar.url.format.https=https://secure.gravatar.com/avatar/%1\$s.jpg?s=%2\$d&d=%3\$s</code>	Defines the default URL format for retrieving user avatars over HTTPS. The default uses any G-rated avatar provided by the Gravatar service [http://www.gravatar.com] The following format parameters are available: <code>%1\$s</code> – the user's e-mail address, MD5 hashed, or "00000000000000000000000000000000" if the user has no e-mail. <code>%2\$d</code> – the requested avatar size. <code>%3\$s</code> – the fallback URL, URL-encoded which may be defined using "avatar.url.default". <code>%4\$s</code> – the user's e-mail address, not hashed, or an empty string if the user has no e-mail.

Backup

Property	Description

<code>backup.drain.database.timeout=60</code>	Defines the number of seconds Bitbucket Server will wait for connections to the database to drain and latch in preparation for a backup. Value is in SECONDS.

Changesets

Property	Description
<code>changeset.diff.context=10</code>	Defines the number of context lines to include around diff segments in changeset diffs.

Changeset indexing

These properties control how changesets are indexed when new commits are pushed to Bitbucket Server.

Property	Description
<code>indexing.max.threads=2</code>	Controls the maximum number of threads which are used to perform indexing. The resource limits configured below are not applied to these threads, so using a high number may negatively impact server performance.
<code>indexing.job.batch.size=250</code>	Defines the number of changesets which will be indexed in a single database transaction.
<code>indexing.job.queue.size=150</code>	Defines the maximum number of pending indexing requests. When this limit is reached, attempts to queue another indexing operation will be rejected.
<code>indexing.process.timeout.execution=3600</code>	Controls how long indexing processes are allowed to execute before they are interrupted, even if they are producing output or consuming input. Value is in SECONDS.

Commit graph cache

Property	Description
<code>commit.graph.cache.min.free.space=1073741824</code>	Controls how much space needs to be available on disk (specifically under <code><Bitbucket home directory>/caches</code>) for caching to be enabled. This setting ensures that the cache plugin does not fill up the disk. Value is in BYTES.
<code>commit.graph.cache.max.threads=2</code>	Defines the number of threads that will be used to create commit graph cache entries.

<code>commit.graph.cache.max.job.queue=1000</code>	Defines the maximum number of pending cache creation jobs.

Database

Database properties allow very specific configuration for your database connection parameters, which are set by Bitbucket Server during database setup and migration, and allow you to configure a database of your own. We don't expect that you will edit these, except in collaboration with Atlassian Support.

If none of the properties below are specified in `bitbucket.properties`, then the internal HSQL database will be used.

If the `jdbc.driver`, `jdbc.url`, `jdbc.password` and `jdbc.user` properties are specified in `bitbucket.properties` when the Setup Wizard runs after installing Bitbucket Server, then those values will be used, and the Setup Wizard will not display the database configuration screen.

Any other driver must be placed in `WEB-INF/lib` in order to use the associated database.

⚠ Warning: `jdbc.driver` and `jdbc.url` are available to plugins via the `ApplicationPropertiesService`. Some JDBC drivers allow the username and password to be defined in the URL. Because that property is available throughout the system (and will be included in STP support requests), that approach should not be used. The `jdbc.username` and `jdbc.password` properties should be used for these values instead.

Property	Description
<code>jdbc.driver=org.hsqldb.jdbcDriver</code>	The JDBC driver class that to connect to the database The internal Bitbucket Server <code>org.hsqldb.jdbcDriver</code> is located in the Server home directory. Bitbucket Server bundles the following JDBC drivers: <code>org.postgresql.Driver</code> <code>com.microsoft.sqljdbc4</code> (more info) <code>oracle.jdbc.driver</code> The JDBC drivers for MySQL and MariaDB are not bundled with Bitbucket Server (due to licensing requirements). You must download and install the drivers yourself. See Bitbucket Server to MySQL for more information.
<code>jdbc.url=jdbc:hsqldb:\${bitbucket.home}/data/db;shutdown=true</code>	This is the JDBC url that connects to the database. This should be in the format <code>jdbc:hsqldb:hostname:port</code> , the hostname is the host you are connecting to. Please refer to the documentation of the database you are connecting to for more information.
<code>jdbc.user=bitbucket</code>	This is the user that Bitbucket Server uses to connect to the database with. The user must have the necessary permissions to create tables and indexes, as well as to manage the entire database schema.
<code>jdbc.password=bitbucket</code>	The password that the user must use to connect with.

<code>jdbc.ignoreunsupported=false</code>	Allows using a given data type that is <code>UNSUPPORTED</code> . This is not documented, nor to be used as a mechanism to override the default behavior in the event that it incorrectly blocks.

Database pool

These properties control the database pool. The pool implementation used is HikariCP. Documentation for these settings can be found at: <https://github.com/brettwooldridge/HikariCP/wiki/Configuration>

To get a feel for how these settings really work in practice, the most relevant classes in HikariCP are:

- `com.zaxxer.hikari.HikariConfig` Holds the configuration for the database pool and has documentation for the available settings.
- `com.zaxxer.hikari.pool.HikariPool` Provides the database pool and manages connections.
- `com.zaxxer.hikari.util.ConnectionBag` Holds references to open connections, whether in-use or idle.

Property	Description
<code>db.pool.size.idle=0</code>	Defines the number of connections the pool tries to keep idle. <i>The system can have more idle connections than the value configured here.</i> As connections are borrowed from the pool, this value is used to control whether the pool will eagerly open new connections to try and keep some number idle, which can help smooth ramp-up for load spikes. By default, the system does not eagerly open new idle connections. Connections will be opened as needed. Once opened, connections may <i>become</i> idle and will be retained for <code>db.pool.timeout.idle</code> seconds.
<code>db.pool.size.max=80</code>	Defines the maximum number of connections the pool can have open at once.
<code>db.pool.timeout.connect=15</code>	Defines the amount of time the system will wait when attempting to open a new connection before throwing an exception. The system may hang, during startup, for the configured number of seconds if the database is unavailable. As a result, the timeout configured here should not be generous. This value is in SECONDS.
<code>db.pool.timeout.idle=1750</code>	Defines the maximum period of time a connection may be idle before it is closed. In general, generous values should be used here to prevent creating and destroying many short-lived database connections (which defeats the purpose of pooling). Note: If an aggressive timeout is configured on the database server, a <i>more aggressive</i> timeout must be used here to avoid issues caused by the database server closing connections from its end. The value applied here should ensure the system closes idle connections before the database server does. This value needs to be less than <code>db.pool.timeout.lifetime</code> otherwise the idle timeout will be ignored. This value is in SECONDS.

<code>db.pool.timeout.leak=0</code>	Defines the maximum period of time a connection may be checked out before it is reported as a potential leak. <i>By default, leak detection is not enabled.</i> Long-running tasks, such as taking a backup or migrating databases, can easily exceed this threshold and trigger a false positive detection. This value is in MINUTES.
<code>db.pool.timeout.lifetime=30</code>	Defines the maximum <i>lifetime</i> for a connection. Connections which exceed this threshold are closed the first time they become idle and fresh connections are opened. This value is in MINUTES.

Display

Property	Description
<code>display.max.source.lines=20000</code>	Controls how many lines of a source file will be retrieved before a warning banner is shown that encourages the user is to download the raw file for further inspection. This property relates to <code>page.max.source.lines</code> (see Paging below) in that up to $(\text{display.max.source.lines} / \text{page.max.source.lines})$ requests will be made to view the page.

Downloads

Property	Description
<code>http.download.raw.policy=Smart</code>	Controls the download policy for raw content. Possible values are: Insecure – allows all file types to be viewed in the browser. Secure – requires all file types to be downloaded rather than viewed in the browser. Smart – forces "dangerous" file types to be downloaded, rather than allowing them to be viewed in the browser. These options are case-sensitive and defined in <code>com.atlassian.http.mime.DownloadPolicy</code> .

Elasticsearch

Bitbucket Server 4.5+ ships with an embedded instance of Elasticsearch. These properties enable admins to configure the base URL of the Elasticsearch instance, and enable basic security measures in the form of a username and password for accessing the Elasticsearch instance.

If an Elasticsearch parameter is set in the properties file, *it cannot be edited later from the admin UI*. Any changes that need to be made to the Elasticsearch configuration must be made within the `bitbucket.properties` file.

Property	Description
<code>plugin.search.elasticsearch.baseurl</code>	Sets the base URL of an Elasticsearch instance.

<code>plugin.search.elasticsearch.username</code>	Username for connecting to an Elasticsearch instance.
<code>plugin.search.elasticsearch.password</code>	Password for connecting to an Elasticsearch instance.

Events

These properties control the number of threads that are used for dispatching asynchronous events. Setting this number too high can decrease overall throughput when the system is under high load because of the additional overhead of context switching. Configuring too few threads for event dispatching can lead to events being queued up, thereby reducing throughput. These defaults scale the number of dispatcher threads with the number of available CPU cores.

Property	Description
<code>event.dispatcher.core.threads=0.8* \${scaling.concurrency}</code>	The minimum number of threads that is available to the event dispatcher. The <code>\${scaling.concurrency}</code> variable is resolved to the number of CPUs that are available.
<code>event.dispatcher.max.threads=\${sc aling.concurrency}</code>	The maximum number of event dispatcher threads. The number of dispatcher threads will only be increased when the event queue is full and <code>max.threads</code> has not been reached yet.
<code>event.dispatcher.queue.size=4096</code>	The number of events that can be queued. When the queue is full and no more threads can be created to handle the events, events will be discarded.
<code>event.dispatcher.keepAlive=60</code>	The time a dispatcher thread will be kept alive when the queue is empty and more than <code>core.threads</code> threads are running. Value is in SECONDS.

Executor

Controls the thread pool that is made available to plugins for asynchronous processing.

Property	Description
<code>executor.max.threads=\${scaling.concurrency}</code>	Specifies the maximum number of threads in the thread pool. When more threads are required than the configured maximum, the thread attempting to schedule an asynchronous task to be executed will block until a thread in the pool becomes available. The <code>\${scaling.concurrency}</code> variable is resolved to the number of CPUs that are available.

Features

Feature properties control high-level system features, allowing them to be disabled for the entire instance. Features that are disabled at this level are disabled *completely*. This means that instance-level configuration for a feature is overridden. It also means that a user's permissions are irrelevant; a feature is still disabled

even if the user has the SYS_ADMIN permission.

Property	Description
<code>attachment.upload.max.size=10</code>	Controls the file size limit for individual attachments to pull request comments and descriptions. Value is in MB.
<code>feature.attachments=true</code>	Controls whether attachments can be added to pull request comments and descriptions.
<code>feature.auth.captcha=true</code>	Controls whether to require CAPTCHA verification when the number of failed logins is exceeded. If enabled, any client who has exceeded the number of failed logins allowed using either the Bitbucket Server web interface or the Git hosting interface will be required to authenticate in the Bitbucket Server web interface and successfully submit a CAPTCHA before continuing. Setting this to <code>false</code> will remove this restriction and allow users to incorrectly authenticate as many times as they like without penalty.  Warning: It is STRONGLY recommended you keep this setting enabled. Disabling it will have the following ramifications: - Your users may lock themselves out of any underlying user directory service (LDAP, Active Directory etc) because Bitbucket Server will pass through all authentication requests (regardless of the number of previous failures) to the underlying directory service. - For Bitbucket Server installations where you use Bitbucket Server for user management or where you use a directory service with no limit on the number of failed logins before locking out users, you will open Bitbucket Server or the directory service up to brute-force password attacks.
<code>feature.forks=true</code>	Controls whether repositories can be forked. This setting <i>supercedes and overrides</i> instance-level configuration. If this is set to <code>false</code> , even repositories which are marked as forkable cannot be forked.
<code>feature.personal.repos=true</code>	Controls whether personal repositories can be created. When set to <code>false</code> , personal repository creation is disabled globally in Bitbucket Server.
<code>feature.public.access=true</code>	Public access to Bitbucket Server allows unauthenticated users to be granted access to projects and repositories for specific read operations including cloning and browsing repositories. This is normally controlled by project and repository administrators but can be switched off system wide by setting this property to <code>false</code> . This can be useful in highly sensitive environments.

Hibernate

Property	Description
<code>hibernate.format_sql=false</code>	When <code>hibernate.show_sql</code> is enabled, this flag controls whether Hibernate will format the output SQL to make it easier to read.

<code>hibernate.jdbc.batch_size=20</code>	Controls Hibernate's JDBC batching limit, which is used to make bulk processing more efficient (both for processing and for memory usage).
<code>hibernate.show_sql=false</code>	Used to enable Hibernate SQL logging, which may be useful in debugging database issues. This value should generally only be set by developers.

JIRA Applications

Property	Description
<code>plugin.jira-integration.pullrequest.attribute.changesets.max=100</code>	Controls the maximum number of changesets to retrieve when retrieving attributes associated with changesets of a pull-request. This value should be between 50 and 1000 as Bitbucket Server will enforce an lower bound of 50 issues and an upper bound of 1000 issues.
<code>plugin.jira-integration.remote.page.max.issues=20</code>	Controls the maximum number of issues to request from a JIRA application. This value should be between 5 and 50 as Bitbucket Server will enforce a lower bound of 5 issues and an upper bound of 50 issues.

<code>plugin.jira-integration.remote.timeout.connection=5000</code>	The connection timeout duration in milliseconds for requests to JIRA applications. This timeout occurs if a JIRA application server does not answer. e.g. the server has been shut down. This value should be between 2000 and 60000 as Bitbucket Server will enforce a lower bound of 2000ms and an upper bound of 60000ms. Value is in MILLISECONDS.
<code>plugin.jira-integration.remote.timeout.socket=10000</code>	The socket timeout duration in milliseconds for requests to JIRA applications. This timeout occurs if the connection to a JIRA application has been stalled or broken. This value should be between 2000 and 60000 as Bitbucket Server will enforce a lower bound of 2000ms and an upper bound of 60000ms. Value is in MILLISECONDS.

JMX

Property	Description
<code>jmx.enabled=true</code>	Controls the publishing of Bitbucket Server specific statistics via JMX. See Enabling JMX counters for performance monitoring .

Liquibase

Property	Description
<code>liquibase.commit.block.size=10000</code>	The maximum number of changes executed against a particular Liquibase database before a commit operation is performed. Very large values may cause DBMS to use excessive amounts of memory when operating within transaction boundaries. If the value of this property is less than one, then changes will not be committed until the end of the change set.

Logging

Logging levels for any number of loggers can be set in the `bitbucket.properties` file using the following format:

```
logging.logger.<name>=<level>
```

For example, to configure all classes in the `com.atlassian.bitbucket` package to DEBUG level:

```
logging.logger.com.atlassian.bitbucket=DEBUG
```

To adjust the ROOT logger, you use the special name ROOT (case-sensitive):

```
logging.logger.ROOT=INFO
```

Notifications

Property	Description
<code>plugin.bitbucket-notification.batch.min.wait.minutes=10</code>	Controls the minimum time to wait for new notifications before sending the batch. This is the inactivity timeout. Value is in MINUTES.
<code>plugin.bitbucket-notification.batch.max.wait.minutes=30</code>	Controls the maximum time to wait for new notifications before sending the batch. This is the staleness timeout. Value is in MINUTES.

<code>plugin.bitbucket-notification.mail.max.comment.size=2048</code>	Controls the maximum allowed size of a single comment in characters (not bytes). Extra characters will be truncated.
<code>plugin.bitbucket-notification.mail.max.description.size=2048</code>	Controls the maximum allowed size of a single description in characters (not bytes). Extra characters will be truncated.
<code>plugin.bitbucket-notification.mentions.enabled=true</code>	Controls whether notifications for mentions are enabled.
<code>plugin.bitbucket-notification.max.mentions=200</code>	Controls the maximum number of allowed mentions in a single comment.
<code>plugin.bitbucket-notification.sendmode.default=BATCHED</code>	Controls the system default for notifications batching for users who have not set an explicit preference. Value is either BATCHED or IMMEDIATE.

Paging

These properties control the maximum number of objects which may be returned on a page, regardless of how many were actually requested by the user. For example, if a user requests `Integer.MAX_INT` branches on a page, their request will be limited to the value set for `page.max.branches`.

This is intended as a safeguard to prevent enormous requests from tying up the server for extended periods of time and then generating responses whose payload is prohibitively large. The defaults configured here represent a sane baseline, but may be overridden if necessary.

Property	Description
<code>page.max.branches=1000</code>	Maximum number of branches per page.
<code>page.max.changes=1000</code>	Maximum number of changes per page. Unlike other page limits, this is a hard limit; subsequent pages cannot be requested when the number of changes in a changeset exceeds this size.
<code>page.max.commits=100</code>	Maximum number of commits per page.
<code>page.max.diff.lines=10000</code>	Maximum number of segment lines (of any type, total) which may be returned for a single diff. Unlike other page limits, this is a hard limit; subsequent pages cannot be requested when a diff exceeds this size.

<code>page.max.directory.children=500</code>	Maximum number of directory entries which may be returned for a given directory.
<code>page.max.directory.recursive.children=100000</code>	Maximum number of file entries which may be returned for a recursive listing of a directory. A relatively high number as this is used by the file finder which needs to load the tree of files upfront.
<code>page.max.groups=1000</code>	Maximum number of groups per page.
<code>page.max.index.results=50</code>	Maximum number of changesets which may be returned from the index when querying by an indexed attribute. For example, this limits the number of changesets which may be returned when looking up commits against a JIRA application issue.
<code>page.max.projects=1000</code>	Maximum number of projects per page.
<code>page.max.repositories=1000</code>	Maximum number of repositories per page.
<code>page.max.source.length=5000</code>	Maximum length for any line returned from a given file when viewing source. This value truncates long lines. There is no mechanism for retrieving the truncated part short of downloading the entire file.
<code>page.max.source.lines=5000</code>	Maximum number of lines which may be returned from a given file when viewing source. This value breaks large files into multiple pages. See also Display above.
<code>page.max.tags=1000</code>	Maximum number of tags per page.
<code>page.max.users=1000</code>	Maximum number of users per page.
<code>page.max.pullrequests=100</code>	Maximum number of pull requests per page.
<code>page.scan.pullrequest.activity.size=500</code>	The size of the page Bitbucket Server should use when scanning activities.
<code>page.scan.pullrequest.activity.count=4</code>	The number of pages of activities Bitbucket Server should scan before giving up.

Password reset

Property	Description
<code>password.reset.validity.period=4320</code>	Controls how long a password reset token remains valid for. Default period is 72 hours. Value is in MINUTES.

Process execution

Property	Description
<code>process.timeout.execution=120</code> <code>process.timeout.idle=60</code>	Controls timeouts for external processes, such as Git and Hg. The idle timeout configures how long the command is allowed to run without producing any output. The execution timeout configures a hard upper limit on how long the command is allowed to run even if it is producing output. Values are in SECONDS. Using 0, or a negative value, disables the timeout completely. USE AT YOUR OWN RISK!

Pull requests

Property	Description
<code>plugin.bitbucket-git.pullrequest.merge.strategy.KEY.slug=no-ff</code>	Control the merge strategy for a repository (where KEY is the repository slug). Note that the URL for the repository is of the following form: <a href="http://<bitbucketdomain>/projects/<PROJECTKEY>/r/<slug>">http://<bitbucketdomain>/projects/<PROJECTKEY>/r/<slug> Overrides project and global settings. Possible values are: • <code>no-ff</code> – no fast-forward; the default setting. • <code>ff</code> – allow fast-forward; will merge when necessary. • <code>ff-only</code> – require fast-forward; will never create a merge if one is required. • <code>squash</code> – collapse all incoming commits into a single commit on the target branch; never create a merge. • <code>squash-ff-only</code> – collapse all the incoming commits into a single commit directly to the target branch, never creating a merge if the source branch is fast-forward.
<code>plugin.bitbucket-git.pullrequest.merge.strategy.KEY=no-ff</code>	Control the merge strategy for a project (where KEY is the project slug). Overrides global settings. Is overridden by repository settings. Possible values are listed above.
<code>plugin.bitbucket-git.pullrequest.merge.strategy=no-ff</code>	Control the merge strategy globally. Is overridden by repository settings. Possible values are listed above.
<code>pullrequest.diff.context=10</code>	Defines the number of context lines to include around request diffs. By default, Git only includes 3 lines. The more context you include, the more useful context around changes, until the context is implemented.
<code>pullrequest.rescope.changesets.display=5</code>	Defines the maximum number of changesets per type (added or removed) to display in a rescope activity.
<code>pullrequest.rescope.changesets.max=1000</code>	Defines the absolute maximum number of changesets to display when attempting to determine, for a given rescope activity, which changesets were added to or removed from a pull request. Adjusting this value has a significant memory footprint impact on the system. It is not recommended to be changed, but the option is provided here to support it.

<code>pullrequest.rescope.detail.threads=2</code>	Defines the maximum number of threads to use for pull request details. These threads perform the requisite processing for commits added and removed when a pull request is rescoped. Most rescopes do not add or remove any commits. Some are deleted during processing. The primary goal is to ensure that the number of merges already been calculated when users try to view a pull request.
<code>pullrequest.rescope.drift.threads=4</code>	Defines the maximum number of threads to use when performing comment drift for a pull request during rescope. Higher number of threads mean higher throughput! Performing comment drift requires a new merge to be created, which can be very I/O intensive. A large number of merges running at the same time can significantly impact the speed of performing comment drift.

Readme parsing

Property	Description
<code>plugin.bitbucket-readme.max.size=65536</code>	Controls the maximum allowed size of a readme file to parse. Value is in BYTES.

Ref metadata

Property	Description
<code>ref.metadata.timeout=2</code>	Controls timeouts for retrieving metadata associated with a collection of refs from all metadata providers collectively. This value is in SECONDS.
<code>ref.metadata.max.request.count=100</code>	Controls the maximum number of refs that can be used in a metadata query.

Ref restrictions

Property	Description
<code>plugin.bitbucket-ref-restriction.case.insensitive=true</code>	Controls whether refs are matched case insensitively for ref restrictions.
<code>plugin.ref-restriction.feature.splash=true</code>	Controls whether new users are shown a splash page when first viewing ref restrictions.
<code>plugin.bitbucket-ref-restriction.max.resources=100</code>	The maximum number of ref restrictions per repository.

```
plugin.bitbucket-ref-restriction.max.resource.entities=50
```

The maximum number of access grants per ref restriction.

Resource throttling

These properties define concurrent task limits for the ThrottleService, limiting the number of concurrent Git operations of a given type that may be run at once. This is intended to help prevent Bitbucket Server from overwhelming a server machine with running processes. Bitbucket Server has two settings to control the number of Git processes that are allowed to process in parallel: one for the web UI and one for the 'hosting' operations (pushing and pulling commits, and cloning a repository).

When the limit is reached for the given resource, the request will wait until a currently running request has completed. If no request completes within a configurable timeout, the request will be rejected.

When requests while accessing the Bitbucket Server UI are rejected, users will see either a 501 error page indicating the server is under load, or a popup indicating part of the current page failed.

When Git client 'hosting' commands (pull/push/clone) are rejected, Bitbucket Server does a number of things:

- Bitbucket Server will return an error message to the client which the user will see on the command line: "Bitbucket Server is currently under heavy load and is not able to service your request. Please wait briefly and try your request again"
- A warning message will be logged for every time a request is rejected due to the resource limits, using the following format:
"A [scm-hosting] ticket could not be acquired (0/12)"
- For five minutes after a request is rejected, Bitbucket Server will display a red banner in the UI to warn that the server is under load.

The hard, machine-level limits these are intended to prevent hitting are very OS- and hardware-dependent, so you may need to tune them for your instance of Bitbucket Server. When hyperthreading is enabled for the server CPU, for example, it is likely that the server will allow sufficient concurrent Git operations to completely bury the I/O on the machine. In such cases, we recommend starting off with a less aggressive default on multi-cored machines – the value can be increased later if hosting operations begin to back up. These defaults are finger-in-the-wind guesstimates (which so far have worked well).

Additional resource types may be configured by defining a key with the format `throttle.resource.<resource-name>`.

When adding new types, it is strongly recommended to configure their ticket counts explicitly using this approach.

Property	Description
<code>throttle.resource.scm-command=25</code>	Limits the number of operations that support the UI, such as <code>git diff</code> , <code>git blame</code> , or <code>git rev-list</code> , that can run concurrently. This is intended to prevent these SCM commands from competing with the running of push and pull operations.
<code>throttle.resource.scm-command.timeout=2</code>	Controls how long threads will wait for SCM commands to complete when the system is already running the maximum number of SCM commands. Value is in SECONDS.

<code>throttle.resource.scm-hosting=1.5*\${scaling.concurrency}</code>	Limits the number of SCM 'hosting' operations, such as <code>git clone</code> , <code>git push</code> and <code>git pull</code> over HTTP or SSH that may be running concurrently. This is intended primarily to prevent pulls, which can be very memory-intensive, from pinning a server's resources. There is limited support for mathematical expressions; <code>+</code> , <code>-</code> , <code>*</code> , <code>\</code> and <code>()</code> are supported. You can also use the <code>\${scaling.concurrency}</code> variable which is resolved to the number of CPUs that are available.
<code>throttle.resource.scm-hosting.timeout=300</code>	Controls how long threads will wait for SCM hosting operations to complete when the system is already running the maximum number of SCM hosting operations. Value is in SECONDS.
<code>throttle.resource.busy.message.timeout=5</code>	Controls how long a warning banner is displayed in the UI after a request is rejected due to excessive load. Value is in MINUTES. Using 0, or a negative value, disables displaying the banner. This is deprecated and replaced by <code>server.busy.on.ticket.rejected.within</code> . It is due to be removed in Bitbucket Server 3.0.

SCM – Cache

See [Scaling Bitbucket Server for Continuous Integration performance](#) for more information about using the SCM Cache Plugin for Bitbucket Server.

Property	Description
<code>plugin.bitbucket-scm-cache.expiry.check.interval=300</code>	Controls how frequently expired caches are checked and deleted from disk. Value is in SECONDS.
<code>plugin.bitbucket-scm-cache.minimum.free.space=1073741824</code>	Controls how much space needs to be available on disk (specifically under <code><Bitbucket home directory>/caches</code>) for caching to be enabled. This setting ensures that the cache plugin does not fill up the disk. Value is in BYTES.
<code>plugin.bitbucket-scm-cache.protocols=HTTP,SSH</code>	Controls which protocols caching is applied to. The HTTP value encapsulates both <code>http</code> and <code>https</code> .
<code>plugin.bitbucket-scm-cache.refs.enabled=false</code>	Controls whether ref advertisement operations are cached.

<code>plugin.bitbucket-scm-cache.refs.ttl=60</code>	Controls how long the caches for ref advertisements are kept around when there no changes to the repository. Caches are automatically invalidated when someone pushes to a repository or when a pull request is merged. Time is in SECONDS.
<code>plugin.bitbucket-scm-cache.upload-pack.enabled=true</code>	Controls whether clone operations are cached.
<code>plugin.bitbucket-scm-cache.upload-pack.ttl=14400</code>	Controls how long the caches for clone operations are kept around when there no changes to the repository. Caches are automatically invalidated when someone pushes to a repository or when a pull request is merged. Time is in SECONDS.

SCM – Git

Property	Description
<code>plugin.bitbucket-git.path.executable=git</code>	Defines the default path to the Windows machines, the .exe configured value automatical general, "git" should be an absolute path, here, assuming the runtime user's PATH. With the new path searching DefaultGitBinaryHelper, setting this is unnecessary, as the plugin will find the path. This is left here purely for documentation.
<code>plugin.bitbucket-git.path.libexec=</code>	Defines the path to the Git libexec (the git-core directory). This path is the executable and is used for the git-http-backend. If this value is not set, the process will directly fork out those processes, which is unnecessary (git -> git-http-backend) to improve scalability.
<code>plugin.bitbucket-git.backend.http.buffer.size=32768</code>	Defines the buffer size in bytes for marshaling data between the socket.

<code>plugin.bitbucket-git.backend.ssh.buffer.size=32768</code>	Defines the buffer size in bytes for marshaling data between the socket.
<code>plugin.bitbucket-git.backend.timeout.idle=1800</code>	Defines the idle timeout for the plugin. It is a limit to how long the operation can execute without either producing output or receiving input. The default value is 30 seconds. This value is in SECONDS.
<code>plugin.bitbucket-git.backend.timeout.execution=86400</code>	Defines the execution timeout for the plugin. It is a hard limit to how long the operation can run even if it is producing output. The default value is 1 day. This value is in SECONDS.
<code>plugin.bitbucket-git.diff.renames=copies</code>	Defines whether copy and/or rename are performed. By default, both are performed. Only files modified in the origin are considered as rename or copy to avoid overhead. The possible settings are: • <code>copy</code> or <code>copies</code> – apply copy • <code>rename</code> or <code>renames</code> – apply rename • <code>off</code> – disables rename and copy When using <code>copy</code> or <code>copies</code>, the setting can be suffixed with a "+" to use <code>--full-index</code>. This setting should be used with caution as it is expensive. It considers every file in the origin even files not modified in the origin for copies. When copy and/or rename disabled, the <code>plugin.bitbucket-git.diff.renames.threshold</code> is used to control the similarity index required for a copy or rename.
<code>plugin.bitbucket-git.diff.renames.threshold=50</code>	Defines the threshold, as a percentage, for a file to be detected as a rename or a copy. The default threshold applied is 50% (if copy and/or rename are enabled). The default threshold applied is 50% (if copy and/or rename are enabled). Git diff and Git diff-tree <i>do not</i> apply the threshold. They only apply the default 50% threshold. The default threshold of 100 will be applied if the configured threshold that is 0 or less than 1.

<code>plugin.bitbucket-git.environment.variablesize=2000</code>	Defines the maximum number of files added to a single environment. This is for operating systems (and even the same operating system) have. They apply to environment variables intended to be low enough to fit on platforms out of the box, but not usable. It is configurable in configuration on some platform.
<code>plugin.bitbucket-git.pullrequest.merge.auto.forceadd=false</code>	Defines whether conflicted files are added using <code>Git add --force</code>. By default, this behavior is <i>off</i>. However, when merging across branches with <code>.gitignore</code> settings, enabling this system to create a conflicted file. The common ancestor will be used. Note: This value has <i>no effect</i> on merges. It is <i>only</i> applied during producing a pull request's merge.
<code>plugin.bitbucket-git.pullrequest.merge.auto.timeout=120</code>	Defines the maximum amount of time to perform a merge to support a pull request. Allowed to execute or idle. But generally do not produce output timeout. This value is in SECONDS.
<code>plugin.bitbucket-git.pullrequest.merge.real.timeout=300</code>	Defines the maximum amount of time to merge a pull request is allowed. Because the commands use <code>git merge</code> output, there is no separate timeout. This value is in SECONDS.
<code>plugin.bitbucket-git.repository.size.timeout=75</code>	Defines the maximum amount of time the size of a single repository repository and/or remote storage value. This value is in MILLISECONDS.

Server busy banners

Property	Description
<code>server.busy.on.ticket.rejected.within=5</code>	Controls how long a warning banner is displayed in the UI after a request is rejected due to excessive load. Value is in MINUTES. Using 0, or a negative value, disables displaying the banner.
<code>server.busy.on.queue.time=60</code>	Controls how long requests need to be queued before they cause a warning banner to appear. Value is in SECONDS. Using 0, or a negative value, disables displaying the banner.

Setup automation

If these properties are specified in `bitbucket.properties` when the Setup Wizard runs after installing Bitbucket Server, then those values will be used, and the Setup Wizard will not display the corresponding configuration screens.

You can use these properties to automate Bitbucket Server setup and remove the need to interact with the Bitbucket Server Setup Wizard when provisioning Bitbucket Server. See [Automated setup for Bitbucket Server](#).

Property	Description
<code>setup.displayName=displayName</code>	The display name for the Bitbucket Server instance.
<code>setup.baseUrl= https://bitbucket.yourcompany.com</code>	The base URL to use for the Bitbucket Server instance.
<code>setup.license=AAAB...\u000a1ev...\u000aA4N...</code>	The Bitbucket Server license. Use the the <code>\u000a</code> character to <i>not</i> break the license over multiple lines.
<code>setup.sysadmin.username=username</code>	Credentials for the system admin account.
<code>setup.sysadmin.password=password</code>	
<code>setup.sysadmin.displayName=John Doe</code>	The display name for the system admin account.
<code>setup.sysadmin.emailAddress=sysadmin@yourcompany.com</code>	The email address for the system admin account.

SMTP

Property	Description
<code>mail.timeout.connect=60</code> <code>mail.timeout.send=60</code> <code>mail.test.timeout.connect=30</code> <code>mail.test.timeout.send=30</code>	Controls timeouts for establishing an SMTP connection and sending an e-mail. Shorter timeouts should be applied for when sending test e-mails, as the test occurs in user time. Values are in SECONDS.
<code>mail.error.pause.log=300</code>	Controls how frequently logs will go to the standard log file about mail sending errors. All errors are logged to the <code>atlassian-bitbucket-mail.log</code> file, but Bitbucket Server will periodically log a warning to the standard log file if there are errors sending messages. Value is in SECONDS.
<code>mail.error.pause.retry=5</code>	Controls how long Bitbucket Server will wait before retrying to send a message if an error occurs. Value is in SECONDS.

<code>mail.threads=1</code>	Controls the number of threads to use for sending emails. Setting this to a higher value will put greater load on your mail server when Bitbucket Server generates a lot of emails, but will make Bitbucket Server clear its internal queue faster.
<code>mail.max.message.size=1048576</code>	Controls the maximum allowed size of a single mail message, which is the sum of the subject and body sizes. Value is in BYTES.
<code>mail.max.queue.size=157286400</code>	Controls the maximum allowed size for the mail queue (any new message will be rejected if the mail queue reaches that size). Value is in BYTES.

SSH command execution

Property	Description
<code>plugin.ssh.command.timeout.idle=86400</code>	Controls timeouts for all SSH commands, such as those that service Git and hg operations over SSH. The idle timeout configures how long the command is allowed to run without writing any output to the client. For SCM commands, the <code>plugin.*.backend.timeout.idle</code> properties defined above will be applied to the underlying process. The default value is 1 day. Value is in SECONDS.

SSH security

Property	Description
<code>plugin.ssh.disabled.ciphers</code>	Controls which default ciphers are disabled when executing all SSH commands. Non existent ciphers are ignored. Names are case sensitive. Example value: <code>arcfour128,3des-cbc</code> To enable additional ciphers see the KB article Disable default SSH algorithms .
<code>plugin.ssh.disabled.key.exchanges</code>	Controls which default key exchange algorithms are disabled when executing all SSH commands. Non existent key exchange algorithms are ignored. Names are case sensitive. Example value: <code>ecdh-sha2-nistp256,ecdh-sha2-nistp384</code> To enable additional key exchange algorithms see the KB article Disable default SSH algorithms .

<code>plugin.ssh.disabled.macs</code>	Controls which default macs are disabled when executing all SSH commands. Non existent macs are ignored. Names are case sensitive. Example value: <code>hmac-sha1-96, hmac-md5-96, hmac-md5</code> To enable additional macs see the KB article Disable default SSH algorithms.

Syntax highlighting

See [Configuring syntax highlighting for file extensions](#) for more information.

Bitbucket Server applies syntax highlighting to diffs as well as source files.

Property	Description
<code>syntax.highlighter.<MIME type>.executables=exe1,exe2</code>	Controls the language highlighter used for a given set of hashbang executables. The <MIME type> refers to the MIME type CodeMirror uses.
<code>syntax.highlighter.<MIME type>.extensions=ext1,ext2</code>	Controls the language highlighter used for a given set of file extensions. The <MIME type> refers to the MIME types CodeMirror uses.

Webhooks

See [POST service webhook for Bitbucket Server](#) for more information.

Property	Description
<code>plugin.com.atlassian.bitbucket.plugin.hook.threadPoolCoreSize=2</code>	Core size of thread pool – the default number of concurrent hooks notifications.
<code>plugin.com.atlassian.bitbucket.plugin.hook.threadPoolMaxSize=3</code>	Maximal size of thread pool – the maximum number of concurrent hooks notifications.

<code>plugin.com.atlassian.bitbucket.plugin.hook.queueSize=1024</code>	The maximum size of the queue which holds queued requests that are yet to be sent. When this size is exceeded the oldest unsent message will be dropped and a warning message logged.
<code>plugin.com.atlassian.bitbucket.plugin.hook.connectionTimeout=10000</code>	Connection timeout for hook request in MILLISECONDS. When the connection times out a warning message will be logged.
<code>plugin.com.atlassian.bitbucket.plugin.hook.changesetsLimit=500</code>	Limit of maximum count of changesets that will be sent in the POST data for a single ref change.
<code>plugin.com.atlassian.bitbucket.plugin.hook.changesLimit=100</code>	Limit of maximum count of changes for a single changeset in the POST data.

Proxying and securing Bitbucket Server

This page provides an overview of some common network topology options for running Bitbucket Server, including running Bitbucket Server behind a reverse proxy and securing access to Bitbucket Server by using HTTPS (HTTP over SSL).

Note that Bitbucket Server does not need to run behind a web server – it is capable of serving web requests directly using the bundled Tomcat application server. On this page, 'connecting to Bitbucket Server' really means connecting to Tomcat, which is used to serve Bitbucket Server content.

Connecting to Bitbucket Server directly over HTTP

Connecting directly to Bitbucket Server (that is, Tomcat) is the default install configuration, as described in the Bitbucket Server install

On this page:

- [Connecting to Bitbucket Server directly over HTTP](#)
- [Securing access to Bitbucket Server using HTTPS](#)
- [Using a reverse proxy for Bitbucket Server](#)
- [Securing a reverse proxy using HTTPS](#)

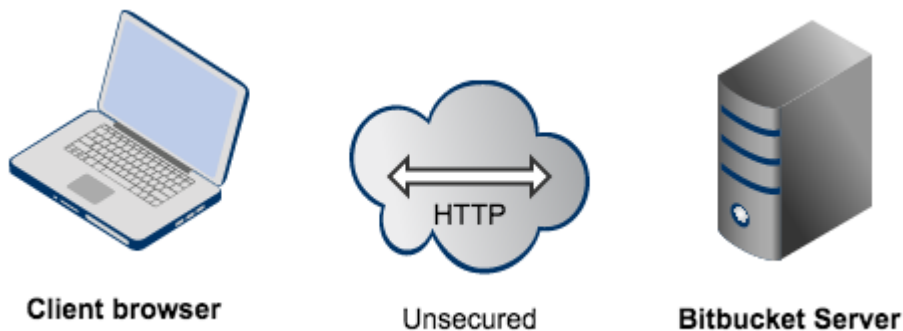
Related pages:

- [Changing the port that Bitbucket Server listens on](#)

documentation:

- [Getting started](#)

When set up this way, the user accesses Bitbucket Server directly over HTTP, without using SSL – all communication between the user's browser and Bitbucket Server will be unsecured.



You may also wish to consider the following:

- Bitbucket Server, by default, will listen for requests on port 7990 – this [port can be changed](#) if required.
- The address with which to access Bitbucket Server, by default, will be `http://<computer name>:7990`. Change the [base URL for Bitbucket Server](#) if required.
- You can [set the context path](#) for Bitbucket Server if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket Server.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).

Securing access to Bitbucket Server using HTTPS

Access to Bitbucket Server can be secured by enabling HTTPS (HTTP over SSL) for the Tomcat application server that is bundled with Bitbucket Server. You should consider doing this, and making secure access mandatory, if Bitbucket Server will be internet-facing and usernames, passwords and other proprietary data may be at risk.

When set up in this way, access to Bitbucket Server is direct, and all communication between the user's browser and Bitbucket Server will be secured using SSL.

See [Securing Bitbucket Server with Tomcat using SSL](#) for configuration details.



Note that:

- Bitbucket Server will listen for requests on port 8443. This port can be [changed if required](#).
- The address with which to access Bitbucket Server, by default, will be `https://<computer name>:8443`. Change the [base URL for Bitbucket Server](#) if required.
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket

Server.

- You can [set the context path](#) for Bitbucket Server if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket Server.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).

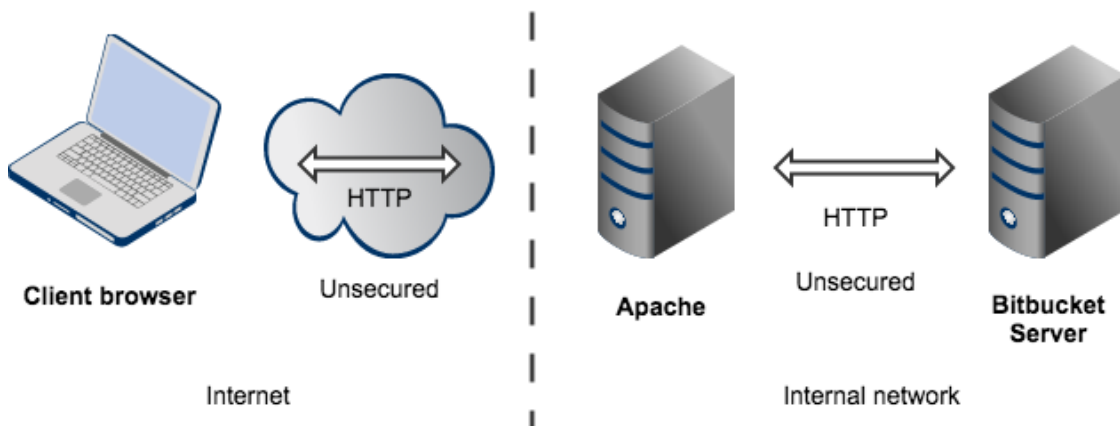
Using a reverse proxy for Bitbucket Server

You can run Bitbucket Server behind a reverse proxy, such as Apache HTTP Server. You may wish to do this if you want to:

- use a different port number to access Bitbucket Server, particularly if you are [Integrating JIRA Cloud with Bitbucket Server](#).
- use a different context path to access Bitbucket Server

When set up this way, external access to Bitbucket Server is via a reverse proxy, without using SSL. All communication between the user's browser and Apache, and so Bitbucket Server, will be unsecured, but users do not have direct access to Bitbucket Server. An example scenario is where Apache provides a gateway through which users outside the firewall can access Bitbucket Server.

See [Integrating Bitbucket Server with Apache HTTP Server](#) for configuration details.



Note that:

- Bitbucket Server, by default, will listen for requests on port 7990 – this port can be changed if required.
- Bitbucket Server (Tomcat) needs to know the URL (proxy name) that Apache serves.
- The address with which to access Bitbucket Server will be `http://<proxy name>:7990`. Change the [base URL for Bitbucket Server](#) if required.
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket Server.
- You can [set the context path](#) for Bitbucket Server if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket Server.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).

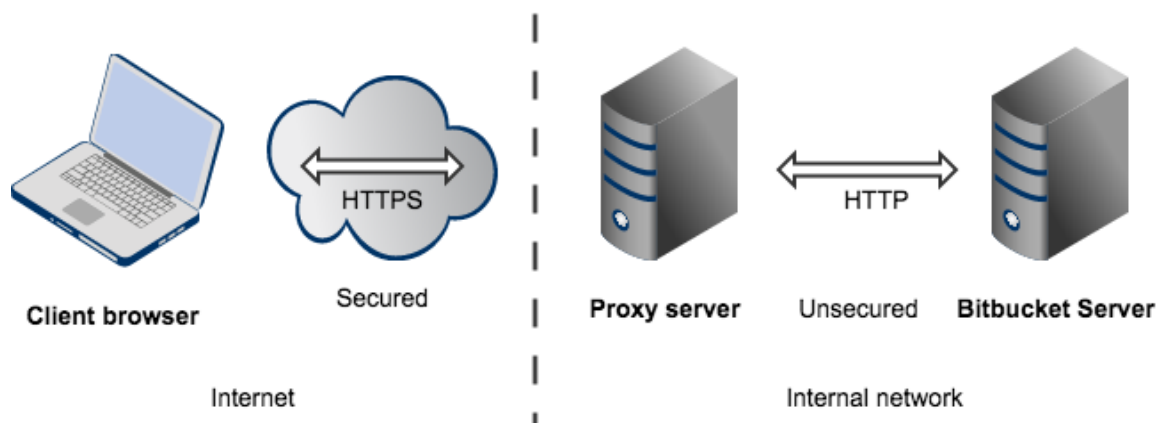
Securing a reverse proxy using HTTPS

You can run Bitbucket Server behind a reverse proxy, such as Apache HTTP Server or nginx, that is secured using HTTPS (HTTP over SSL). You should consider doing this, and making secure access mandatory, if usernames, passwords and other proprietary data may be at risk. An example scenario is where Apache HTTP Server provides a gateway through which users outside the firewall can access Bitbucket Server.

When set up in this way, external access to Bitbucket Server is via a reverse proxy, where external communication with the proxy uses HTTPS. All communication between the user's browser and the reverse proxy will be secured, whereas communication between the proxy and Bitbucket Server will not be secured (it doesn't use SSL).

See the following pages for configuration details:

- [Securing Bitbucket Server with Apache using SSL](#)
- [Securing Bitbucket Server behind nginx using SSL](#)



Note that:

- The reverse proxy (for example, Apache) will listen for requests on port 443.
- Bitbucket Server, by default, will listen for requests on port 7990. Bitbucket Server (Tomcat) needs to know the URL (proxy name) that the proxy serves.
- The address with which to access Bitbucket Server will be `https://<proxyName>:<proxyPort>/<context path>`, for example <https://mycompany.com:443/bitbucket>
- Any existing links with other applications will need to be reconfigured using this new URL for Bitbucket Server.
- Bitbucket Server (Tomcat) should be configured to refuse requests on port 7990 and to redirect those to the proxy on port 443.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).
- It would be possible to set up an SSL connection between the proxy server and Tomcat (Bitbucket Server), but that configuration is very unusual, and not recommended in most circumstances.
- Incidentally, note that Bitbucket Server 4.0 and later versions do not support `mod_auth_basic`.

Securing Bitbucket Server with Tomcat using SSL

This page is intended for administrators setting up Bitbucket Server for a small team. It describes how to enable HTTPS (HTTP over SSL) access for Tomcat, the webserver distributed with Bitbucket Server, using a self-signed certificate. You should consider doing this, and making secure access mandatory, if Bitbucket Server will be internet-facing and usernames, passwords and other proprietary data may be at risk.

If you are setting up a production instance of Bitbucket Server you should consider [using a CA certificate](#), briefly described below.

There are other network topology options for running Bitbucket Server, including running Bitbucket Server behind a reverse proxy. For an overview of some common options, see [Proxying and securing Bitbucket Server](#).

When Bitbucket Server is set up following the instructions on this page, access to Bitbucket Server is direct, and all communication between the user's browser and Bitbucket Server will be secured using SSL.

On this page:

- [1. Generate a self-signed certificate](#)
- [2. Configure HTTPS in Tomcat](#)
- [Exporting the self-signed certificate](#)
- [Requesting a CA certificate](#)
- [Troubleshooting](#)

Related pages:

- [Integrating Bitbucket Server with Apache HTTP Server](#)
- [Securing Bitbucket Server with Apache using SSL](#)

**Client browser****Secured****Bitbucket Server**

Note that:

- Bitbucket Server will listen for requests on port 8443. This port can be [changed if required](#).
- The address with which to access Bitbucket Server, by default, will be `https://<computer name>:8443`. Change the [base URL for Bitbucket Server](#) if required.
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket Server.
- You can [set the context path](#) for Bitbucket Server if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket Server.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).

Please note that Atlassian Support will refer SSL-related support to the issuing authority for the certificate. The documentation on this page is for reference only.

1. Generate a self-signed certificate

Self-signed certificates are useful where you require encryption but do not need to verify the website identity. They are commonly used for testing and on internal corporate networks (intranets). If you are setting up a production instance of Bitbucket Server you should consider [using a CA certificate](#), briefly described below.

Users may receive a warning that the site is untrusted and have to "accept" the certificate before they can access the site. This usually will only occur the first time they access the site.

The following approach to creating a certificate uses Java's [keytool](#). Other tools for generating certificates are available.

To generate a self-signed certificate:

Log in with the user account that Bitbucket Server will run under, and run the following command:

Windows	<code>"%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA</code>
Linux, Mac OS X	<code>\$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA</code>

This will create (if it doesn't already exist) a new `.keystore` file located in the home directory of the user you used to run the `keytool` command.

If you used the Bitbucket Server installer to install Bitbucket Server as a service on your system, the installer will have created a user account called `atlassian-bitbucket`. This account is locked (it cannot be used to log in to the system) and doesn't have a home directory. In this case you need to specify a location for the `.keystore` file using the `keystore` parameter like this:

Windows	<code>"%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA -keystore /path/to/keystore/bitbucket.jks</code>
Linux, Mac OS X	<code>\$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA -keystore /path/to/keystore/bitbucket.jks</code>

Note the following:

- When running the `keytool` command you will be prompted with: What is your first and last name?
You **must** enter the **fully qualified hostname** of the server running Bitbucket Server. This is the name you would type in your web browser after `'http:/'` (no port number) to access your Bitbucket Server installation. The qualified host name should match the base URL you have set in Bitbucket Server (without the port number).
- The `keytool` utility will also prompt you for two passwords: the keystore password and the key password for Tomcat.
You **must** use the same value for both passwords, and the value **must** be either:
 - "changeit", which is the default value Tomcat expects, or
 - any other value, but you must also specify it in `<Bitbucket home directory>/shared/server.xml` (see [the next section](#) for details)

2. Configure HTTPS in Tomcat

To configure HTTPS in Tomcat:

- Edit `<Bitbucket home directory>/shared/server.xml` and, at the bottom, before the `</Service>` tag, add this section (or uncomment this if it already exists):

```
<Connector port="8443"
  maxHttpHeaderSize="8192"
  SSLEnabled="true"
  maxThreads="150"
  minSpareThreads="25"
  maxSpareThreads="75"
  enableLookups="false"
  disableUploadTimeout="true"
  useBodyEncodingForURI="true"
  acceptCount="100"
  scheme="https"
  secure="true"
  clientAuth="false"
  sslProtocol="TLS" />
```

This enables SSL access on port 8443 (the default for HTTPS is 443, but 8443 is used here instead of

443 to avoid conflicts).

If you created the keystore somewhere else on the filesystem, add the `keystoreFile` attribute to the connector tag as well:

```
keystoreFile="/path/to/keystore/bitbucket.jks"
```

If your keystore password is anything other than "changeit", add the `keystorePass` attribute to the connector tag as well:

```
keystorePass="<password value>"
```

2. Comment out the existing Connector directive for port 7990 in `<Bitbucket home directory>/shared/server.xml`, so as to disable HTTP access, if you want all access to Bitbucket Server to make use of HTTPS. That is, comment out this directive:

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="8443"
  compression="on"

  compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,application/javascript,application/x-javascript" />
```

3. Start, or re-start, Bitbucket Server. You will be able to access Bitbucket Server at `https://localhost:8443/` in your browser.

Exporting the self-signed certificate

If Bitbucket Server will run as the user who ran the `keytool -genkey` command, *you do not need to export the certificate*.

You may need to export the self-signed certificate, so that you can import it into a different keystore, if Bitbucket Server will not be run as the user executing `keytool -genkey`. You can do so with the following command:

Windows	<code>"%JAVA_HOME%\bin\keytool" -export -alias tomcat -file file.cer</code>
Linux, Mac OS X	<code>\$JAVA_HOME/bin/keytool -export -alias tomcat -file file.cer</code>

If you generate the certificate as one user and run Bitbucket Server as another, you'll need to do the certificate export as the generating user and the import as the target user.

Requesting a CA certificate

Digital certificates that are issued by trusted 3rd party CAs (Certification Authorities) provide verification that your website does indeed represent your company.

When running Bitbucket Server in a production environment, you will need a certificate issued by a CA, such as [VeriSign](#), [DigiCert](#) or [Thawte](#). 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 then 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 [section above](#).
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:

Windows	<code>"%JAVA_HOME%\bin\keytool" -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <MY_KEYSTORE_FILENAME></code>
Linux, Mac OS X	<code>\$JAVA_HOME/bin/keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <MY_KEYSTORE_FILENAME></code>

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. Assuming your certificate is called "file.cer" whether obtained from a CA or self-generated, the following command will add the certificate to the keystore:

Windows	<code>"%JAVA_HOME%\bin\keytool" -import -alias tomcat -file file.cer</code>
Linux, Mac OS X	<code>\$JAVA_HOME/bin/keytool -import -alias tomcat -file file.cer</code>

Troubleshooting

Here are some troubleshooting tips if you are using a self-signed key created by `keytool`, or a CA certificate, as described above.

When you enter "<https://localhost:8443/>" in your browser, if you get a message such as "Cannot establish a connection to the server at localhost:8443", look for error messages in your `logs/catalina.out` log file. Here are some possible errors with explanations:

SSL + Apache + IE problems

Some people have reported errors when uploading attachments over SSL using Internet Explorer. This is due to an IE bug, and can be fixed in Apache by setting:

```
BrowserMatch ".MSIE." \
  nokeepalive ssl-unclean-shutdown \
  downgrade-1.0 force-response-1.0
```

[Google](#) has plenty more on this.

Can't find the keystore

```
java.io.FileNotFoundException: /home/user/.keystore (No such file or
directory)
```

This indicates that Tomcat cannot find the keystore. The `keytool` utility creates the keystore as a file called `.key` store in the current user's home directory. For Unix and Linux the home directory is likely to be `/home/<user name>`. For Windows it is likely to be `C:\User\<UserName>`.

Make sure you are running Bitbucket Server as the same user who created the keystore. If this is not the case, or if you are running Bitbucket Server on Windows as a service, you will need to specify where the keystore file is in `<Bitbucket home directory>/shared/server.xml`. Add the following attribute to the connector tag you uncommented:

```
keystoreFile="<location of keystore file>"
```

Incorrect password

```
java.io.IOException: Keystore was tampered with, or password was
incorrect
```

You used a different password than "changeit". You must either use "changeit" for both the keystore password and for the key password for Tomcat, or if you want to use a different password, you must specify it using the `keystorePass` attribute of the Connector tag, as described above.

Passwords don't match

```
java.io.IOException: Cannot recover key
```

You specified a different value for the keystore password and the key password for Tomcat. Both passwords must be the same.

Wrong certificate

```
javax.net.ssl.SSLException: No available certificate corresponds to the
SSL cipher suites which are enabled.
```

If the Keystore has more than one certificate, Tomcat will use the first returned unless otherwise specified in the SSL Connector in `<Bitbucket home directory>/shared/server.xml`.

Add the `keyAlias` attribute to the Connector tag you uncommented, with the relevant alias, for example:

```
<Connector port="8443"
  maxHttpHeaderSize="8192"
  SSLEnabled="true"
  maxThreads="150"
  minSpareThreads="25"
  maxSpareThreads="75"
  enableLookups="false"
  disableUploadTimeout="true"
  useBodyEncodingForURI="true"
  acceptCount="100"
  scheme="https"
  secure="true"
  clientAuth="false"
  sslProtocol="TLS"
  keystoreFile="/opt/local/.keystore"
  keystorePass="removed"
  keyAlias="tomcat"/>
```

Using Apache Portable Runtime

APR uses a different SSL engine, and you will see an exception like this in your logs:

```
SEVERE: Failed to initialize connector [Connector[HTTP/1.1-8443]]
LifecycleException: Protocol handler initialization failed:
java.lang.Exception: No Certificate file
specified or invalid file format
```

The reason for this is that the APR Connector uses OpenSSL and cannot use the keystore in the same way.

You can rectify this in one of two ways:

Use the Http11Protocol to handle SSL connections

Edit the server.xml so that the SSL Connector tag you just uncommented specifies the Http11Protocol instead of the APR protocol:

```
<Connector port="8443"
  protocol="org.apache.coyote.http11.Http11Protocol"
  maxHttpHeaderSize="8192"
  SSLEnabled="true"
  keystoreFile="${user.home}/.keystore"
  maxThreads="150"
  enableLookups="false"
  disableUploadTimeout="true"
  acceptCount="100"
  scheme="https"
  secure="true"
  clientAuth="false"
  sslProtocol="TLS"
  useBodyEncodingForURI="true" />
```

Configure the Connector to use the APR protocol

This is only possible if you have PEM encoded certificates and private keys. If you have used OpenSSL to generate your key, then you will have these PEM encoded files - in all other cases contact your certificate provider for assistance.

```
<Connector port="8443"
  maxThreads="200"
  scheme="https"
  secure="true"
  SSLEnabled="true"
  SSLCertificateFile="${user.home}/certificate.pem"
  SSLCertificateKeyFile="${user.home}/key.pem"
  clientAuth="optional"
  SSLProtocol="TLSv1"/>
```

Enabling client authentication

To enable client authentication in Tomcat, ensure that the value of the `clientAuth` attribute in your Connector element of your Tomcat's server.xml file is true.

```
<Connector
...
  clientAuth="true"
... />
```

For more information about Connector element parameters, please refer to the 'SSL Support' section of the [Tomcat 6.0](#) documentation.

Wrong certificate type

If the certificate from the CA is in PKSC12 format, add the `keystoreType` attribute to the SSL Connector in `<Bitbucket home directory>/shared/server.xml`.

```
keystoreFile="/opt/local/wildcard_atlassian_com.p12"
keystorePass="removed"
keystoreType="PKCS12"/>
```

Certificate chain is incomplete

If the root certificate and intermediary certificate(s) aren't imported into the keystore before the entity/domain certificate, you will see the following error:

```
[root@dev atlas]# /usr/java/jdk1.7.0_17/bin/keytool -import -alias
tomcat -file my_entity_cert.crt
Enter keystore password:
keytool error: java.lang.Exception: Failed to establish chain from reply
```

Most likely, the CA sent a compressed file containing several certificates. The import order matters so you must import the root certificate first, followed by one or many intermediate certificates, followed lastly by the entity/domain certificate. There are many resources online that provide guidance for [certificate installation for Tomcat \(Java-based\) web servers using keytool](#).

Integrating Bitbucket Server with Apache HTTP Server

Error rendering macro 'viewport-redirect' : null

This page explains how to establish a network topology in which Apache HTTP Server acts as a [reverse proxy](#) for Bitbucket Server. Typically, such a configuration would be used when Bitbucket Server is installed in a protected zone 'behind the firewall', and Apache HTTP Server provides a gateway through which users outside the firewall can access Bitbucket Server. You may wish to do this if you want to:

- use a different port number to access Bitbucket Server
- use a different context path to access Bitbucket Server

Be aware that Bitbucket Server does not need to run behind a web server, since it is capable of serving web requests directly; to secure Bitbucket Server when run in this way see [Securing Bitbucket Server with Tomcat using SSL](#). For an overview of other network topology options, see [Proxying and securing Bitbucket Server](#). Otherwise, if you want to install Bitbucket Server in an environment that incorporates Apache HTTP Server, this document is for you.

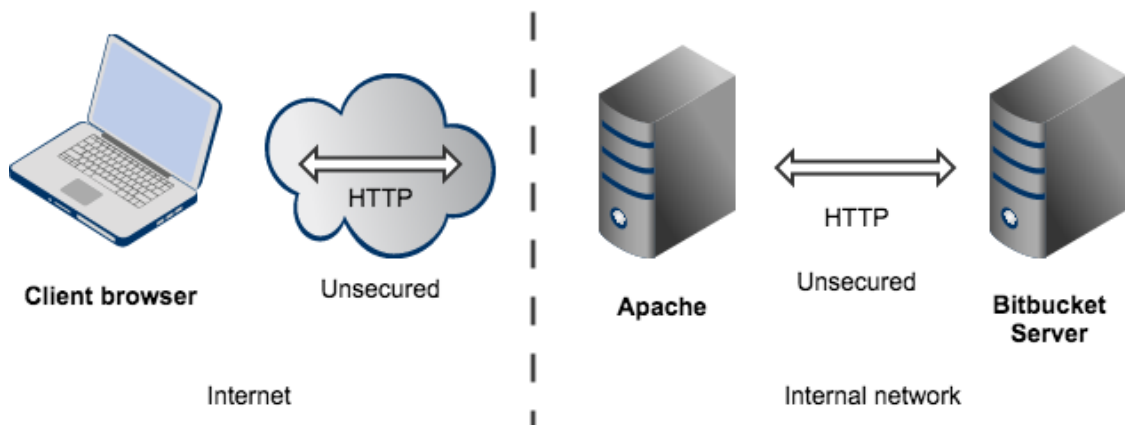
When Bitbucket Server is set up following the instructions on this page, external access to Bitbucket Server uses a reverse proxy, without using SSL. All communication between the user's browser and Apache, and so Bitbucket Server, will be unsecured, but users do not have direct access to Bitbucket Server.

On this page:

- [About using Apache software](#)
- [Step 1: Configure the Tomcat Connector](#)
- [Step 2: Change Bitbucket Server's base URL](#)
- [Step 3 \(optional\): Set a context path for Bitbucket Server](#)
- [Step 4: Enable mod_proxy and mod_proxy_http in Apache HTTP Server](#)
- [Step 5: Configure mod_proxy to map requests to Bitbucket Server](#)
- [Step 6: Configure mod_proxy to disable forward proxying](#)
- [Step 7: Allow proxying to Bitbucket Server from everywhere](#)
- [Step 8 \(optional\): Configure Apache HTTP Server for SSL](#)
- [A note about application links](#)
- [Troubleshooting](#)

Related pages:

- [Securing Bitbucket Server with Apache using SSL](#)
- [Securing Bitbucket Server with Tomcat using SSL](#)



Note that:

- Bitbucket Server, by default, will listen for requests on port 7990 – this port can be changed if required.
- Bitbucket Server (Tomcat) needs to know the URL (proxy name) that Apache serves.
- The address with which to access Bitbucket Server will be `http://<proxy name>:7990`. Change the [base URL for Bitbucket Server](#) if required.
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket Server.
- You can [set the context path](#) for Bitbucket Server if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket Server.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).

About using Apache software

This section has general information pertaining to the use of [Apache HTTP Server](#) and [Apache Tomcat](#). It is important that you read this section before proceeding to the steps that follow.

Configuring Tomcat 7

The Bitbucket Server distribution includes an instance of Tomcat 7, the configuration of which is determined by the contents of the `<Bitbucket home directory>/shared/server.xml` file. Note that any changes that you make to the `server.xml` file will only be effective upon starting or re-starting Bitbucket Server.

You may find it helpful to refer to the [Apache Tomcat 7.0 Proxy Support HowTo](#) page.

Configuring Apache HTTP Server

Since Apache HTTP Server is not an Atlassian product, Atlassian does not guarantee to provide support for its configuration. You should consider the material on this page to be for your information only; use it at your own risk. If you encounter problems with configuring Apache HTTP Server, we recommend that you refer to the [Apache HTTP Server Support](#) page.

Note that Bitbucket Server 2.10 and later versions do not support `mod_auth_basic`.

You may find it helpful to refer to the [Apache HTTP Server Documentation](#), which describes how you can control Apache HTTP Server by changing the contents of the `httpd.conf` file. The section on [Apache Module `mod\_proxy`](#) is particularly relevant. Note that any changes you make to the `httpd.conf` file will only be effective upon starting or re-starting Apache HTTP Server.

This document relates to Apache HTTP Server version 2.4.2; the configuration of other versions may differ.

Step 1: Configure the Tomcat Connector

Find the normal (non-SSL) Connector directive in Tomcat's `<Bitbucket home directory>/shared/server.xml` file, and add the `scheme`, `proxyName`, and `proxyPort` attributes as shown below. Instead of `mycompany.com`, set the `proxyName` attribute to your domain name that Apache HTTP Server will be configured to serve. This informs Bitbucket Server of the domain name and port of the requests that reach it via Apache HTTP Server, and is important to the correct operation of the Bitbucket Server functions that construct URLs.

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="8443"
  compression="on"

  compressableMimeType="text/html,text/xml,text/plain,text/css,application
/json,application/javascript,application/x-javascript"
  scheme="http"
  proxyName="mycompany.com"
  proxyPort="80" />
```

Note: Apache HTTP Server's [ProxyPreserveHost](#) directive is another way to have the hostname of the incoming request recognized by Bitbucket Server instead of the hostname at which Bitbucket Server is actually running. However, the `ProxyPreserveHost` directive does not cause the scheme to be properly set. Since we have to mess with Tomcat's Connector directive anyway, we recommend that you stick with the above-described approach, and don't bother to set the `ProxyPreserveHost` in Apache HTTP Server.

For more information about configuring the Tomcat Connector, refer to the [Apache Tomcat 7.0 HTTP Connector Reference](#).

Step 2: Change Bitbucket Server's base URL

After re-starting Bitbucket Server, open a browser window and log into Bitbucket Server using an administrator account. Go to the Bitbucket Server administration area and click **Server settings** (under 'Settings'), and change **Base URL** to match the proxy URL (the URL that Apache HTTP Server will be serving).

Step 3 (optional): Set a context path for Bitbucket Server

By default, Bitbucket Server is configured to run with an empty context path; in other words, from the 'root' of the

server's name space. In that default configuration, Bitbucket Server is accessed at:

```
http://localhost:7990/
```

It's perfectly fine to run Bitbucket Server with the empty context path as above. Alternatively, you can set a context path by changing the `Context` directive in Tomcat's `<Bitbucket home directory>/shared/server.xml` file:

```
<Context path="/bitbucket"
docBase="${catalina.home}/atlassian-bitbucket" reloadable="false"
useHttpOnly="true">
    ....
</Context>
```

If you do set a context path, it is important that the same path be used in [Step 5](#), when setting up the `ProxyPass` and `ProxyPassReverse` directives. You should also append the context path to Bitbucket Server's base URL (see [Step 2](#)).

Step 4: Enable `mod_proxy` and `mod_proxy_http` in Apache HTTP Server

In the [mod_proxy](#) documentation, you will read that `mod_proxy` can be used as a forward proxy, or as a reverse proxy (gateway); you want the latter. Where the `mod_proxy` documentation mentions '*origin server*', it refers to your Bitbucket Server instance. Unless you have a good reason for doing otherwise, load `mod_proxy` and `mod_proxy_http` dynamically, using the [LoadModule directive](#); that means un-commenting the following lines in the `httpd.conf` file:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

Experienced administrators may be aware of the Apache Connector module, `mod_jk`. Atlassian does not recommend use of the `mod_jk` module with Bitbucket Server, since it has proven itself to be less reliable than `mod_proxy`.

Step 5: Configure `mod_proxy` to map requests to Bitbucket Server

To configure `mod_proxy` for use with Bitbucket Server, you need to use the `ProxyPass` and `ProxyPassReverse` directives in Apache HTTP Server's `httpd.conf` file as follows:

```
ProxyPass          / http://localhost:7990/ connectiontimeout=5
timeout=300
ProxyPassReverse   / http://localhost:7990/
```

Suppose Apache HTTP Server is configured to serve the `mycompany.com` domain; then the above directives tell Apache HTTP Server to forward web requests of the form `http://mycompany.com/*` to the Tomcat connector (Bitbucket Server) running on port 7990 on the same machine.

The `connectiontimeout` attribute specifies the number of seconds Apache HTTP Server waits for the creation of a connection to Bitbucket Server.

The `timeout` attribute specifies the number of seconds Apache HTTP Server waits for data to be sent to Bitbucket Server.

If you set up a context path for Bitbucket Server in [Step 3](#), you'll need to use that context path in your `ProxyPass` and `ProxyPassReverse` directives. Suppose your context path is set to `/bitbucket`, the directives would be as follows:

```
ProxyPass          /bitbucket http://localhost:7990/bitbucket
connectiontimeout=5 timeout=300
ProxyPassReverse   /bitbucket http://localhost:7990/bitbucket
```

If Bitbucket Server is to run on a different domain and/or different port, you should use that domain and/or port number in the `ProxyPass` and `ProxyPassReverse` directives; for example, suppose that Bitbucket Server will run on port 9900 on `private.mycompany.com` under the context path `/bitbucket`, then you would use the following directives:

```
ProxyPass          /bitbucket http://private.mycompany.com:9900/bitbucket
connectiontimeout=5 timeout=300
ProxyPassReverse   /bitbucket http://private.mycompany.com:9900/bitbucket
```

Step 6: Configure `mod_proxy` to disable forward proxying

If you are using Apache HTTP Server as a reverse proxy only, and not as a forward proxy server, you should turn forward proxying off by including a `ProxyRequests` directive in the `httpd.conf` file, as follows:

```
ProxyRequests Off
```

Step 7: Allow proxying to Bitbucket Server from everywhere

Strictly speaking, this step is unnecessary because access to proxied resources is unrestricted by default. Nevertheless, we explicitly allow access to Bitbucket Server from any host so that this policy will be applied regardless of any subsequent changes to access controls at the global level. Use the `Proxy` directive in the `httpd.conf` file as follows:

```
<Proxy *>
    Order Deny,Allow
    Allow from all
</Proxy>
```

The `Proxy` directive provides a context for the directives that are contained within its delimiting tags. In this case, we specify a wild-card url (the asterisk), which applies the two contained directives to all proxied requests.

The `Order` directive controls the order in which any `Allow` and `Deny` directives are applied. In the above configuration, we specify "Deny, Allow", which tells Apache HTTP Server to apply any `Deny` directives first, and if any match, the request is denied unless it also matches an `Allow` directive. In fact, "Deny, Allow" is the default; we include it merely for the sake of clarity. Note that we specify one `Allow` directive, which is described below, and don't specify any `Deny` directives.

The `Allow` directive, in this context, controls which hosts can access Bitbucket Server via Apache HTTP Server. Here, we specify that all hosts are allowed access to Bitbucket Server.

Step 8 (optional): Configure Apache HTTP Server for SSL

If you want to set up SSL access to Bitbucket Server, follow the instructions on [Securing Bitbucket Server with Apache using SSL](#). When you are finished, users will be able to make secure connections to Apache HTTP Server; connections between Apache HTTP Server and Bitbucket Server will remain unsecured (not using SSL). If you don't want to set up SSL access, you can skip this section entirely.

Note: It would be possible to set up an SSL connection between Apache HTTP Server and Tomcat (Bitbucket Server), but that configuration is very unusual, and not recommended in most circumstances.

A note about application links

When an [application link](#) is established between Bitbucket Server and another Atlassian product (for example JIRA), and Bitbucket Server is operating behind Apache HTTP Server, the link from the other product to Bitbucket Server must be via the proxy URL; that is, the 'reciprocal URL' from, say JIRA, to Bitbucket Server must match the proxy name and port that you set at [Step 1](#).

Troubleshooting

In general, if you are having problems:

1. Ensure that Bitbucket Server works as expected when running directly from Tomcat on <http://localhost:7990/bitbucket>.
2. Watch the log files (usually in `/var/log/httpd/` or `/var/log/apache2/`). Check that you have a `LogLevel` directive in your `httpd.conf`, and turn up logging (`LogLevel debug`) to get more info.
3. Check out the [Bitbucket Server Knowledge Base](#).

In particular:

- On **Fedora Core 4** people have reported 'permission denied' errors when trying to get `mod_proxy` (and `mod_jk`) working. Disabling SELinux (`/etc/selinux/config`) apparently fixes this.
- Some users have reported problems with user sessions being hijacked when the `mod_cache` module is enabled. If you have such problems, disable the `mod_cache` module. Note that this module is enabled by default in some Apache HTTP Server version 2 distributions.

Securing Bitbucket Server with Apache using SSL

You can run Bitbucket Server behind a reverse proxy, such as Apache HTTP Server or nginx, that is secured using HTTPS (HTTP over SSL). You should consider doing this, and making secure access mandatory, if usernames, passwords and other proprietary data may be at risk.

There are other network topology options for running Bitbucket Server; for an overview of some common options, see [Proxying and securing Bitbucket Server](#).

When Bitbucket Server is set up following the instructions on this page, external access to Bitbucket Server is via Apache HTTP Server as a reverse proxy, where external communication with the proxy uses HTTPS. All communication between the user's browser and Apache will be secured, whereas communication between Apache and Bitbucket Server will not be secured (it doesn't use SSL).

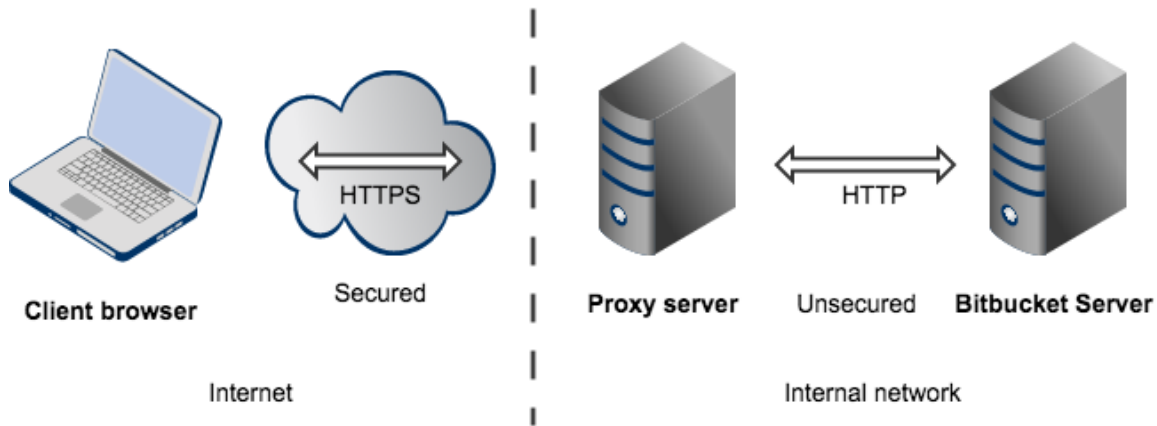
- The steps on this page would normally be performed after [integrating Bitbucket Server with Apache HTTP Server](#).

On this page:

- [Step 1: Configure the Tomcat Connector for SSL](#)
- [Step 2: Set up a virtual host in Apache HTTP Server](#)
- [Step 3: Create SSL certificate and key files](#)
- [Step 4: Update the base URL to use HTTPS](#)
- [Using a self-signed certificate](#)

Related pages:

- [Integrating Bitbucket Server with Apache HTTP Server](#)
- [Securing Bitbucket Server with Tomcat using SSL](#)
- [Securing Bitbucket Server behind nginx using SSL](#)



Note that:

- The reverse proxy (for example, Apache) will listen for requests on port 443.
- Bitbucket Server, by default, will listen for requests on port 7990. Bitbucket Server (Tomcat) needs to know the URL (proxy name) that the proxy serves.
- The address with which to access Bitbucket Server will be `https://<proxyName>:<proxyPort>/<context path>`, for example <https://mycompany.com:443/bitbucket>
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket Server.
- Bitbucket Server (Tomcat) should be configured to refuse requests on port 7990 and to redirect those to the proxy on port 443.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).
- It would be possible to set up an SSL connection between the proxy server and Tomcat (Bitbucket Server), but that configuration is very unusual, and not recommended in most circumstances.
- Incidentally, note that Bitbucket Server 4.0 and later versions do not support `mod_auth_basic`.

Step 1: Configure the Tomcat Connector for SSL

Find the normal (non-SSL) Connector directive in Tomcat's `<Bitbucket home directory>/shared/server.xml` file, and change the `redirectPort`, `scheme`, `proxyName` and `proxyPort` attributes as follows:

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="443"
  compression="on"

  compressableMimeType="text/html,text/xml,text/plain,text/css,application
  /json,application/javascript,application/x-javascript"
  secure="true"
  scheme="https"
  proxyName="mycompany.com"
  proxyPort="443" />
```

The `redirectPort` directive causes Tomcat-initiated redirections to secured resources to use the specified port. Right now, the Bitbucket Server configuration of Tomcat does not involve Tomcat-initiated redirections, so the change to `redirectPort` is redundant. Nevertheless, we suggest that you change it as directed above for the sake of completeness.

Start, or restart, Bitbucket Server.

Step 2: Set up a virtual host in Apache HTTP Server

Un-comment the following LoadModule directive in Apache HTTP Server's `httpd.conf` file:

```
LoadModule ssl_module modules/mod_ssl.so
```

Add the following directives to the `httpd.conf` file:

```
Listen 443
<VirtualHost *:443>
    SSLEngine On
    SSLCertificateFile    "/usr/local/apache2/conf/server.crt"
    SSLCertificateKeyFile "/usr/local/apache2/conf/server.key"
    SSLCertificateChainFile "/usr/local/apache2/conf/server.crt"
    ProxyPass             / http://localhost:7990/ connectiontimeout=5
    timeout=300
    ProxyPassReverse      / http://localhost:7990/
</VirtualHost>
```

The `Listen` directive instructs Apache HTTP Server to listen for incoming requests on port 443. Actually, we could omit that directive in this case, since Apache HTTP Server listens for `https` requests on port 443 by default. Nevertheless, it's good to make one's intentions explicit.

The `VirtualHost` directive encloses a number of child directives that apply only and always to requests that arrive at port 443. Since our `VirtualHost` block does not include a `ServerName` directive, it inherits the server name from the main server configuration.

The `SSLEngine` directive toggles the use of the SSL/TLS Protocol Engine. In this case, we're using it to turn SSL on for all requests that arrive at port 443.

The `SSLCertificateFile` directive tells Apache HTTP Server where to find the PEM-encoded certificate file for the server.

The `SSLCertificateKeyFile` directive tells Apache HTTP Server where to find the PEM-encoded private key file corresponding to the certificate file identified by the `SSLCertificateFile` directive. Depending on how the certificate file was generated, it may contain a RSA or DSA private key file, making the `SSLCertificateKeyFile` directive redundant; however, Apache strongly discourages that practice. The recommended approach is to separate the certificate and the private key. If the private key is encrypted, Apache HTTP Server will require a pass phrase to be entered when it starts up.

The `SSLCertificateChainFile` is optional. Please consult with the CA vendor to verify if this is required. This directive sets the optional all-in-one file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate.

The `ProxyPass` and `ProxyPassReverse` directives should be set up in the manner described in [Step 5](#) of the [Integrating Bitbucket Server with Apache HTTP Server](#) page. In particular, if Bitbucket Server is to run on a separate machine from Apache, you should use that domain (and perhaps the port number and context path) in the `ProxyPass` and `ProxyPassReverse` directives.

For more information about the support for SSL in Apache HTTP Server, refer to the [Apache SSL/TLS Encryption](#) manual. In addition, you will find lots of relevant information in the `<apache>/conf/extra/httpd-ssl.conf` file, which is included in the standard Apache distribution.

Start, or restart, Apache.

Step 3: Create SSL certificate and key files

In [Step 2](#), you specified `server.crt` and `server.key` as the certificate file and private key file respectively. Those two files must be created before we can proceed. This step assumes that [OpenSSL](#) is installed on your

server.

Generate a server key file:

```
openssl genrsa -des3 -out server.key 2048
```

You will be asked to provide a password. Make sure that the password is strong because it will form the one real entry point into the SSL encryption set-up. *Make a note of the password because you'll need it when starting Apache HTTP Server later.*

If you don't wish to specify a password, don't use the `-des3` option in the command above.

Generate a certificate request file (`server.csr`):

```
openssl req -new -key server.key -out server.csr
```

Generate a self-signed certificate (`server.crt`):

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

The above command generates a self-signed certificate that is valid for one year. You can use the certificate signing request to purchase a certificate from a [certificate authority](#). For testing purposes though, the self-signed certificate will suffice. Copy the certificate file and private key file to the locations you specified in [Step 2](#).

```
cp server.key /usr/local/apache2/conf/
cp server.crt /usr/local/apache2/conf/
```

Step 4: Update the base URL to use HTTPS

Open a browser window and log into Bitbucket Server using an administrator account. Go to the Bitbucket Server administration area and click **Server settings** (under 'Settings'). Change **Base URL** to use HTTPS, for example, "https://bitbucket.mycompany.com").

Using a self-signed certificate

There are two implications of using the self-signed certificate:

- When you access Bitbucket Server in a web browser, you can expect a warning to appear, alerting you that an un-trusted certificate is in use. Before proceeding you will have to indicate to the browser that you trust the certificate.
- When you perform a Git clone operation, SSL verification will fail.

The SSL verification error message will look something like this:

```
error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify
failed while accessing https://justme@mycompany/git/TP/test.git
```

It's easy to fix. Turn SSL verification off for individual Git operations by setting the `GIT_SSL_NO_VERIFY` environment variable. In Unix, you can set the variable in-line with Git commands as follows:

```
GIT_SSL_NO_VERIFY=true git clone https://justme@mycompany/git/TP/test.git
```

In Windows you have to set the variable in a separate shell statement:

```
set GIT_SSL_NO_VERIFY=true
git clone https://justme@mycompany/git/TP/test.git
```

Once you have purchased and installed a signed certificate from a certificate authority, you will no longer have to include the `GIT_SSL_NO_VERIFY` modifier.

Securing Bitbucket Server behind nginx using SSL

This page describes how to establish a network topology in which the nginx server acts as a [reverse proxy](#) for

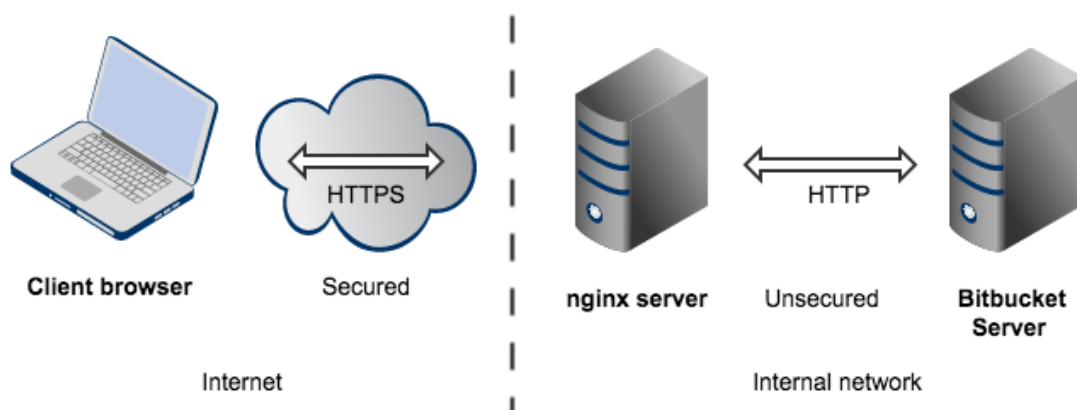
Bitbucket Server. Typically, such a configuration would be used when Bitbucket Server is installed in a protected zone 'behind the firewall', and nginx provides a gateway through which users outside the firewall can access Bitbucket Server.

The configuration described on this page results in a scenario where:

- External client connections with nginx are secured using SSL. Connections between nginx and Bitbucket Server are unsecured.
- Bitbucket Server and nginx run on the same machine.
- Bitbucket Server is available at <https://mycompany.com:7990/bitbucket>.

On this page:

- [Step 1: Configure the Tomcat Connector](#)
- [Step 2: Set a context path for Bitbucket Server](#)
- [Step 3: Change Bitbucket Server's base URL](#)
- [Step 4: Configure nginx](#)
- [Resources](#)



Please note that:

- We assume that you already have a running instance of nginx. If not, refer to the [nginx documentation](#) for instructions on downloading and installing nginx.
- SSL certificates must be installed on the server machine.
- Any existing [links with other applications](#) will need to be reconfigured using the new URL for Bitbucket Server.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).

Be aware that Bitbucket Server does not need to run behind a web server, since it is capable of serving web requests directly; to secure Bitbucket Server when run in this way see [Securing Bitbucket Server with Tomcat using SSL](#). Otherwise, if you want to install Bitbucket Server in an environment that incorporates nginx, this document is for you. (You can of course run Bitbucket Server behind nginx without securing client connections to nginx using SSL – we don't describe this option on this page.)

Note that the [Atlassian Support Offering](#) does not cover nginx integration. Assistance with nginx may be obtained through the Atlassian community from answers.atlassian.com or from an [Atlassian Expert](#).

Step 1: Configure the Tomcat Connector

Find the normal (non-SSL) Connector directive in Tomcat's `<Bitbucket home directory>/shared/server.xml` file, and add the `scheme`, `proxyName`, and `proxyPort` attributes as shown below. Instead of `mycompany.com`, set the `proxyName` attribute to your domain name that the nginx server will be configured to serve. This informs Bitbucket Server of the domain name and port of the requests that reach it via nginx, and is important for the correct operation of the Bitbucket Server functions that construct URLs.

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="443"
  compression="on"

  compressableMimeType="text/html,text/xml,text/plain,text/css,application
/json,application/javascript,application/x-javascript"
  secure="true"
  scheme="https"
  proxyName="mycompany.com"
  proxyPort="443" />
```

For more information about configuring the Tomcat Connector, refer to the [Apache Tomcat 7.0 HTTP Connector Reference](#).

Step 2: Set a context path for Bitbucket Server

By default, Bitbucket Server is configured to run with an empty context path; in other words, from the 'root' of the server's name space. In that default configuration, Bitbucket Server would be accessed at:

```
http://mycompany.com:7990/
```

For the example configuration on this page, we want Bitbucket Server to be accessed at:

```
https://mycompany.com:7990/bitbucket
```

In Tomcat's `<Bitbucket home directory>/shared/server.xml` file, set the context path to `/bitbucket`:

```
<Context path="/bitbucket"
  docBase="${catalina.home}/atlassian-bitbucket" reloadable="false"
  useHttpOnly="true">
  ....
</Context>
```

If you use a context path, it is important that the same path is:

- appended to the context path of Bitbucket Server's base URL ([Step 3](#)).
- used when setting up the location for the `proxy_pass` directive ([Step 4](#)).

Step 3: Change Bitbucket Server's base URL

After re-starting Bitbucket Server, open a browser window and log into Bitbucket Server using an administrator account. Go to the Bitbucket Server administration area and click **Server settings** (under 'Settings'), and change **Base URL** to match the proxy URL (the URL that the nginx server will be serving).

For this example, use `http://mycompany.com:7990/bitbucket` (Note the context path included with this.)

Step 4: Configure nginx

Edit `/etc/nginx/nginx.conf`, using the example server configuration below, to configure nginx as a proxy server.

Put the `proxy_pass` directive in the location block, and specify the protocol, name and port of the proxied server in the parameter (in our case, it is `http://localhost:7990`):


```

server {
    listen      443;
    server_name mycompany.com;

    ssl         on;
    ssl_certificate      <path/to/your/certificate>;
    ssl_certificate_key  <path/to/your/certificate/key>;
    ssl_session_timeout 5m;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers          HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    # Optional optimisation - please refer to
    http://nginx.org/en/docs/http/configuring_https_servers.html
    # ssl_session_cache  shared:SSL:10m;
    location /bitbucket {
        proxy_pass      http://localhost:7990/bitbucket;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_redirect   off;
    }
}

```

Refer to http://nginx.org/en/docs/http/nginx_http_proxy_module.html.

Changes made in the configuration file will not be applied until the command to reload configuration is sent to nginx or it is restarted. To reload the configuration, execute:

```
nginx -s reload
```

This command should be executed under the same user that started nginx.

Resources

You may find the following resources helpful in setting up Bitbucket Server behind nginx:

- http://nginx.org/en/docs/http/configuring_https_servers.html
- <http://www.cyberciti.biz/tips/using-nginx-as-reverse-proxy.html>
- <https://mywushublog.com/2012/08/atlassian-tools-and-nginx/>

Securing Bitbucket Server behind HAProxy using SSL

This page describes how to establish a network topology in which the HAProxy server acts as a [reverse proxy](#) for Bitbucket Server. Typically, such a configuration would be used when either when:

1. Bitbucket Server is installed in a protected zone 'behind the firewall', and HAProxy

provides a gateway through which users outside the firewall can access Bitbucket Server.

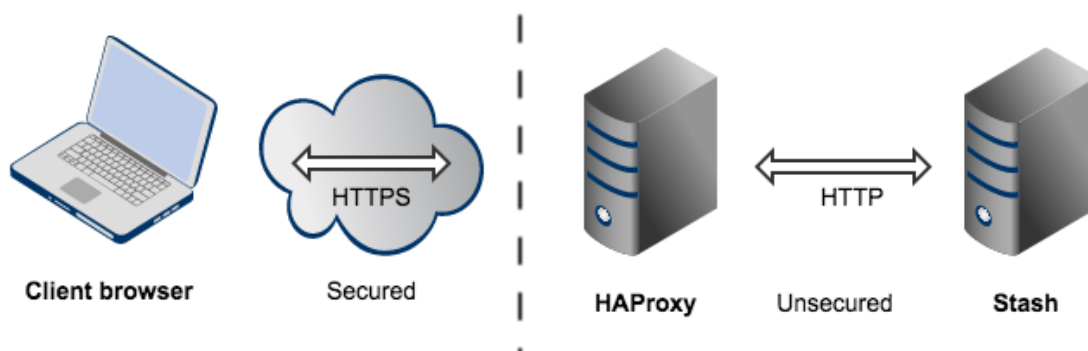
2. Bitbucket Server needs to be served on protected ports (e.g. ports < 1024 on Linux). Bitbucket Server cannot access these ports directly as *it must not* be run as a privileged user (e.g. root). In this case HAProxy can bind to these ports and forward the requests to Bitbucket Server.

On this page:

- [Step 1: Set a context path for Bitbucket Server](#)
- [Step 2: Change Bitbucket Server's base URL](#)
- [Step 3: Configure the Tomcat Connector](#)
- [Step 4: Configure HAProxy](#)
- [\(Optional\) Step 4: Redirect SSH connections](#)
- [Resources](#)

The configuration described on this page results in a scenario where:

- External client connections with HAProxy are secured using SSL. Connections between HAProxy and Bitbucket Server are unsecured.
- Bitbucket Server and HAProxy run on the same machine.
- Bitbucket Server is currently available at <http://mycompany.com:7990>.
- Bitbucket Server is to be made available at <https://mycompany.com/bitbucket>.



Please note that:

- We assume that you already have a running instance of HAProxy.
- SSL certificates must be installed on the server machine.
- Any existing [links with other applications](#) will need to be reconfigured using the new URL for Bitbucket Server.
- Securing Git operations between the user's computer and Bitbucket Server is a separate consideration - see [Enabling SSH access to Git](#).
- It is also possible to get Bitbucket Server to directly use SSL without the help of a proxy as documented in [Securing Bitbucket Server with Tomcat using SSL](#).

Note that the [Atlassian Support Offering](#) does not cover HAProxy integration, but you can get assistance with HAProxy from the Atlassian community on answers.atlassian.com, or from an [Atlassian Expert](#).

Step 1: Set a context path for Bitbucket Server

Bitbucket Server and HAProxy need to be serving from the same context. Bitbucket Server is currently accessed at <http://mycompany.com:7990>. It needs to be changed to serve from <http://mycompany.com:7990/bitbucket> to match context <https://mycompany.com/bitbucket>.

In Tomcat's <Bitbucket Server home directory>/shared/server.xml file, set the context path to /bitbucket:

```
<Context path="/bitbucket"
docBase="${catalina.home}/atlassian-bitbucket" reloadable="false"
useHttpOnly="true">
    . . .
</Context>
```

- If you use a context path, it is important that the same path is appended to the context path of Bitbucket Server's base URL ([Step 2](#)).
- The context path for serving from the *root* context is `path=""` (i.e **not** `path="/"`).

Step 2: Change Bitbucket Server's base URL

Open a browser window and log into Bitbucket Server using an administrator account. Go to the Bitbucket Server administration area and click **Server settings** (under 'Settings'), and change **Base URL** to match the URL HAProxy will be serving. For this example, use `https://mycompany.com/bitbucket`.

Step 3: Configure the Tomcat Connector

Find the normal (non-SSL) Connector directive in Tomcat's `<Bitbucket Server home directory>/shared/server.xml` file, and add the `secure`, `scheme`, `proxyName`, `proxyPort` and `redirectPort` attributes. These attributes tell Tomcat how **HAProxy** is serving Bitbucket Server so it can generate correct URLs. Following our example:

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="443"
  compression="on"

  compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,application/javascript,application/x-javascript"
  secure="true"
  scheme="https"
  proxyName="mycompany.com"
  proxyPort="443" />
```

- `proxyPort` is set to 443 to indicate that HAProxy is accepting connections over on the standard HTTPS port 443.
- `proxyName` and `scheme` are set to the values that HAProxy is serving Bitbucket Server over.
- `secure` attribute is also set to `true` to tell Bitbucket Server that the connection between the client and HAProxy is considered secure.
- `redirectPort` is set to 443 so that Tomcat knows how to send a user to a secure location when necessary (this is not really necessary in this example because this connector is already secure).

For more information about configuring the Tomcat Connector, refer to the [Apache Tomcat 7.0 HTTP Connector Reference](#).

Step 4: Configure HAProxy

Merge the example below into your HAProxy configuration (e.g `/etc/haproxy/haproxy.cfg`). This is a complete HAProxy 1.5.x configuration. Note that HAProxy 1.5.x or greater is required for SSL support. You can just take the bits that fit your needs. The important configuration is in the `bitbucket_http_frontend` and `bitbucket_http_backend`.

```

global
    log /dev/log local0
    log /dev/log local1 notice
    user haproxy
    group haproxy
    daemon
        ssl-default-bind-options no-ssl3
        maxconn 1000

defaults
    log global
    mode http
    option httplog
    option dontlognull
        timeout connect 5000
        timeout client 50000
        timeout server 50000

# Tells HAProxy to start listening for HTTPS requests. It uses the
# SSL key
# and certificate found within certAndKey.pem. All requests will be
# routed
# to the bitbucket_http_backend.
frontend bitbucket_http_frontend
    bind *:443 ssl crt /etc/haproxy/certAndKey.pem ciphers
    HIGH:!aNULL:!MD5
    default_backend bitbucket_http_backend
    # This is an optional rule that will redirect all requests to
    # https://mycompany.com/bitbucket.
    # to https://mycompany.com/bitbucket.
    redirect location /bitbucket if { path -i / }

# The bitbucket_http_backend simply forwards all requests
# onto http://mycompany.com:7990/.
# It will only allow 50 concurrent connections to the server at once.
backend bitbucket_http_backend
    mode http
    option httplog
    option forwardfor
    option http-server-close
    option httpchk
    server bitbucket01 mycompany.com:7990 maxconn 50

```

(Optional) Step 4: Redirect SSH connections

HAProxy also has the ability to proxy all Bitbucket Server SSH traffic. See [Setting up SSH port forwarding](#) for details.

Resources

Here are some resources you may find helpful in setting up Bitbucket Server behind HAProxy:

- <http://www.haproxy.org/#docs>
- [Setting up SSH port forwarding](#)

Enabling SSH access to Git repositories in Bitbucket Server

A Bitbucket Server administrator can enable SSH access to Git repositories in Bitbucket Server. This allows your Bitbucket Server users to:

- add their own SSH keys to Bitbucket Server
- use those SSH keys to secure Git operations between their computer and the Bitbucket Server instance.

Bitbucket Server users must each [add their own SSH key pairs](#) to their Bitbucket Server account to be able to use SSH access to repositories.

Supported key types are DSA and RSA2. Note that RSA1 is not supported. We've tested key sizes of 768, 1024, 2048, 4096 and 8192 bytes.

On this page:

- [Enabling SSH access](#)
- [SSH base URL](#)
- [When running Bitbucket Server behind a proxy](#)

Related pages:

- [Setting up SSH port forwarding](#)
- [Creating SSH keys](#)

Performance

There are performance implications for Bitbucket Server when using SSH. When users connect to Bitbucket Server using SSH the encryption of data adds to overall CPU usage. See [Scaling Bitbucket Server](#) for more information.

Security

To implement SSH authentication support, Bitbucket Server bundles a version of the [Apache Mina](#) SSH D server. Bitbucket Server's SSH server is not integrated with the SSH server on the host Bitbucket Server is running on nor does it consider the users on the host when authenticating Bitbucket Server users. To prevent security issues, the embedded SSH server has been locked down to allow execution of a small set of commands for Git hosting. The only commands that are supported are `git upload-pack`, `git receive-pack`, `git archive-pack` and `whoami` (a custom `whoami` implemented in Bitbucket Server not the `whoami` command that exists on Linux). It is not possible to open an SSH shell using the embedded server to execute arbitrary commands on the server.

Enabling SSH access

To enable SSH access:

1. Go to the Bitbucket Server administration area and click **Server settings** (under 'Settings').
2. Under 'SSH access', check **SSH enabled**.
3. Enter values for **SSH port** and **SSH base URL**, according the information in the sections below.
4. Click **Save**.

These options will only be available if the "Bitbucket Server - SSH" add-on is enabled. For instructions on how to enable this add-on on your instance, please refer to [Disabling and enabling add-ons](#).

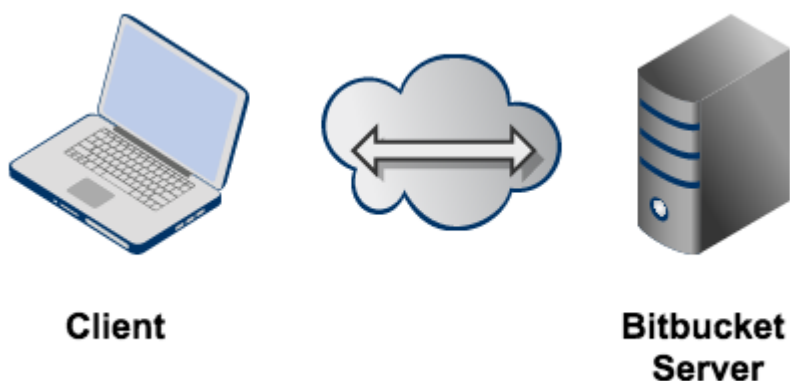
SSH base URL

The **SSH base URL** is the base URL with which users can access the SSH push/pull/clone functionality of Bitbucket Server.

This is the base URL that Bitbucket Server will use when displaying SSH URLs to users. If you do not set this, it will default to the host that is set in **Bitbucket Server base URL**, with the port that SSH is listening on. See [Specifying the base URL for Bitbucket Server](#).

For example, if the **SSH base URL** is not set and the **Bitbucket Server base URL** is `https://bitbucket.atlassian.com` and the SSH port is 7999, the SSH URL for the repository Jira in the project Atlassian will be `ssh://git@bitbucket.atlassian.com:7999/ATLASSIAN/jira.git`

If you set up [port forwarding](#), you will need to set the **SSH base URL** to the machine and port that is being forwarded to Bitbucket Server. However, you do not need to specify the port portion of the URL if the default SSH port (port 22) is being forwarded to Bitbucket Server.



Port forwarding	SSH base URL	Bitbucket Server base URL	SSH port	F
	Not set	<code>https://bitbucket.atlassian.com</code>	7999	s
 Port 22 → 7999	<code>https://bitbucket.atlassian.com</code>	<code>https://bitbucket.atlassian.com</code>	7999	s

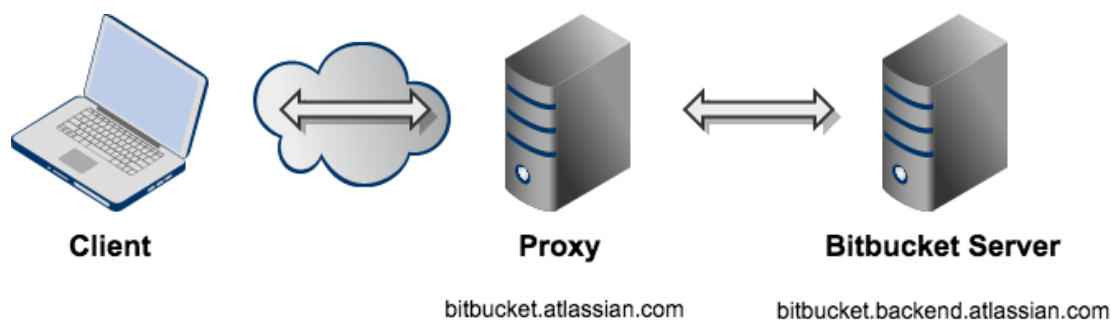
When running Bitbucket Server behind a proxy

If you run Bitbucket Server behind a http proxy such as Apache (e.g. as per our [instructions](#)), and if Apache runs on a different host, SSH will not be available on that host. Instead, you will need to set the SSH base URL to the machine Bitbucket Server is actually running on (and the URL should include the SSH port Bitbucket Server is serving from).

For example, if the **SSH base URL** is set to `ssh://bitbucket.backend.atlassian.com:7999`, the SSH URL for the repository Jira in the project Atlassian will be `ssh://git@bitbucket.backend.atlassian.com:7999/ATLASSIAN/jira.git`

If you set up [port forwarding](#), you will need to set the **SSH base URL** to the proxy machine and port that is being forwarded to Bitbucket Server. However, you do not need to specify the port portion of the URL if the default SSH port (port 22) is being forwarded to Bitbucket Server.

For example, if you set up port forwarding from your http proxy host, `bitbucket.atlassian.com`, port 22, to `bitbucket.backend.atlassian.com` port 7999, set the **SSH base URL** to `ssh://bitbucket.atlassian.com`. Then, the SSH URL for the repository Jira in the project Atlassian will be `ssh://git@bitbucket.atlassian.com/ATLASSIAN/jira.git`



Port forwarding	SSH base URL	SSH port	Bitbucket Server base URL
	ssh://bitbucket.backend.atlassian.com:7999	7999	https://bitbucket.backe
 Port 22→7999	ssh://bitbucket.atlassian.com	7999	https://bitbucket.backe
 Port 44→7999	ssh://bitbucket.atlassian.com:44	7999	https://bitbucket.backe

Setting up SSH port forwarding

Why set up port forwarding?

There are two scenarios where you might want to set up port forwarding.

Remove port numbers from your SSH URLs

Bitbucket Server listens for SSH connections on port 7999 by default.

Your users will need to include the port in the URL they use to clone from Bitbucket Server, for example:

```
git clone ssh://git@bitbucket.mycompany.com:7999/PROJECT/repo.git
```

Rather than have the port number in the URL, you may wish to set up port forwarding so that connections to the default SSH port are forwarded to the port Bitbucket Server is listening on (e.g. you could forward port 22 to port 7999).

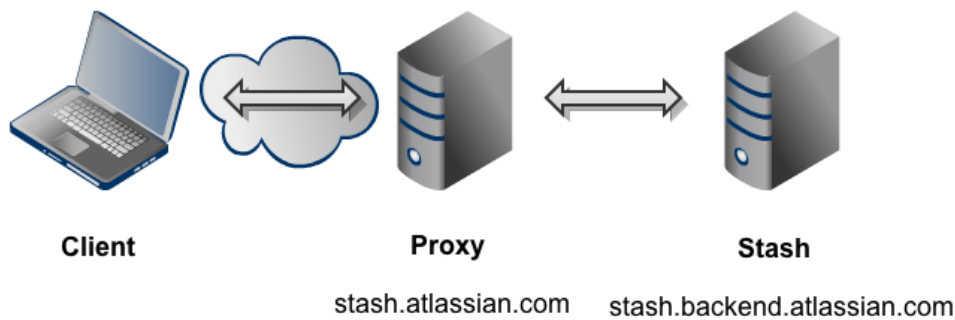
This would allow your users to use a URL without a port number in it, like this:

```
git clone ssh://git@bitbucket.mycompany.com/PROJECT/repo.git
```

Bitbucket Server is running behind a reverse proxy on a separate machine

You may be following our instructions for [setting up Bitbucket Server behind an Apache front-end](#).

In this case, your users may not be able to access Bitbucket Server directly for SSH connections, or if they can, you may wish to make the SSH and HTTPS URLs consistent.



For example, if you have the above topology, without port forwarding (and assuming the default port of 7999), your users will need to clone Bitbucket Server directly from the backend, like this:

```
git clone
ssh://git@bitbucket.backend.atlassian.com:7999/PROJECT/repo.git
```

In your network, the `bitbucket.backend.atlassian.com` machine may not be accessible directly, or you may want the URL to be consistent with the HTTPS URL of `https://bitbucket.atlassian.com/scm/PROJECT/repo.git`.

In this case, you need to set up port forwarding on the `bitbucket.atlassian.com` machine to accept connections and forward them to port 7999 on the `bitbucket.backend.atlassian.com` machine.

How to set up port forwarding

HAProxy

Atlassian recommends the use of [HAProxy](#) for forwarding SSH connections through to Bitbucket Server.

HAProxy is [supported](#) on Linux, Solaris and FreeBSD.

HAProxy is not an Atlassian product, so Atlassian does not guarantee to provide support for its configuration. This section is provided for your information only – use it at your own risk. We recommend that you refer to the [HAProxy documentation](#).

Installing HAProxy

Your operating system may support installing HAProxy using its system package manager, such as `apt-get`, `yum` or `rpm`. This will be the easiest way.

Alternatively, you may build HAProxy yourself and install it.

1. Download the latest version of HAProxy from <http://haproxy.1wt.eu/#down>.
2. Extract the archive and `cd` into the directory:

```
tar xzvf haproxy-1.4.21.tar.gz
cd haproxy-1.4.21
```

3. Read the instructions in the README for how to build on your system. This is generally quite simple - on a Linux 64 bit 2.6 Kernel, the command is:

```
make TARGET=linux26 ARCH=x86_64
```

4. If it completes successfully, install it following the instructions in the README:


```
sudo make install
```

Configuring HAProxy

HAProxy is extremely powerful - it is designed as a HTTPS load balancer, but also can serve as a port forwarder for ssh.

The full documentation for version 1.4 is [here](#). More documentation is available on the [HAProxy web site](#).

An example simple configuration is as follows:

```
global
    daemon
    maxconn 10000

defaults
    timeout connect 500s
    timeout client 5000s
    timeout server 1h

frontend sshd
    bind *:7999
    default_backend ssh
    timeout client 1h

backend ssh
    mode tcp
    server localhost-bitbucket-ssh 127.0.0.1:7999 check port 7999
```

The above configuration will listen on port 7999 (indicated by the [bind](#) directive) on all network interfaces. As indicated by the [server](#) directive, traffic is forwarded to 127.0.0.1, port 7999. You will need to replace 127.0.0.1 with the IP address of the machine running Bitbucket Server.

You can check your configuration by running:

```
haproxy -f haproxyconf.txt -c
```

To run haproxy, simply start it using

```
haproxy -f haproxyconf.txt
```

If you use HAProxy to additionally proxy HTTP traffic, ensure that the running mode configuration is set to http:

```
backend http
    mode http
    bind *:80
    server localhost-bitbucket-http 127.0.0.1:7990
```

Using the default SSH port

You can configure HAProxy to listen on the default SSH port instead, so that the port does not need to be specified in the clone URL.

By default, the normal ssh daemon is running on port 22. You have several options:

- Configure HAProxy to listen on an alternate port as in the previous example.
- Configure multiple network interfaces on the physical machine and force the default ssh daemon to listen on all but the interface for accessing Bitbucket Server. Configure HAProxy to only listen on that interface.
- Move the default ssh daemon to listen on another port and let HAProxy bind on port 22.

We do not provide instructions on the last two options, except for how to configure HAProxy.

Use the same configuration as the last example, but change the bind port to 22, e.g.

```
...
frontend sshd
    bind *:22
...
```

You will have to run this configuration as the `root` user, using `sudo`, because it specifies a port to listen on that is less than 1024.

```
sudo haproxy -f haproxyconf.txt
```

Configuring the SSH base URL

Once port forwarding is set up, you will need to configure the SSH base URL in Bitbucket Server so that the clone urls presented in Bitbucket Server indicate the correct host and port to clone from. See the [SSH base URL](#) section in [Enabling SSH access to Git repositories in Bitbucket Server](#).

Using diff transcoding in Bitbucket Server

As of Bitbucket Server 3.1, Bitbucket Server supports transcoding for diffs. This allows Bitbucket Server to convert files in encodings like EUC-JP, GB18030 and UTF-16 to UTF-8, so they are processed correctly by `git diff`, which only supports UTF-8. Similar transcoding has been applied to Bitbucket Server's source view since it was released, so this change brings the diff view in line with the source view. Diff transcoding is applied to commit and pull request diffs, as well as the diff-to-previous view.

Git for Windows, formerly known as msysgit, has known issues with Unicode paths. Diff transcoding works on all supported versions of Git for Windows, but 1.8.0 or higher is required to support Unicode paths.

Enabling diff transcoding

Diff transcoding must be explicitly enabled for each repository (unlike source view transcoding, which is always performed).

Repository administrators can enable diff transcoding on the repository settings page:

Repository details

Move repository
Delete repository

Name

Changing this repository's name will change its clone URL. You can update a remote with the following command: `git remote set-url REMOTE_NAME NEW_URL`
<http://localhost:7990/scm/tis/apollo-ui.git>

Location on disk
/opt/atlassian/data/

Approximate size
[Retrieve size details](#)

Default branch

master

Select the default branch for browsing and cloning the repository. Git chooses the "master" branch by default.

☒ Allow forks
Clear the checkbox to prevent forking of this repository

☐ Transcode diffs
Transcode diffs, allowing non-UTF-8 diffs to be displayed. [Learn more](#)

Save
Cancel

Performance and scaling

There's a performance consideration with transcoding. It is implemented using Git's `textconv` support, so using it adds overhead to displaying diffs. Where possible, the best approach, given `git` only supports UTF-8 content, is to use UTF-8 encoding so that transcoding is not necessary. In repositories without non-UTF-8 content, diff transcoding should be left disabled. Other encodings are often a necessity, however, and for repositories containing such content enabling diff transcoding allows using the full range of Bitbucket Server features.

▼ [Click here to read more...](#)

When transcoding is enabled, `git diff` writes the before and after blobs to temporary files and invokes the `textconv` script once for each file. The script Bitbucket Server installs uses Perl to send a request back to Bitbucket Server with the path to each temporary file. Bitbucket Server then opens each file, detects the encoding using the same algorithm the source view uses, converts the file to UTF-8 and streams it out for `git diff` to use. After `git diff` has invoked the `textconv` script the temporary files it created are deleted.

Writing the blobs to disk, starting Perl and calling back into Bitbucket Server are all overhead processing compared to performing a diff without transcoding. How much overhead that is varies by the size of the diff. When nominally-sized files containing two or three thousand lines or less are being compared the overhead is miniscule, under 50 milliseconds on an average server. However, when comparing larger files the overhead can result in a noticeable delay displaying the diff.

Changing the port that Bitbucket Server listens on

You may wish to change the port that Bitbucket Server listens on from the default '7990' to a different value if another application is already running on that port.

To change the port, edit the `shared/server.xml` file in the `<Bitbucket home directory>`.

Find the following lines in `server.xml`:

```
<Server port="8006" shutdown="SHUTDOWN">
...
<Connector port="7990" protocol="HTTP/1.1"
    connectionTimeout="20000"
    useBodyEncodingForURI="true"
    redirectPort="8443"
    compression="on"

    compressableMimeType="text/html,text/xml,text/plain,text/css,application
    /json"/>
```

You need to modify both the server port (the default is 8006) and the connector port (the default is 7990) to ports that are free on your machine. The server port is required by Tomcat but is not user-facing in any way. The connector port is the port you use to access Bitbucket Server. For example, in the snippet above, the URL would be <http://example.com:7990>.

✔ Hint: You can use netstat to identify free ports on your machine. See more information on using netstat on [Windows](#) or on [Linux](#).

⚠ If you are using a firewall, you should ensure that it is configured to allow HTTP or HTTPS traffic over the connector port you have chosen.

⚠ If you are running Bitbucket Server on a Linux server and want to bind to privileged ports (those below 1024, for example port 80), you will **need to start Bitbucket Server as root** in order to successfully bind to the port. Alternatively, you can bind Bitbucket Server to a port over 1024 and then configure iptables to redirect traffic from port 80 to the higher port.

Related pages:

- [Specifying the base URL for Bitbucket Server](#)
- [Proxying and securing Bitbucket Server](#)

Moving Bitbucket Server to a different context path

There are various reasons why you may wish to change the context path for Bitbucket Server. Two of those are:

- You are running Bitbucket Server behind a proxy.
- You have another Atlassian application, or Java web application, available at the same hostname and context path as Bitbucket Server, and are experiencing login problems (see [Login and session conflicts with multiple Atlassian applications](#)).

Related pages:

- [Integrating Bitbucket Server with Apache HTTP Server](#)
- [Login and session conflicts with multiple Atlassian applications](#)

Upgrade Note

Note that the location of `server.xml` changed in Bitbucket Server 3.8. See the [Bitbucket Server upgrade guide](#).

Changing the context path for Bitbucket Server:

1. Navigate to your [Bitbucket Server home directory](#).
2. Stop Bitbucket Server. See [Starting and stopping Bitbucket Server](#).
3. Edit `<Bitbucket home directory>/shared/server.xml` and find the element below:

```
<Context path="" docBase="${catalina.home}/atlassian-bitbucket"
reloadable="false" useHttpOnly="true"/>
```

Update the `path` attribute to reflect the context path that you want Bitbucket Server to be accessible at, e.g. `"/bitbucket"`:

```
<Context path="/bitbucket"
docBase="${catalina.home}/atlassian-bitbucket" reloadable="false"
useHttpOnly="true"/>
```

Then save the file.

4. Start Bitbucket Server. See [Starting and stopping Bitbucket Server](#).

Bitbucket Server should now be available at the same host as before under the new context path. For example a server that was at <http://localhost:7990> will now be reachable at <http://localhost:7990/bitbucket>.

5. Once Bitbucket Server has started, go to the administration area and click **Server settings** (under 'Settings'). Append the new context path to your base URL:

```
https://my-bitbucket-hostname:7990/bitbucket
```

6. Click **Save**.

Bitbucket Server + Apache

Note that if you are running Bitbucket Server behind Apache:

- You will need to make sure that the host or context path that Bitbucket Server is exposed on is not also being used by another web application that is listening on a different port.
- If you have updated the Bitbucket Server context path using the steps outlined above, you will need to update your Apache configuration, as described in [Integrating Bitbucket Server with Apache HTTP Server](#).

Application Links

If you had Application Links set up before changing the context path in Bitbucket Server, you will have to recreate those using the new Bitbucket Server URL. See [Linking Bitbucket Server with JIRA](#).

SSH

The context path does not affect the URL at which SSH operations occur. After changing the context path so that Bitbucket Server is accessible at `https://my-bitbucket-hostname:7990/bitbucket`, SSH operations occur without the context path at `ssh://my-bitbucket-hostname:7999`.

Running Bitbucket Server with a dedicated user

For production installations, we recommend that you create a new dedicated user that will run Bitbucket Server on your system. This user:

- Should be local.
- Should *not* have admin privileges.
- Should be a non-privileged user with read, write and execute access (called "Full control" permission on Windows) on the Bitbucket Server install directory and [home directory](#).

Note that, on Windows, running Bitbucket Server (whether as a service, or not) as a user that is part of the Administrator group can cause Windows to spend a lot of time running permission checks, with a consequent [performance impairment for Git operations](#).

See also [Running Bitbucket Server as a Windows service](#) and [Running Bitbucket Server as a Linux service](#).

For **Linux**, here is an example of how to create a dedicated user to run Bitbucket Server:

```
$ sudo /usr/sbin/useradd --create-home --home-dir  
/opt/atlassian/bitbucket --shell /bin/bash atlbitbucket
```

Bitbucket Server debug logging

On this page:

- [Debug logging for the Bitbucket Server instance](#)
 - [Enabling debug logging via the UI](#)
 - [Enabling debug logging on startup](#)
 - [Enabling debug logging at runtime](#)
- [Profiling logging for the Bitbucket Server instance](#)

- [Enabling profiling logging via the UI](#)
- [Debug logging for Git operations on the client](#)
 - [On Linux](#)
 - [On Windows](#)
- [Debug logging for the Bitbucket Server Backup Client](#)

Debug logging for the Bitbucket Server instance

This section describes how to enable debug level logging in Bitbucket Server. Bitbucket Server logs can be found in `<Bitbucket home directory>/log`.

When using the standard Bitbucket Server distribution, logs for the Tomcat webserver that hosts Bitbucket Server can be found in `<Bitbucket Server installation directory>/log`.

Enabling debug logging via the UI

To enable debug logging, go to the Bitbucket Server admin area, choose **Logging and Profiling** (under 'Support') and select **Enable debug logging**.

Enabling debug logging on startup

To enable debug logging whenever Bitbucket Server is started, edit the `<Bitbucket home directory>/shared/bitbucket.properties` file (if this file doesn't exist then you should create it) and add the following two lines:

```
logging.logger.ROOT=DEBUG
logging.logger.com.atlassian.bitbucket=DEBUG
```

If your Bitbucket Server instance is earlier than version 3.2, the `bitbucket.properties` file is at the top level of the [Bitbucket Server home directory](#).

Enabling debug logging at runtime

To enable debug logging for the root logger once Bitbucket Server has been started, run the following two commands in your terminal:

```
curl -u <ADMIN_USERNAME> -v -X PUT -d "" -H "Content-Type:
application/json" <BASE_URL>/rest/api/latest/logs/rootLogger/debug
curl -u <ADMIN_USERNAME> -v -X PUT -d "" -H "Content-Type:
application/json"
<BASE_URL>/rest/api/latest/logs/logger/com.atlassian.bitbucket/debug

# e.g.
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json"
http://localhost:7990/rest/api/latest/logs/rootLogger/debug
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json"
http://localhost:7990/rest/api/latest/logs/logger/com.atlassian.bitbucket/debug
```

To enable debug logging for a specific logger, run the following command in your terminal:

```
curl -u <ADMIN_USERNAME> -v -X PUT -d "" -H "Content-Type:
application/json"
<BASE_URL>/rest/api/latest/logs/logger/<LOGGER_NAME>/debug

# e.g.
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json"
http://localhost:7990/rest/api/latest/logs/logger/com.atlassian.crowd/de
bug
```

Profiling logging for the Bitbucket Server instance

This section describes how to enable profiling in Bitbucket Server. This log is essential when troubleshooting performance issues. Bitbucket Server logs can be found in <Bitbucket home directory>/log.

When using the standard Bitbucket Server distribution, logs for the Tomcat webserver that hosts Bitbucket Server can be found in <Bitbucket Server installation directory>/log.

Enabling profiling logging via the UI

To turn on detailed trace information, go to the Bitbucket Server admin area, choose **Logging and Profiling** (under 'Support') and select **Enable profiling**.

Debug logging for Git operations on the client

Atlassian Support might request DEBUG logs for Git operations (on the client) when troubleshooting issues. You can enable DEBUG logging on the Git client by setting the following variables. If you are using HTTP/S please do remove the Authorization header from the output as it will contain your Basic-Auth information. Atlassian provides a set of [scripts](#) that simplify the collection of git client debug information.

On Linux

Execute the following on the command line before executing the Git command:

```
export GIT_TRACE_PACKET=1
export GIT_TRACE=1
export GIT_CURL_VERBOSE=1
```

To measure the length of time a command takes to complete, use the `time` utility/command prior to the actual git command. For example, to test a push to branch "master":

```
time git push origin master
```

On Windows

Execute the following on the command line before executing the Git command:

```
set GIT_TRACE_PACKET=1
set GIT_TRACE=1
set GIT_CURL_VERBOSE=1
```


Setting `GIT_CURL_VERBOSE` is only useful for connections over HTTP/S since SSH doesn't use the `libcurl` library.

Debug logging for the Bitbucket Server Backup Client

Atlassian Support might request `DEBUG` logs for the Backup client when troubleshooting issues.

You can enable `DEBUG` logging on the Backup client by adding a file named `logback.xml` to your working directory (`pwd`) with the following content:

logback.xml

```
<included><logger name="com.atlassian.bitbucket"
level="DEBUG"/></included>
```

Data recovery and backups

This page provides an overview of the backup and restore strategies that Atlassian recommends for use with Bitbucket Server:

- [Bitbucket Server backup essentials](#)
- [Two ways to back up Bitbucket Server](#)
- [How Bitbucket Server backup and restore works](#)

Related pages:

- [Connecting Bitbucket Server to an external database](#)
- [Supported platforms](#)

Questions? Check out [FAQ - Data recovery and backup](#).

Bitbucket Server backup essentials

An effective backup strategy is essential:

- for avoiding data loss in the event of any system breakdown
- for restoring Bitbucket Server after any system breakdown
- as part of the Bitbucket Server upgrade process.

We highly recommend that you establish a data recovery plan that is aligned with your company's policies. At the very least, you should consider these aspects:

- How frequently should Bitbucket Server be backed up? We recommend that backups are made daily.
- How much downtime is acceptable?
- How long should backups be stored for? We recommend that backups be kept for at least one month.
- Where should the backups be stored? We recommend that backups are stored offsite.

With any strategy, you should schedule the backup window so as to minimize the impact on Bitbucket Server availability. You might consider checking the access logs to determine patterns of lowest usage to help with this.

Two ways to back up Bitbucket Server

When it comes to backing up, every organization has slightly different policies and requirements. Some organizations will want a backup solution that just works with minimal intervention, is independent of the underlying database and file system configuration, and don't mind a little downtime when the backup is run as part of a nightly maintenance schedule. Other organizations will have more specific policies and requirements surrounding the use of vendor-specific database and storage backup tools, the maximum acceptable downtime, the format of backups, and where the backups are ultimately stored.

To cater for these different policies and requirements, Bitbucket Server provides two different backup strategies:

- the Bitbucket Server Backup Client,
- Bitbucket Server DIY Backup.

The features of each backup strategy are summarized in the following table:

	Bitbucket Server Backup Client	Bitbucket Server DIY Backup
Audience	Recommended for people who don't want to have to script the backup process at the database and file system level.	Recommended for developers and system administrators who want to customize the Bitbucket Server backup process for their specific database and file system configuration.
Usage	Ready to use out of the box.	Requires you to write some code (in your preferred language) to perform the backup steps.
Downtime	Locks Bitbucket Server for the entire duration of the backup.	Only locks Bitbucket Server for the minimum time necessary.
Backup	Backup files are in a vendor-independent format and do not depend on the underlying database configuration.	Backup files rely on vendor-specific database and/or storage tools; for example, <code>pg_dump</code> is used if your back end database is PostgreSQL.
Destination	Backups are saved on the local filesystem.	Backups can be saved anywhere.
Supports Bitbucket Server	✔ Yes	✔ Yes
Supports Bitbucket Data Center	✘ Not supported with two or more cluster nodes running.	✔ Yes
Documentation	Using the Bitbucket Server Backup Client	Using Bitbucket Server DIY Backup

The Bitbucket Server Backup Client does not support clustered Bitbucket Data Center instances, even if you switch to a single node. In order to back up a clustered instance of Bitbucket Data Center, you must switch to Bitbucket Server DIY Backup.

How Bitbucket Server backup and restore works

Whether you choose Bitbucket Server Backup Client or Bitbucket Server DIY Backup, it is important to understand that there is a tight coupling between the Bitbucket Server file system on disk and the database that Bitbucket Server uses. The following types of data are stored in the backup process:

- The Bitbucket Server **home directory** on the file system, containing your repository data, cache and log files (see [Bitbucket Server home directory](#) for more detail).
- The Bitbucket Server **database**, containing data about pull requests (pointers to branches in the repos, comments and pull request diffs) and user management.

The backup process must ensure that you keep the repository data and the database perfectly synchronized, by:

- Shutting down Bitbucket Server before performing the backup.
- Restoring the database and file system at the same time.
- Using the same version or snapshot of the database and file system.

Any backup strategy that captures both the file system and database while Bitbucket Server is still available to users would run the risk that the backed up Git repositories might be corrupted or that the data in the

database doesn't reflect the repository state on disk. Therefore, strategies for backing up and restoring Bitbucket Server data must keep the repository data and the database perfectly synchronized.

Backing up Bitbucket Server when using the internal database

When Bitbucket Server uses the built-in HSQL database, the database files are stored in the Bitbucket Server file system. See [Bitbucket Server home directory](#) for more detail.

Making a backup of Bitbucket Server involves copying the Bitbucket home directory.

Note that Atlassian does not recommend using the internal database for a production instance.

Backing up Bitbucket Server when using an external database

When Bitbucket Server uses an external database, both the [Bitbucket Server home directory](#) and the external database must be backed up.

If you use the Bitbucket Server Backup Client, the external database is automatically included as part of the backup in a vendor-independent format.

If you use Bitbucket Server DIY Backup, you have full control over the database backup tools and procedures, and can use your database vendor's specific backup tooling that is optimized for your database back end and stores the dump in a vendor-specific format.

Restoring Bitbucket Server from a cold backup

Whether you use the Bitbucket Server Backup Client or Bitbucket Server DIY Backup, recovering a Bitbucket Server instance from backup requires restoring both:

1. the file system backup, and
2. the database backup.

If you use the Bitbucket Server Backup Client, these components are both automatically included when you run the Bitbucket Server Restore Client.

If you use Bitbucket Server DIY Backup, the backup script restores the file system backup, and you need to use your database vendor's specific restore tooling to restore the database backup.

Using the Bitbucket Server Backup Client

This page describes using the Bitbucket Server Backup Client. This tool can be used to backup data from Bitbucket Server instances from release 2.7.0 and later.

The Bitbucket Server Backup Client is not compatible with clustered Bitbucket Data Center instances (even if you switch back to a single node). To back up a Bitbucket Data Center instance, you must switch to [Bitbucket Server DIY Backup](#) instead of the Bitbucket Server Backup Client.

For information about other backup strategies for Bitbucket Server, see [Data recovery and backups](#).

With any strategy, you should consider scheduling the backup window so as to minimize the impact on Bitbucket Server availability. You might consider checking the access logs to determine patterns of lowest usage to help with this.

On this page:

- [How it works](#)
- [What is backed up](#)
- [Backing up Bitbucket Server using the client](#)
- [Canceling the client backup](#)
- [Restoring Bitbucket Server to use the existing DB](#)
- [Restoring Bitbucket Server to use a newly created DB](#)
- [Debug logging](#)

Related pages:

- [Data recovery and backups](#)
- [Using Bitbucket Server DIY Backup](#)
- [Scheduling tasks on Linux](#)
- [Scheduling tasks on Windows](#)
- [Debug logging for the Bitbucket Server Backup Client](#)
- [Bitbucket Server - FAQ - Data recovery and backups](#)
- [How to establish staging server environments for Bitbucket Server](#)

Download the Bitbucket Server Backup Client:

[Download](#)

We highly recommend that you establish a data recovery plan that is aligned with your company's policies.

Questions? Check out [FAQ - Data recovery and backup](#).

Unzip the client into a directory on the Bitbucket Server server.

How it works

The Backup Client implements a common and universal way to back up a Bitbucket Server instance, and does the following:

1. Locks access to the Bitbucket Server application, the repositories managed by Bitbucket Server and the Bitbucket Server database for the entire duration of the back up. This state is called 'maintenance mode'.
2. Checks that all Git and database operations have completed.
3. Performs an application-specific backup of the [Bitbucket Server home directory](#) and the Bitbucket Server database. The backup is generic and does not depend on the server or database configuration.
4. Stores the backup as a single tar file on the local filesystem in the specified location.
5. Unlocks Bitbucket Server from maintenance mode.

You will get an error message if you try to access the Bitbucket Server web interface, or use the Bitbucket Server hosting services, when Bitbucket Server is in maintenance mode.

The client supports Windows and Linux platforms, and Bitbucket Server versions 4.0 and higher, but does not provide ways to integrate with your organizations IT policies or processes.

As an indication of the unavailability time that can be expected when using the Bitbucket Server Backup Client, in our testing and internal use we have seen downtimes for Bitbucket Server of 7–8 minutes with repositories totalling 6 GB in size. For comparison, using [Bitbucket Server DIY Backup](#) for the same repositories typically results in a downtime of less than a minute.

What is backed up

The Backup Client backs up all the following data:

- the database Bitbucket Server is connected to (either the internal or external DB)
- managed Git repositories
- the Bitbucket Server audit logs
- installed plugins and their data

The backup does NOT include the following files and directories:

- `export/*`
- `log/*` (except for the audit logs)
- `shared/data/db*` (HSQL data in the DB is backed up, but the files on disk are not)
- `tmp`
- the plugins directory (except for the `installed-plugins` directory)

Backing up Bitbucket Server using the client

Before you begin backing up Bitbucket Server using the client, you should first ensure you are using a [release of the Bitbucket Server Backup Client that is compatible with your Bitbucket Server instance](#).

The Backup Client must be run from somewhere with access to the Bitbucket Server home directory. Usually, you will run the Backup Client directly on the Bitbucket Server server. Run the client with the following commands:

```
cd <path/to/backup-config.properties file>
java -jar <path/to/bitbucket-backup-client.jar>
```

Configuration options are kept in the `backup-config.properties` file, an example of which is included with the client. This file is automatically read from the directory you were in when the `bitbucket-backup-client` is run. The properties are fully documented in the `backup-config.properties` file, but include:

<code>bitbucket.home</code>	Defines the location of the home directory of the Bitbucket Server instance you wish to back up or restore to. REQUIRED If omitted here it will be taken from the <code>BITBUCKET_HOME</code> environment variable or the Java system property of the same name if supplied to the Backup and Restore Client on the command line. As a required value, backup and restore will fail if it is not supplied through one of these mechanisms.
<code>bitbucket.user</code>	Defines the username of the Bitbucket Server user with administrative privileges you wish to perform the backup. REQUIRED If omitted here it will be taken from the Java system property of the same name if supplied to the Backup Client on the command line. As a required value, backup will fail if it is not supplied through one of these mechanisms.
<code>bitbucket.password</code>	Defines the password of the Bitbucket Server user with administrative privileges you wish to perform the backup. REQUIRED If omitted here it will be taken from the Java system property of the same name if supplied to the Backup Client on the command line. As a required value, backup will fail if it is not supplied through one of these mechanisms.
<code>bitbucket.baseUrl</code>	Defines base URL of the Bitbucket Server instance you wish to back up. REQUIRED E.g. <code>http://localhost:7990/bitbucket</code> or <code>http://bitbucketserver/</code>. If omitted here it will be taken from the Java system property of the same name if supplied on the command line to the Backup Client. As a required value, backup will fail if it is not supplied through one of these mechanisms.
<code>backup.home</code>	Defines where the Backup Client will store its own files, such as backup archives. If not specified, these files are stored beneath the working directory for the Backup Client. Backup files will be stored in a <code>backup</code> subdirectory and logs will be stored in a <code>logs</code> subdirectory. Note that on Windows, you must use two backslashes between paths. E.g. <code>C:\path\\to\\folder</code> or instead use the forward slash e.g. <code>C:/path/to/folder</code>. The location defined by <code>backup.home</code> must not be located in the directory defined by <code>bitbucket.home</code>. If that is the case, the Backup Client will fail.

Alternatively, these properties can be given on the command-line, when they need to be prefixed with "-D", and be placed before the "-jar" parameter. For example:

```
java -Dbitbucket.password="admin" -Dbitbucket.user="admin"  
-Dbitbucket.baseUrl="http://localhost:7990"  
-Dbitbucket.home=path/to/bitbucket/home  
-Dbackup.home=path/to/backup-home -jar bitbucket-backup-client.jar
```

Canceling the client backup

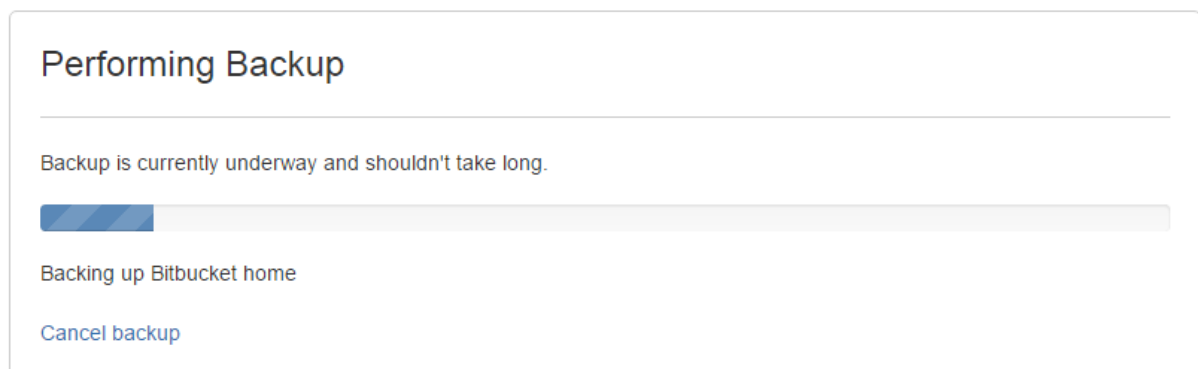
You can cancel the running client backup operation if necessary.

To cancel the backup:

1. Copy the cancel token echoed by the client in the terminal (or the Command Prompt on Windows):

```
$ ./bitbucket.diy-backup.sh  
[http://localhost:7990/bitbucket] INFO: Prepared backup of DB  
bitbucket in /bitbucket-backup/bitbucket-db/  
building file list ... done.  
sent 4.17M bytes received 484 bytes 2.78M bytes/sec  
total size is 121.12M speedup is 29.06  
[http://localhost:7990/bitbucket] INFO: Prepared backup of  
/bitbucket-home to /bitbucket-backup/bitbucket-home/  
[http://localhost:7990/bitbucket] INFO: locked with  
'7187ae1824ce1ede38a8e7de4bccf58d9a8e1a7a'  
[http://localhost:7990/bitbucket] INFO: backup started with  
'82c73f89e790b27fef3032e81c7071388ae4e371'  
[http://localhost:7990/bitbucket] INFO: Waiting for DRAINED  
state..... done  
[http://localhost:7990/bitbucket] INFO: db state 'DRAINED'  
[http://localhost:7990/bitbucket] INFO: scm state 'DRAINED'
```

2. Go to the Bitbucket Server interface in your browser. Bitbucket Server will display this screen:



3. Click **Cancel backup**, and enter the cancel token:

Performing Backup

Backup is currently underway and shouldn't take long.

Backing up Bitbucket home

Authentication token

82c73f89e790b27fef3032e81c7071388ae4e371

To cancel, please enter an authentication token. This can be found in the Bitbucket logs

Cancel backup

4. Click **Cancel backup**.

Restoring Bitbucket Server to use the existing DB

This section applies if you are restoring Bitbucket Server to fix a corrupted installation, but are able to use the existing DB that Bitbucket Server was backed up from. This scenario assumes that Bitbucket Server is to be restored to the same server from which Bitbucket Server was originally backed up.

The Restore Client must be run on the machine that Bitbucket Server should be restored to. To ensure restores do not accidentally delete existing data, the Restore Client will only restore into an empty home directory and an empty database.

The Restore Client will use the JDBC connection configuration contained in the backup you are restoring from.

Follow this process:

1. Stop your Bitbucket Server instance.
2. Delete the content of the current home directory, so that it is empty.
3. Drop the existing tables in your database so it is empty. The database still needs to exist with the same user/password, and it should have the configuration described in the 'Create the Bitbucket Server database' section of the relevant page here:
 - [MySQL](#)
 - [Oracle](#)
 - [PostgreSQL](#)
 - [SQL Server](#)
4. Run the Restore Client using the following command (replacing 'path/to/bitbucket/home' and '/path/to/original/backup/file' with your own values):

```
java -Dbitbucket.home="path/to/bitbucket/home" -jar  
bitbucket-restore-client.jar /path/to/original/backup/file
```

5. If you are restoring Bitbucket Server to fix a corrupted installation, now follow Steps 4 to 6 of the [Bitbucket Server upgrade guide](#). Note that you should use the same version of Bitbucket Server that was used to back up Bitbucket Server.

Restoring Bitbucket Server to use a newly created DB

This section applies if you intend to perform a restore into a newly created DB. This scenario assumes the restore is done to a server different from the one from which Bitbucket Server was originally backed up.

The restore process is database agnostic, meaning the database you are restoring your backup to could be

of a different configuration or type from the originally backed up database.

When restoring Bitbucket Server, the Restore Client must be run on the machine that Bitbucket Server should be restored to. To ensure restores do not accidentally delete existing data, the Restore Client will only restore into an empty home directory and an empty database.

Follow this process:

1. Create a new empty home directory using the user account that will be used to run Bitbucket Server.
2. Create the database. It should have the configuration described in the 'Create the Bitbucket Server database' section of the relevant page here:
 - [MySQL](#)
 - [Oracle](#)
 - [PostgreSQL](#)
 - [SQL Server](#)
3. Run the Restore Client. [See the following section](#) for details.
4. Install Bitbucket server on the new server by following the instructions on [Running the Bitbucket Server installer](#). Point the installation to an empty directory and install it as a service in case you used to have that in your previous server.
5. Make sure your new installation is up and running - refer to [Starting and stopping Bitbucket Server](#). At this stage you should see Bitbucket server guiding you for a configuration from scratch (i.e. it will ask you for license details, admin user etc.) and that's expected.
6. Stop the newly installed instance.
7. Delete the contents from the *newly installed* BITBUCKET_HOME.
8. Copy the contents of the *restored* BITBUCKET_HOME into the *newly installed* BITBUCKET_HOME.
9. Start the *newly installed* Bitbucket server.
10. At this stage, the following will happen:
 - a. If you installed a binary at the same release as it was on your previous server, everything will be like before and you will need to perform an upgrade in a separate step. This is the recommended approach.
 - b. If you installed a binary at a release above the one it was installed on your previous server, an upgrade will be performed at this stage.

Running the Restore Client from the command line

You can run the Restore Client from the command line. In this scenario, as you will have created a new database, you need to specify the JDBC connection parameters that should be used.

The Restore Client uses the JDBC connection configuration specified in the `jdbc.driver`, `jdbc.url`, `jdbc.user` and `jdbc.password` parameters used in the command to run the Restore Client (see below). Once the database backup is successfully restored, the client will write the specified parameters to the `bitbucket-config.properties` file in the newly restored Bitbucket Server home directory, allowing the new instance to connect to the restored database once the steps outlined below are followed.

Before performing a restore to a MySQL database, note that the JDBC driver for MySQL is *not* bundled with it (due to licensing restrictions). You need to download and install the driver yourself.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).
2. Expand the downloaded zip/tar.gz file.
3. Copy the `mysql-connector-java-5.1.XX-bin.jar` file from the extracted directory to your `<path/to/backup/client>/jdbc` directory.

In this example, you can follow the restore into a newly created PostgreSQL database:

```
java -Djdbc.override=true -Djdbc.driver=org.postgresql.Driver
-Djdbc.url=jdbc:postgresql://HOSTNAME:PORT/DATABASE
-Djdbc.user=bitbucketuser -Djdbc.password=password
-Dbitbucket.home="path/to/bitbucket/home" -jar
/path/to/bitbucket-restore-client.jar /path/to/original/backup/file
```

Alternatively, you can configure these parameters on the `backup-config.properties` file – make sure the file exists in the current working directory. A sample file is shipped with the client. The properties are fully

documented in the `backup-config.properties` file and more details are described below:

<code>bitbucket.home</code>	The full path to a directory that the Restore Client will populate with the Bitbucket Server home data. This directory must be empty. On Windows, you must use two backslashes (<code>\\</code>) or a single forward slash (<code>/</code>) to separate paths.
<code>jdbc.override</code>	By default, the Restore Client will restore into the same database that was backed up. If <code>jdbc.override</code> is set to <code>true</code> , the Restore Client will restore into the database specified by the <code>jdbc</code> properties in the table below. The database must be empty.
<code>jdbc.driver</code>	The driver class that Bitbucket Server should use to log in to the new database. See examples below.
<code>jdbc.url</code>	The connection details for the new database, formatted as a JDBC URL. See examples below.
<code>jdbc.user</code>	The username that Bitbucket Server should use to log in to the new database.
<code>jdbc.password</code>	The password that Bitbucket Server should use to log in to the new database.
<code>bitbucket.home.restore.whitelist</code>	Defines a comma-separated list of files and directories that may be present in the target home <i>and</i> shared directories when restoring a backup. Files other than those matching these entries will result in a failure. By default files <code>.snapshot</code> , <code>lost+found</code> , <code>.DS_Store</code> are white listed.

Example use of JDBC properties

Example `jdbc.driver` and `jdbc.url` properties are shown below:

Database	<code>jdbc.driver</code>	<code>jdbc.url</code>
MySQL	<code>com.mysql.jdbc.Driver</code>	<code>jdbc:mysql://HOSTNAME:PORT/DATABASE?</code>
Oracle	<code>oracle.jdbc.driver.OracleDriver</code>	<code>jdbc:oracle:thin:@//HOSTNAME:PORT/SERVICE</code>
PostgreSQL	<code>org.postgresql.Driver</code>	<code>jdbc:postgresql://HOSTNAME:PORT/DATABASE</code>
SQL Server	<code>com.microsoft.sqlserver.jdbc.SQLServerDriver</code>	<code>jdbc:sqlserver://HOSTNAME:PORT;database=</code>

Debug logging

Debug logging can be turned on by adding the following to the `logback.xml` file in the working directory from where you're running the backup client. Create this file if it does not already exist.

<code>logback.xml</code>
<pre><included><logger name="com.atlassian.bitbucket" level="DEBUG"/></included></pre>

Using Bitbucket Server DIY Backup

This article explains use of the Bitbucket Server DIY Backup scripts for use with Bitbucket Server 4.x+. If you are running an earlier version of this product, formerly known as Stash, please see [Using Stash \(3.11\) DIY Backup](#).

The Bitbucket Server DIY Backup is an alternative strategy to using the [Bitbucket Server Backup Client](#). It allows you to:

- significantly reduce the downtime needed to create a consistent backup,
- use the vendor-specific database backup tool appropriate to your back end database, for example:
 - `pg_dump` if your back end database is PostgreSQL, or
 - `sqlcmd` with an appropriate command for differential backup, if your back end database is MS SQL Server,
- use the optimal file system backup tool for your Bitbucket Server home directory, for example:
 - an LVM snapshot logical volume if your Bitbucket Server home directory uses LVM,
 - a SAN-based backup if your Bitbucket Server home directory uses a Storage Area Network, or
 - `rsync`, if available.
- take backups of [Bitbucket Data Center](#) instances without having to bring nodes down manually.

On this page:

- [How it works](#)
- [What is backed up](#)
- [DIY Backups using Bash scripts](#)
- [Restoring a DIY Backup](#)
- [Canceling the backup](#)
- [Advanced - writing your own DIY Backup using the REST APIs](#)

Related pages:

- [Data recovery and backups](#)
- [Using the Bitbucket Server Backup Client](#)
- [Scheduling tasks on Linux](#)
- [Scheduling tasks on Windows](#)

Download the worked example scripts from Bitbucket:

[Download](#)

The key to reducing downtime is the use of optimal, vendor-specific database and file system backup tools. Such tools are generally able to take snapshots (though sometimes in a vendor-specific format) in much less time than the generic, vendor-neutral format used by the Bitbucket Server Backup Client.

Bitbucket Server DIY Backup does require you to write some code in a language of your choice to perform the required backup steps, using the REST API available for Bitbucket Server 4.0.

DIY Backup supports Windows and Linux platforms, and Bitbucket Server version 4.0 and higher. DIY Backup supports both Bitbucket Server and Bitbucket Data Center instances equally - any DIY Backup solution that works on one should work on the other without modification.

For information about other backup strategies for Bitbucket Server, see [Data recovery and backups](#). That page also discusses the tight coupling between the Bitbucket Server file system on disk and the database that the application uses.

Please note that the examples on this page are provided as guidance for developing a DIY Backup solution. As such, the third-party tools described are for example only – you will need to choose the tools that are appropriate to your own specific installation of Bitbucket Server.

Consult the vendor documentation for the third-party tools you choose – unfortunately, Atlassian can not provide support for those tools.

This page:

- Describes a complete DIY Backup solution for a PostgreSQL database and local filesystem, using `bash` shell scripts.
- Provides background information about how the Bitbucket Server REST API can be used for DIY Backups.

You can use this solution directly if your Bitbucket Server instance has the same or similar configuration, or use this as a starting point to develop your own DIY Backup solution tailored to your hardware configuration.

How it works

When you use DIY Backup instead of the Bitbucket Server Backup Client, you have complete control over the backup steps, and can implement any custom processes you like in the language of your choice. For example, you can use your database's incremental or fast snapshot tools and/or your file server's specific tools as part of a DIY Backup.

The DIY Backup works in a similar way to the [Bitbucket Server Backup Client](#) and does the following:

1. Prepares the Bitbucket Server instance for backup. This happens before Bitbucket Server is locked, so we want to do as much processing as possible here in order to minimize downtime later. For example, we can take an initial snapshot using incremental database and filesystem utilities. These do not have to be 100% consistent as Bitbucket Server is still running and modifying the database and filesystem. But taking the initial snapshot now may reduce the amount of work done later (while the application is locked), especially if the amount of data modified between backups is large. The steps include:
 - Taking an initial backup of the database (if it supports progressive/differential backups).
 - Doing an initial `r sync` of the home folder to the backup folder.
2. Initiates the backup, which will:
 - Lock the Bitbucket Server instance.
 - Drain and latch the connections to the database and the filesystem.
 - Wait for the drain/latch step to complete.
3. Once the instance is ready for backup we can start with the actual DIY Backup. This will include steps to:
 - Make a fully consistent backup of the database, using `pg_dump`.
 - Make a fully consistent backup of the filesystem, using `r sync`.
4. Notify the Bitbucket Server instance once the backup process finishes and unlock it.
5. Archive all files created during the backup into one big archive.

A user will get an error message if they try to access the web interface, or use the hosting services, when the application is in maintenance mode.

As an indication of the unavailability time that can be expected, in Atlassian's internal use we have seen downtimes for Bitbucket Server of 7–8 minutes with repositories totalling 6 GB in size when using the Bitbucket Server Backup Client. For comparison, using Bitbucket Server DIY Backup for the same repositories typically results in a downtime of less than a minute.

What is backed up

The Bitbucket Server DIY Backup backs up all the same data as the Bitbucket Server Backup Client:

- the database the instance is connected to (either the internal or external DB)
- managed Git repositories
- the Bitbucket Server logs
- installed plugins and their data

DIY Backups using Bash scripts

This section presents a complete DIY Backup solution that uses the following tools:

- `bash` - for scripting
- `jq` - an open source command line JSON processor for parsing the REST responses from Bitbucket Server
- `pg_dump` (or `sqlcmd`) - for backing up a PostgreSQL database
- `r sync` - for backing up the filesystem

- `tar` - for making a backup archive

This approach (with small modifications) can be used for running DIY Backups on:

- Linux and Unix
- OSX
- Windows with cygwin (note that cygwin Git is *not* supported by Bitbucket Server).

Bash scripts

You can download the example scripts from [Bitbucket](#) or simply clone the repository.

Running the Bash script

Once you have downloaded the Bash scripts, you need to create one file:

- `bitbucket.diy-backup.vars.sh` (you can copy `bitbucket.diy-backup.vars.sh.example` to start)

For example, here's how you might configure `bitbucket.diy-backup.vars.sh` if:

- your Bitbucket Server server is called `bitbucket.example.com`, uses port 7990, and has its home directory in `/bitbucket-home`
- you want to generate the backup in `/bitbucket-backup`, and store your `.tar.gz` backups in `/bitbucket-backup-archives`,
- you have a System Administrator in Bitbucket with the username "admin" and password "admin", and you run Bitbucket (and the backup scripts) as the OS user "atlbitbucket"

Example usage:

```
#!/bin/bash

##
# It is recommended to `chmod 600 bitbucket.diy-backup.vars.sh` after
# copying the template.
##

CURL_OPTIONS="-L -s -f"

# Which database backup script to use (ex: mssql, postgresql, mysql,
# ebs-collocated, rds)
BACKUP_DATABASE_TYPE=postgresql

# Which filesystem backup script to use (ex: rsync, ebs-home)
BACKUP_HOME_TYPE=rsync

# Which archive backup script to use (ex: tar, tar-gpg)
BACKUP_ARCHIVE_TYPE=tar

# Used by the scripts for verbose logging. If not true only errors will
# be shown.
BITBUCKET_VERBOSE_BACKUP=TRUE
```

```
# The base url used to access this bitbucket instance. It cannot end on
a '/'
BITBUCKET_URL=https://bitbucket.example.com:7990

# Used in AWS backup / restore to tag snapshots. It cannot contain
spaces and it must be under 100 characters long
INSTANCE_NAME=bitbucket

# The username and password for the user used to make backups (and have
this permission)
BITBUCKET_BACKUP_USER=admin
BITBUCKET_BACKUP_PASS=admin

# The name of the database used by this instance.
BITBUCKET_DB=bitbucket

# The path to bitbucket home folder (with trailing /)
BITBUCKET_HOME=/bitbucket-home

# OS level user and group information (typically: 'atlbitbucket' for
both)
BITBUCKET_UID=atlbitbucket
BITBUCKET_GID=atlbitbucket

# The path to working folder for the backup
BITBUCKET_BACKUP_ROOT=/bitbucket-backup

BITBUCKET_BACKUP_DB=${BITBUCKET_BACKUP_ROOT}/bitbucket-db/
BITBUCKET_BACKUP_HOME=${BITBUCKET_BACKUP_ROOT}/bitbucket-home/

# The path to where the backup archives are stored
BITBUCKET_BACKUP_ARCHIVE_ROOT=/bitbucket-backup-archives

# List of repo IDs which should be excluded from the backup
# It should be space separated: (2 5 88)
BITBUCKET_BACKUP_EXCLUDE_REPOS=( )

# PostgreSQL options
POSTGRES_HOST=localhost
POSTGRES_USERNAME=dbuser
POSTGRES_PASSWORD=dbpass
POSTGRES_PORT=5432

# MySQL options
MYSQL_HOST=
MYSQL_USERNAME=
MYSQL_PASSWORD=
MYSQL_BACKUP_OPTIONS=
```

```
# HipChat options
HIPCHAT_URL=https://api.hipchat.com
HIPCHAT_ROOM=
HIPCHAT_TOKEN=

# Options for the tar-gpg archive type
BITBUCKET_BACKUP_GPG_RECIPIENT=
```

The supplied `bitbucket.diy-backup.vars.sh` is written to use PostgreSQL, rsync, and tar by default. But if you want to use different tools, you can also customize the top section of this file:

Example usage:

```
# Which database backup script to use (ex: mssql, postgresql, mysql,
ebs-collocated, rds)
BACKUP_DATABASE_TYPE=postgresql

# Which filesystem backup script to use (ex: rsync, ebs-home)
BACKUP_HOME_TYPE=rsync

# Which archive backup script to use (ex: tar, tar-gpg)
BACKUP_ARCHIVE_TYPE=tar
```

You also need to create two directories for DIY Backup to work:

1. `${BITBUCKET_BACKUP_ROOT}` is a working directory (`/bitbucket-backup` in our example) where copies of Bitbucket Server's home directory and database dump are built during the DIY Backup process.
2. `${BITBUCKET_BACKUP_ARCHIVE_ROOT}` is the directory (`/bitbucket-backup-archives` in our example) where the final backup archives are saved.

The Bash scripts may be run on any host, provided it has:

- read/write access to the above `${BITBUCKET_BACKUP_ROOT}` and `${BITBUCKET_BACKUP_ARCHIVE_ROOT}` directories,
- read access to the `${BITBUCKET_HOME}` directory,
- read access to the database, and
- network access to run `curl` commands on the Bitbucket Server server.

This is true regardless of whether you have an instance of Bitbucket Server or Bitbucket Server Data Center. It doesn't matter whether the filesystem access is direct or over NFS, or whether the network access is direct to a Bitbucket Server node or to a load balancer / reverse proxy.

Once your `bitbucket.diy-backup.vars.sh` is correctly configured, run the backup in a terminal window:

```
$ ./bitbucket.diy-backup.sh
```

The first time you run the backup, rsync will do most of the work since the `/bitbucket-backup` working directory is initially empty. This is normal. Fortunately, this script performs one rsync before locking

Bitbucket Server, followed by a second `rsync` while Bitbucket Server is locked. This minimizes downtime.

On second and subsequent backup runs, `/bitbucket-backup` is already populated so the backup process should be faster. The output you can expect to see looks something like this:

```
$ ./bitbucket.diy-backup.sh
[http://localhost:7990/bitbucket] INFO: Prepared backup of DB
bitbucket in /bitbucket-backup/bitbucket-db/
building file list ... done.
sent 4.17M bytes received 484 bytes 2.78M bytes/sec
total size is 121.12M speedup is 29.06
[http://localhost:7990/bitbucket] INFO: Prepared backup of
/bitbucket-home to /bitbucket-backup/bitbucket-home/
[http://localhost:7990/bitbucket] INFO: locked with
'7187ae1824ce1ede38a8e7de4bccf58d9a8e1a7a'
[http://localhost:7990/bitbucket] INFO: backup started with
'82c73f89e790b27fef3032e81c7071388ae4e371'
[http://localhost:7990/bitbucket] INFO: Waiting for DRAINED
state..... done
[http://localhost:7990/bitbucket] INFO: db state 'DRAINED'
[http://localhost:7990/bitbucket] INFO: scm state 'DRAINED'
[http://localhost:7990/bitbucket] INFO: Performed backup of DB
bitbucket in /bitbucket-backup/bitbucket-db/
[http://localhost:7990/bitbucket] INFO: Backup progress updated to
50
building file list ... done.
sent 4.87M bytes received 484 bytes 3.25M bytes/sec
total size is 121.82M speedup is 24.99
[http://localhost:7990/bitbucket] INFO: Performed backup of
/bitbucket-home to /bitbucket-backup/bitbucket-home/
[http://localhost:7990/bitbucket] INFO: Backup progress updated to
100
[http://localhost:7990/bitbucket] INFO: Bitbucket instance unlocked
[http://localhost:7990/bitbucket] INFO: Archiving /bitbucket-backup
into /bitbucket-backup-archives/bitbucket-20150917-082818-498.tar.gz
[http://localhost:7990/bitbucket] INFO: Archived /bitbucket-backup
into /bitbucket-backup-archives/bitbucket-20150917-082818-498.tar.gz
```

Restoring a DIY Backup

When restoring Bitbucket Server, you must run the `bitbucket.diy-restore.sh` script on the machine that Bitbucket Server should be restored to. In order to ensure accidental restores do not delete existing data, you should never restore into an existing home directory.

The new database should be configured following the instructions in [Connecting Bitbucket Server to an external database](#) and its sub-page that corresponds to your database type.

To see the available backups in your `${BITBUCKET_BACKUP_ARCHIVE_ROOT}` directory, just type:

```
$ ./bitbucket.diy-restore.sh
```

You should see output similar to this:

```
$ ./bitbucket.diy-restore.sh
Usage: ./bitbucket.diy-restore.sh <backup-file-name>.tar.gz
Available backups:
bitbucket-20150917-082818-498.tar.gz
bitbucket-20150918-083745-001.tar.gz
```

To restore a backup, run `bitbucket.diy-restore.sh` with the file name as the argument:

```
$ ./bitbucket.diy-restore.sh bitbucket-20150917-082818-498.tar.gz
```

You should see output like this:

```
$ ./bitbucket.diy-restore.sh bitbucket-20150917-082818-498.tar.gz
[http://localhost:7990/bitbucket] INFO: Extracted
/bitbucket-backup-archives/bitbucket-20150917-082818-498.tar.gz into
/tmp/bitbucket.diy-restore.dQsbzU
[http://localhost:7990/bitbucket] INFO: Performed restore of
/tmp/bitbucket.diy-restore.dQsbzU/bitbucket-db to DB bitbucket2
[http://localhost:7990/bitbucket] INFO: Performed restore of
/tmp/bitbucket.diy-restore.dQsbzU/bitbucket-home to /bitbucket-home2
```

Canceling the backup

You can cancel the running backup operation if necessary.

To cancel the backup:

1. Copy the cancel token echoed in the terminal (or the Command Prompt on Windows). Look for the line "backup started with token"

```
$ ./bitbucket.diy-backup.sh
[http://localhost:7990/bitbucket] INFO: Prepared backup of DB
bitbucket in /bitbucket-backup/bitbucket-db/
building file list ... done.
sent 4.17M bytes received 484 bytes 2.78M bytes/sec
total size is 121.12M speedup is 29.06
[http://localhost:7990/bitbucket] INFO: Prepared backup of
/bitbucket-home to /bitbucket-backup/bitbucket-home/
[http://localhost:7990/bitbucket] INFO: locked with
'7187ae1824ce1ede38a8e7de4bccf58d9a8e1a7a'
[http://localhost:7990/bitbucket] INFO: backup started with
'82c73f89e790b27fef3032e81c7071388ae4e371'
[http://localhost:7990/bitbucket] INFO: Waiting for DRAINED
state..... done
[http://localhost:7990/bitbucket] INFO: db state 'DRAINED'
[http://localhost:7990/bitbucket] INFO: scm state 'DRAINED'
```

E.g. use "82c73f89e790b27fef3032e81c7071388ae4e371"

2. Go to the Bitbucket Server interface in your browser. Bitbucket Server will display this screen:

Performing Backup

Backup is currently underway and shouldn't take long.

Backing up Bitbucket home

[Cancel backup](#)

- Click **Cancel backup**, and enter the cancel token:

Performing Backup

Backup is currently underway and shouldn't take long.

Backing up Bitbucket home

Authentication token

To cancel, please enter an authentication token. This can be found in the Bitbucket logs

[Cancel backup](#)

- Click **Cancel backup**.

Note that Bitbucket Server will still be locked in maintenance mode. Repeat these steps using the "locked with" token (e.g. "7187ae1824ce1ede38a8e7de4bccf58d9a8e1a7a") to exit maintenance mode as well, and unlock Bitbucket Server.

Advanced - writing your own DIY Backup using the REST APIs

This section is optional and provides background information about how you might use the Bitbucket Server REST APIs if you need to rewrite the DIY Backup scripts described above in your preferred language or to customize them heavily.

Note that this discussion shows `curl` commands in Bash, however you can use any language.

The following steps are involved:

Preparation

Before you lock Bitbucket Server you can perform any preparation you like. It makes sense to perform as much processing as possible before you lock the application, to minimize downtime later. For example, you could perform an `rsync`:

```
rsync -avh --delete --delete-excluded --exclude=/caches/
--exclude=/data/db.* --exclude=/export/ --exclude=/log/
--exclude=/plugins/.*/ --exclude=/tmp --exclude=/.lock
${BITBUCKET_HOME} ${BITBUCKET_BACKUP_HOME}
```


Lock the Bitbucket Server instance

The next step in making a backup of a Bitbucket Server instance is to lock the instance for maintenance. This can be done using a POST request to the `/mvc/maintenance/lock` REST point (where `BITBUCKET_URL` points to the Bitbucket Server instance, `BITBUCKET_BACKUP_USER` is a Bitbucket Server user with backup permissions, and `BITBUCKET_BACKUP_PASS` is this user's password).

REQUEST

```
curl -s \
  -u
  ${BITBUCKET_BACKUP_USER}:${BITBUCKET_BACKUP_PASS} \
  -X POST \
  -H "Content-type:
  application/json" \

  "${BITBUCKET_URL}/mvc/maintenance/lock"
```

RESPONSE

```
{
  "unlockToken": "0476adeb6cde
  3a41aa0cc19fb394779191f5d306
  ",
  "owner": {
    "displayName": "admin",
    "name": "admin"
  }
}
```

If successful, the Bitbucket Server instance will respond with a 202 and will return a response JSON similar to the one above. The `unlockToken` should be used in all subsequent requests where `$BITBUCKET_LOCK_TOKEN` is required. This token can also be used to manually unlock the instance.

Start the backup process

Next, all connections to both the database and the filesystem must be drained and latched. Your code must handle backing up of *both* the filesystem and the database.

At this point, you should make a POST request to `/mvc/admin/backups`. Notice that the `curl` call includes the `?external=true` parameter:

REQUEST

```
curl -s \
  -u
  ${BITBUCKET_BACKUP_USER}:${BITBUCKET_BACKUP_PASS} \
  -X POST \
  -H
  "X-Atlassian-Maintenance-Token: ${BITBUCKET_LOCK_TOKEN}" \
  -H "Accept:
  application/json" \
  -H "Content-type:
  application/json" \

  "${BITBUCKET_URL}/mvc/admin/backups?external=true"
```

RESPONSE

```
{
  "id": "d2e15c3c2da282b0990e8
  efb30b4bffbcbf09e04",
  "progress": {
    "message": "Closing
    connections to the current
    database",
    "percentage": 5
  },
  "state": "RUNNING",
  "type": "BACKUP",
  "cancelToken": "d2e15c3c2da2
  82b0990e8efb30b4bffbcbf09e04
  "
}
```

If successful the instance will respond with 202 and a response JSON similar to the one above will be returned. The `cancelToken` can be used to manually cancel the back up process.

Wait for the instance to complete preparation.

Part of the back up process includes draining and latching the connections to the database and the filesystem. Before continuing with the back up we have to wait for the instance to report that this has been done. To get details on the current status we make a GET request to the `/mvc/maintenance` REST point.

REQUEST	RESPONSE
<pre>curl -s \ -u \${BITBUCKET_BACKUP_USER}:\${BITBUCKET_BACKUP_PASS} \ -X GET \ -H "X-Atlassian-Maintenance-Token: \${BITBUCKET_LOCK_TOKEN}" \ -H "Accept: application/json" \ -H "Content-type: application/json" \ "\${BITBUCKET_URL}/mvc/maintenance"</pre>	<pre>{ "task":{ "id":"0bb6b2ed52a6a12322e515e88c5d515d6b6fa95e", "progress":{ "message":"Backing up Bitbucket home", "percentage":10 }, "state":"RUNNING", "type":"BACKUP" }, "db-state":"DRAINED", "scm-state":"DRAINED" }</pre>

This causes the Bitbucket Server instance to report its current state. We have to wait for both `db-state` and `scm-state` to have a status of `DRAINED` before continuing with the backup.

Perform the actual backup

At this point we are ready to create the actual backup of the filesystem. For example, you could use `rsync` again:

```
rsync -avh --delete --delete-excluded --exclude=/cache/
--exclude=/data/db.* --exclude=/export/ --exclude=/log/
--exclude=/plugins/.*/ --exclude=/tmp --exclude=/.lock
${BITBUCKET_HOME} ${BITBUCKET_BACKUP_HOME}
```

The `rsync` options shown here are for example only, but indicate how you can include only the required files in the backup process and exclude others. Consult the documentation for `rsync`, or the tool of your choice, for a more detailed description.

When creating the database backup you could use your vendor-specific database backup tool, for example `pg_dump` if you use PostgreSQL:

```
pg_dump -Fd ${BITBUCKET_DB} -j 5 --no-synchronized-snapshots -f
${BITBUCKET_BACKUP_DB}
```

While performing these operations, good practice is to update the instance with progress on the backup so that it's visible in the UI. This can be done by issuing a `POST` request to `/mvc/admin/backups/progress/client` with the token and the percentage completed as parameters:

REQUEST

```
curl -s \
  -u ${BITBUCKET_BACKUP_USER}:${BITBUCKET_BACKUP_PASS} \
  -X POST \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \

"${BITBUCKET_URL}/mvc/admin/backups/progress/client?token=${BITBUCKET_LOCK_TOKEN}&percentage=${BITBUCKET_BACKUP_PERCENTAGE}"
```

Bitbucket Server will respond to this request with an empty 202 if everything is OK.

When displaying progress to users, Bitbucket Server divides the 100 percent progress into 90 percent user DIY Backup, and 10 percent application preparation. This means, for example, if your script sends percent age=0, Bitbucket Server may display up to 10 percent progress for its own share of the backup work.

Inform the Bitbucket Server instance that the backup is complete

Once we've finished the backup process we must report to the Bitbucket Server instance that progress has reached 100 percent. This is done using a similar request to the progress request. We issue a POST request to `/mvc/admin/backups/progress/client` with the token and 100 as the percentage:

REQUEST

```
curl -s \
  -u ${BITBUCKET_BACKUP_USER}:${BITBUCKET_BACKUP_PASS} \
  -X POST \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \

"${BITBUCKET_URL}/mvc/admin/backups/progress/client?token=${BITBUCKET_LOCK_TOKEN}&percentage=100"
```

Bitbucket Server will respond with an empty 202 if everything is OK. The back up process is considered completed once the percentage is 100. This will unlatch the database and the filesystem for this Bitbucket Server instance.

Unlock the Bitbucket Server instance

The final step we need to do in the back up process is to unlock the instance. This is done with a DELETE request to the `/mvc/maintenance/lock` REST point:

REQUEST

```
curl -s \
  -u ${BITBUCKET_BACKUP_USER}:${BITBUCKET_BACKUP_PASS} \
  -X DELETE \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \

"${BITBUCKET_URL}/mvc/maintenance/lock?token=${BITBUCKET_LOCK_TOKEN}"
```

The Bitbucket Server instance will respond to this request with an empty 202 if everything is OK, and will unlock access.

Lockout recovery process

This page describes how to recover administrator access for Stash 2.11, now known as Bitbucket Server 4.X+, and later. For releases prior to that, refer to [Restoring the Bitbucket Server Administrator's Password](#).

As an administrator, you may find yourself locked out of Bitbucket Server and unable to log in. This situation can arise when all users are managed externally from Bitbucket Server, and Bitbucket Server becomes unable to access those user directories for some reason, including:

- The external user directory server is not accessible (because the network is down, or the directory is down, or the directory has been moved to another IP address).
- Users are managed within a JIRA application and the Application Link from Bitbucket Server to a JIRA application has been accidentally deleted.
- The admin password has been forgotten or lost.
- The admin account is shaded by a remote account in an LDAP or JIRA application that is connected to Bitbucket Server but which is unavailable.

The lockout recovery process for Bitbucket Server is:

1. Edit the <Bitbucket Server installation directory>\bin\setenv.sh file (or setenv.bat on Windows) and add the "-Datlassian.recovery.password=temporarypassword" argument to the JVM_SUPPORT_RECOMMENDED_ARGS property. The property value must be non-blank, and should look like this when you've done that:

```
#
# Occasionally Atlassian Support may recommend that you set some specific JVM arguments. You can use this variable
# below to do that.
#
JVM_SUPPORT_RECOMMENDED_ARGS="-Datlassian.recovery.password=temporarypassword"
```

Here we are using "temporarypassword", but you should use your own value.

2. Start Bitbucket Server.
3. Log in to Bitbucket Server using the 'recovery_admin' username and the temporary password specified in Step 1.
4. Repair your Bitbucket Server configuration. We strongly recommend that you do not perform other actions while Bitbucket Server is in recovery mode.
5. Confirm your ability to log in with your usual admin profile.
6. Shut down Bitbucket Server, remove the atlassian.recovery.password argument from setenv.sh, and restart Bitbucket Server as usual.

Scaling Bitbucket Server

This page discusses performance and hardware considerations when using Bitbucket Server.

Note that [Bitbucket Data Center](#), not discussed on this page, uses a cluster of Bitbucket Server nodes to provide Active/Active failover, and is the deployment option of choice for larger enterprises that require high availability and performance at scale.

Hardware requirements

The type of hardware you require to run Bitbucket Server depends on a number of factors:

- The number and frequency of clone operations. Cloning a repository is one of the most demanding operations. One major source of clone operations is continuous integration. When your CI builds involve multiple parallel stages, Bitbucket Server will be asked to perform multiple clones concurrently, putting significant load on your system.
- The size of your repositories – there are many operations in Bitbucket Server that require more memory and more CPUs when working with very large repositories. Furthermore, huge Git repositories (larger than a few GBs) are likely to impact the performance of the Git client – see [this discussion](#).
- The number of users.

The following are rough guidelines for choosing your hardware:

- Estimate the number of concurrent clones that are expected to happen regularly (look at continuous integration). Add one CPU for every 2 concurrent clone operations. Note that enabling the [SCM Cache Plugin](#) (bundled with Bitbucket Server from version 2.5.0) can help to reduce the cloning load on the Bitbucket Server instance due to CI polling. See [Scaling Bitbucket Server for Continuous Integration performance](#).
- Estimate or calculate the average repository size and allocate 1.5 x number of concurrent clone operations x min(repository size, 700MB) of memory.

On this page:

- [Hardware requirements](#)
- [Understanding Bitbucket Server's resource usage](#)
 - [Memory](#)
 - [CPU](#)
 - [Clones examined](#)
- [Configuring Bitbucket Server scaling options and system properties](#)
- [Database requirements](#)

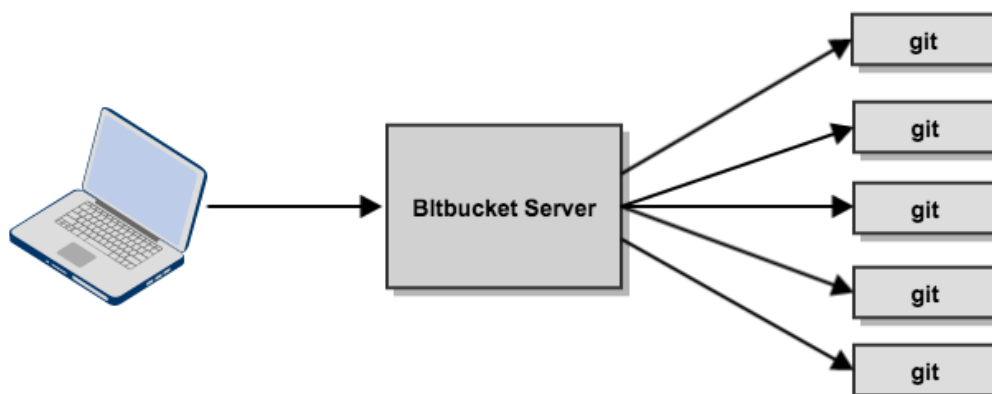
Related pages:

- [Using Bitbucket Server in the enterprise](#)
- [Resources for migrating to Git](#)
- [Bitbucket Server production server data](#)
- [Scaling Bitbucket Server for Continuous Integration performance](#)
- [Potential performance impact of embedded Crowd directory ordering](#)
- [Bitbucket Server config properties](#)

Understanding Bitbucket Server's resource usage

Most of the things you do in Bitbucket Server involve both the Bitbucket Server instance and one or more Git processes created by Bitbucket Server. For instance, when you view a file in the Bitbucket Server web application, Bitbucket Server processes the incoming request, performs permission checks, creates a Git process to retrieve the file contents and formats the resulting webpage. In serving most pages, both the Bitbucket Server instance and Git processes are involved. The same is true for the 'hosting' operations: pushing your commits to Bitbucket Server, cloning a repository from Bitbucket Server or fetching the latest changes from Bitbucket Server.

As a result, when configuring Bitbucket Server for performance, CPU and memory consumption for both Bitbucket Server *and* Git should be taken into account.



Memory

When deciding on how much memory to allocate for Bitbucket Server, the most important factor to consider is the amount of memory required for Git. Some Git operations are fairly expensive in terms of memory consumption, most notably the initial push of a large repository to Bitbucket Server and cloning large repositories from Bitbucket Server. For large repositories, it is not uncommon for Git to use up to 500 MB of memory during the clone process. The numbers vary from repository to repository, but as a rule of thumb 1.5 x the repository size on disk (contents of the `.git/objects` directory) is a rough estimate of the required memory for a single clone operation for repositories up to 400 MB. For larger repositories, memory usage flattens out at about 700 MB.

The clone operation is the most memory intensive Git operation. Most other Git operations, such as viewing file history, file contents and commit lists are lightweight by comparison.

Bitbucket Server has been designed to have fairly constant memory usage. Any pages that could show large amounts of data (e.g. viewing the source of a multi-megabyte file) perform incremental loading or have hard limits in place to prevent Bitbucket Server from holding on to large amounts of memory at any time. In general, the default memory settings (max. 768 MB) should be sufficient to run Bitbucket Server. The maximum amount of memory available to Bitbucket Server can be configured in `setenv.sh` or `setenv.bat`.

The memory consumption of Git is not managed by the memory settings in `setenv.sh` or `setenv.bat`. The Git processes are executed outside of the Java virtual machine, and as a result the JVM memory settings do not apply to Git.

CPU

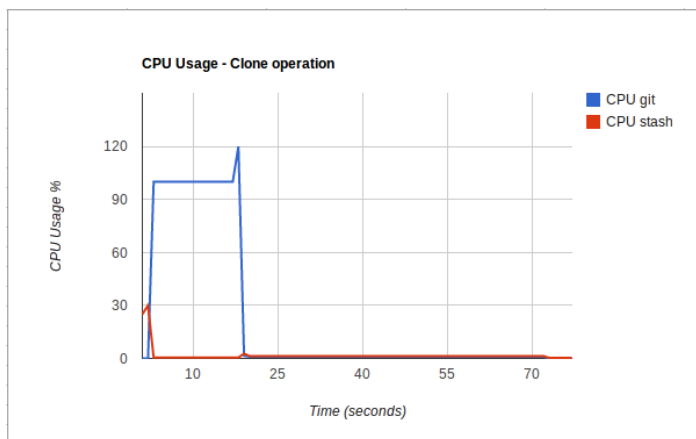
In Bitbucket Server, much of the heavy lifting is delegated to Git. As a result, when deciding on the required hardware to run Bitbucket Server, the CPU usage of the Git processes is the most important factor to consider. And, as is the case for memory usage, cloning large repositories is the most CPU intensive Git operation. When you clone a repository, Git on the server side will create a *pack file* (a compressed file containing all the commits and file versions in the repository) that is sent to the client. While preparing a pack file, CPU usage will go up to 100% for one CPU.

Encryption (either SSH or HTTPS) will have a significant CPU overhead if enabled. As for which of SSH or HTTPS is to be preferred, there's no clear winner, each has advantages and disadvantages as described in the following table.

	HTTP	HTTPS	SSH
Encryption	No CPU overhead for encryption, but plaintext transfer and basic auth may be unacceptable for security.	Encryption has CPU overhead, but this can be offloaded to a separate proxy server (if the secure connection is terminated there).	Encryption has CPU overhead.
Authentication	Authentication is slower – it requires remote authentication with the LDAP or Crowd server.		Authentication is much faster – it only requires a simple lookup.
Cloning	Cloning a repository is slightly slower – it takes at least 2 and sometimes more requests, each of which needs authentication and permission checks. The extra overhead is small though - usually in the 10-100ms range		Cloning a repository takes only a single request.

Clones examined

Since cloning a repository is the most demanding operation in terms of CPU and memory, it is worthwhile analyzing the clone operation a bit closer. The following graphs show the CPU and memory usage of a clone of a 220 MB repository:

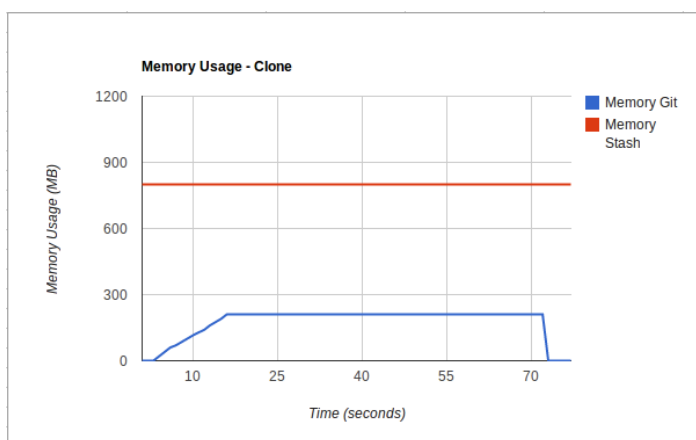


Git process (blue line)

- CPU usage goes up to 100% while the pack file is created on the server side.
- CPU peaks at 120% when the pack file is compressed (multiple CPUs used).
- CPU drops back to 0.5% while the pack file is sent back to the client.

Bitbucket Server (red line)

- CPU usage briefly peaks at 30% while the clone request is processed.
- CPU drops back to 0% while Git prepares the pack file.
- CPU hovers around 1% while the pack file is sent to the client.

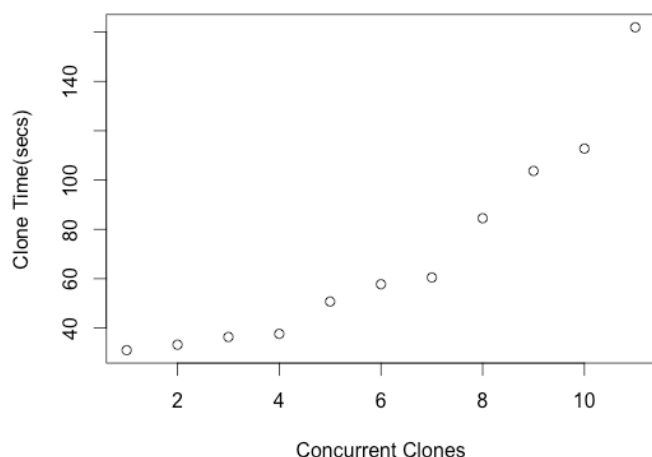


Git process (blue line)

- Memory usage slowly climbs to 270 MB while preparing the pack file.
- Memory stays at 270 MB while the pack file is transmitted to the client.
- Memory drops back to 0 when the pack file transmission is complete.

Bitbucket Server (red line)

- Memory usage hovers around 800 MB and is not affected by the clone operation.



This graph shows how concurrency affects average response times for clones:

- Vertical axis: average response times.
- Horizontal axis: number of concurrent clone operations.

The measurements for this graph were done on a 4 CPU server with 12 GB of memory. Response times become exponentially worse as the number of concurrent clone operations exceed the number of CPUs.

Configuring Bitbucket Server scaling options and system properties

Bitbucket Server limits the number of Git operations that can be executed concurrently, to prevent the performance for all clients dropping below acceptable levels. These limits can be adjusted – see [Bitbucket Server config properties](#).

Database requirements

The size of the database required for Bitbucket Server depends in large part on the number of repositories and the number of commits in those repositories.

A very rough guideline is: $100 + ((\text{total number of commits across all repos}) / 2500)$ MB.

So, for example, for 20 repositories with an average of 25,000 commits each, the database would need $100 + (20 * 25,000 / 2500) = 300$ MB.

Scaling Bitbucket Server for Continuous Integration performance

If you've got CI or other automatic tooling set up to poll Bitbucket Server for changes, you can end up with high load on your Bitbucket Server instance. Consider for instance a CI server that has a number of builds set up for a given repository. Each of those builds polls Bitbucket Server for changes and when it detects a change, it starts a new build. If your CI server supports parallel and/or chained build steps, each of these builds typically results in multiple clone operations of the same repository. The result: lots of polling for changes, and bursts of clones of a repository.

Caching

CI results in highly repetitive calls to Bitbucket Server: polling for changes typically results in the same response 90% of the time and a build triggers a number of identical clone calls to Bitbucket Server. Both operations can benefit greatly from caching when you experience repetitive load from CI.

Bitbucket Server 2.5, and later versions, ship with the SCM Cache Plugin already bundled. The plugin is enabled by default, but note that ref advertisement is disabled by default – see the 'Limitations' section below. The plugin is also available from the [Atlassian Marketplace](#).

On this page:

- [Limitations](#)
- [Considerations](#)
- [Enabling and disabling caching](#)
- [Configuration](#)
- [REST API](#)

Related pages:

- [Scaling Bitbucket Server](#)
- [Bitbucket Server config properties](#)

Limitations

- Caching can be applied to 'ref advertisement' (polling for changes), clone and shallow clone requests only. Fetch/pull operations are not cached, but these operations will still benefit from the 'ref advertisement' cache.
- As a precaution, ref advertisement caching is *disabled* by default. The openness of the plugin system means that plugins (or manually installed git hooks) could be updating refs in repository without the caching plugin detecting these changes. The result would be a stale refs cache and failing clone/fetch operations. However, if you know that there are no plugins or git hooks installed that make changes to the repository directly, you can enable the ref advertisement caching using the system properties listed in the Configuration section below. Note that as an additional precaution, the ref advertisement caches are configured to automatically expire after a minute.

Considerations

Cache data is stored on disk for both ref advertisements and for clone operations for a configurable period of time. Since large instances can potentially consume an entire disk the SCM Cache Plugin monitors remaining disk space and is automatically disabled when the configured minimum free disk space is reached.

See [Bitbucket Server config properties](#) for descriptions of the available system properties.

Enabling and disabling caching

Enable the SCM Cache Plugin from the admin area in Bitbucket Server. Click **Manage Add-ons** (under 'Add-Ons') and filter for system add-ons. Click **SCM Cache Plugin for Bitbucket Server** and then either **Enable** or **Disable**.

Configuration

The SCM Cache Plugin for Bitbucket Server can be configured using either the REST endpoint (some settings, not all) or the system properties in `<BITBUCKET_HOME>/bitbucket.properties`. Settings configured through REST are considered *before* the settings in `bitbucket.properties`.

REST API

Method	Url	Description
GET	<code>/rest/scm-cache/latest/config/protocols</code>	Retrieve the protocols for which caching has been enabled.
PUT	<code>/rest/scm-cache/latest/config/protocols/{protocol}</code>	Enable caching of hosting requests for the protocol (HTTP or SSH).
DELETE	<code>/rest/scm-cache/latest/config/protocols/{protocol}</code>	Disable caching of hosting requests for the protocol (HTTP or SSH).
GET	<code>/rest/scm-cache/latest/config/refs/enabled</code>	Retrieve whether ref advertisement caching is enabled (true) or disabled (false).
PUT	<code>/rest/scm-cache/latest/config/refs/enabled/{status}</code>	Enable (status = true) or disable (status = false) ref advertisement caching.
GET	<code>/rest/scm-cache/latest/config/refs/ttl</code>	Retrieve the expiry in seconds for the ref advertisement caches.
PUT	<code>/rest/scm-cache/latest/config/refs/ttl/{expiryInSec}</code>	Set the expiry in seconds for the ref advertisement caches.
GET	<code>/rest/scm-cache/latest/config/upload-pack/enabled</code>	Retrieve whether clone caching is enabled (true) or disabled (false).

PUT	/rest/scm-cache/latest/config/upload-pack/enabled/{status}	Enable (status = true) or disable (status = false) clone caching.
GET	/rest/scm-cache/latest/config/upload-pack/ttl	Retrieve the expiry in seconds for the clone caches.
PUT	/rest/scm-cache/latest/config/upload-pack/ttl/{expiryInSec}	Set the expiry in seconds for the clone caches.
GET	/rest/scm-cache/latest/caches	Retrieve information about the current caches: size, number of cache hits and misses, etc.
DELETE	/rest/scm-cache/latest/caches	Clear all caches.
GET	/rest/scm-cache/latest/caches/{projectKey}/{repoSlug}	Retrieve information about the caches for the repository identified by projectKey and repoSlug: size, number of cache hits and misses, etc.
DELETE	/rest/scm-cache/latest/caches/{projectKey}/{repoSlug}	Clear the caches for the repository identified by projectKey and repoSlug.

Bitbucket Server production server data

This page provides some data around the Bitbucket Server production instance that we run internally at Atlassian. We're providing this to give some idea of how Bitbucket Server performs in a production environment. Please realise that this information is entirely specific to this particular instance – the details of your own installation may result in different performance data.

This data was collected with New Relic in February 2013, when the server was running a pre-release version of Bitbucket Server 2.2.

On this page:

- [Hardware](#)
- [Load](#)
- [Server load](#)
- [Git operations](#)

Hardware

The performance data below was gathered from the Atlassian Bitbucket Server production server running on:

- Virtualized hardware
- 4 Hyper-threaded cores
- 12 GB RAM

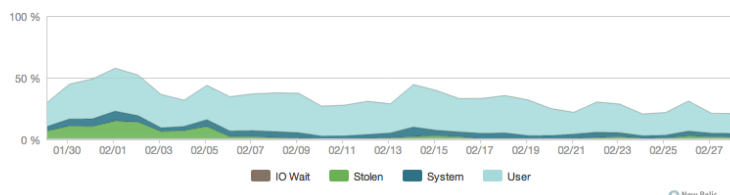
Load

Load data summary for February 2013:

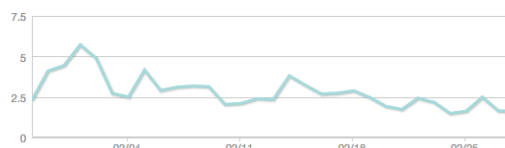
Type	Load
CPU usage	less than 30% on average
Load average	less than 3 on average
Physical Memory	peaked at 31%
Processes	Git: 17.3% CPU Java: 18.8% CPU
Clones	on average less than 300ms
Git operations/hour	peaking at 11,000 with an average of about 3,500
Concurrent connections/hour	peaking at 100 connections with an average of about 40 concurrent connections
CI running against Bitbucket Server instance	3 build servers with approximately 300 agents

Server load

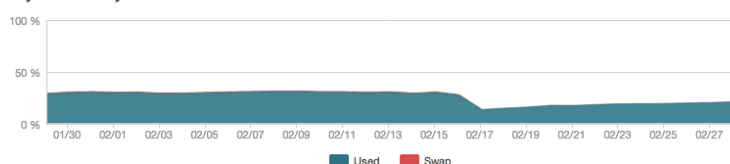
CPU usage



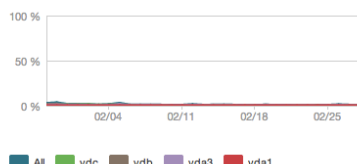
Load average



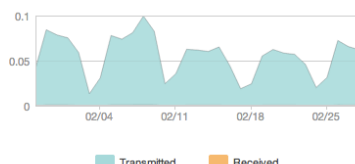
Physical memory



Disk I/O utilization



Network I/O (Gb/s)



stash-prod.private.atlassian.com

8 CPUs
12 GB RAM
Intel QEMU
Scientific Linux release 6.2 (Carbon)
Linux 2.6.32-220.7.1.el6.x86_64
x86_64
New Relic agent 1.1.2.124

Apps	Response time	Throughput	Errors
j2ee_stash-prod.private.atlassian.com	1,290 ms	570 rpm	1.46 %

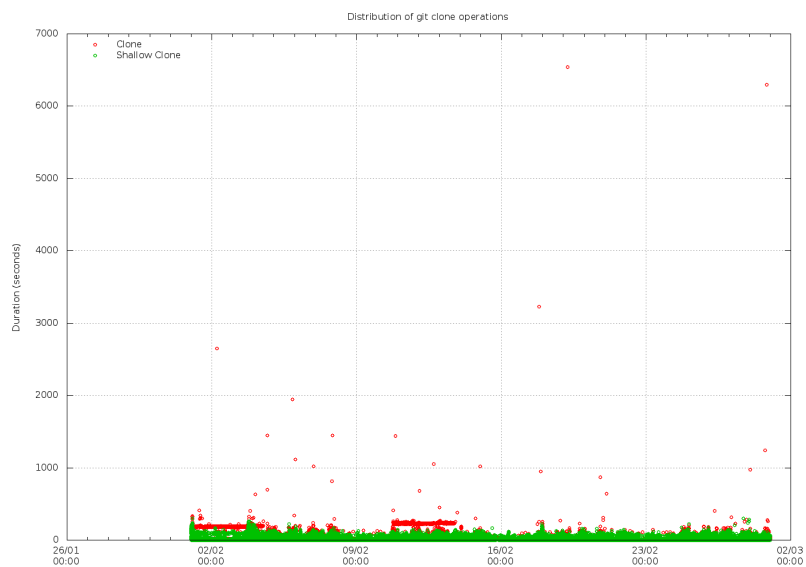
Processes >	User	Count	CPU	Memory
git	j2ee_stash-prod.private.atlassian.com-run	5	17.3%	226 MB
java	j2ee_stash-prod.private.atlassian.com-run	1	13.8%	836 MB
bzip2	root	1	7.0%	7.01 MB
portreserve	root	2	6.2%	393 MB
rsync	j2ee_stash-prod.private.atlassian.com-run	1	6.2%	90.8 MB

65 more...

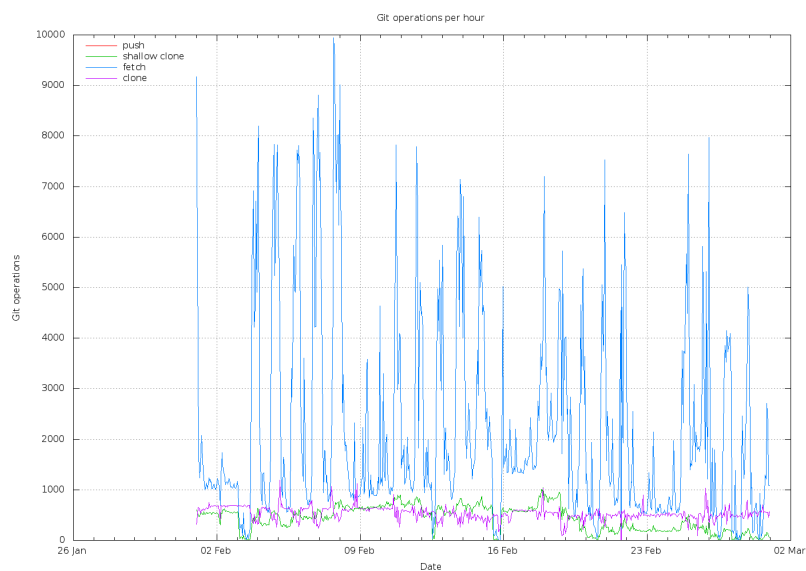
Recent stash-prod.private.atlassian.com events

Git operations

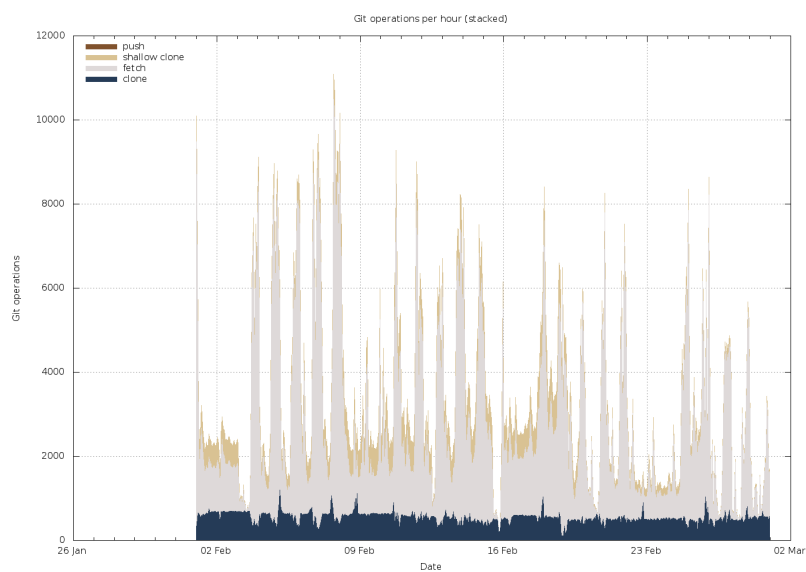
Git clone operations



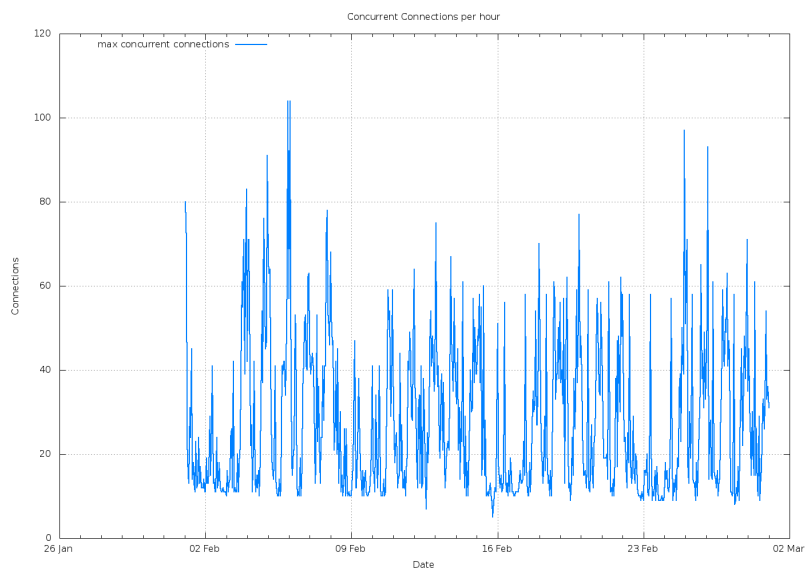
Git operations per hour



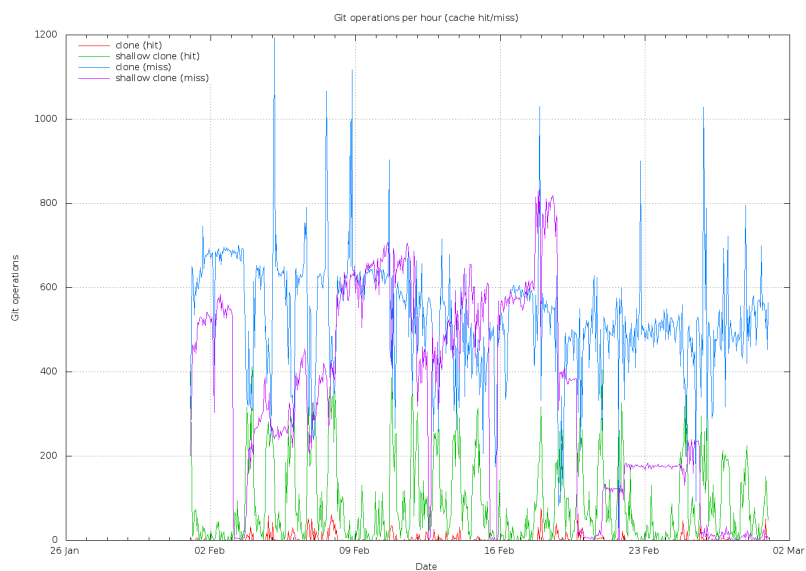
Git operations per hour (stacked)



Concurrent connections per hour



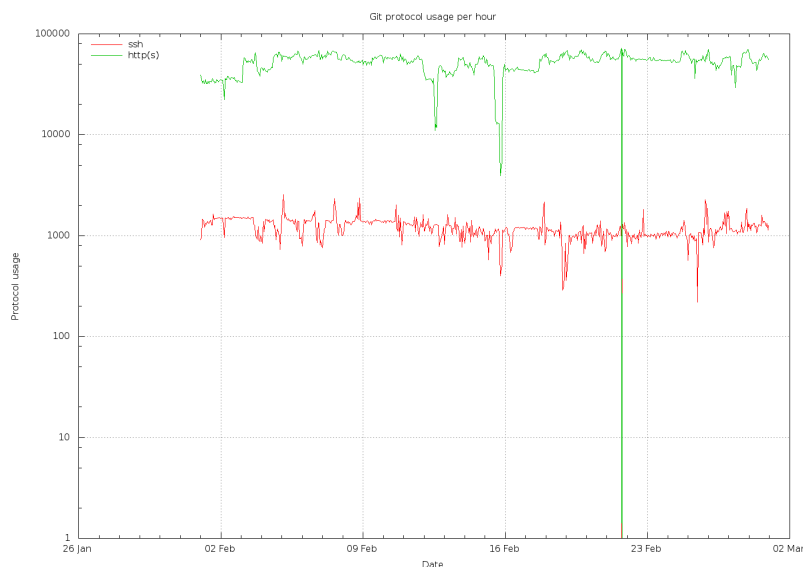
Git operations - cache hit/miss



Git operations - cache hit/miss



Git protocol usage per hour



High availability for Bitbucket Server

This page...

... describes how to set up a single Bitbucket Server instance in a highly available configuration.

For production installs...

... we highly recommend that you first read [Using Bitbucket Server in the enterprise](#).

For Active/Active HA with Bitbucket Server...

... see [Bitbucket Data Center](#) instead.

If Bitbucket Server is a critical part of your development workflow, maximizing application availability becomes an important consideration. There are many possible configurations for setting up a HA environment for Bitbucket Server, depending on the infrastructure components and software (SAN, clustered databases, etc.) you have at your disposal. This guide provides a high-level overview and the background information you need to be able to set up a single Bitbucket Server in a highly available configuration.

Note that Atlassian's [Bitbucket Data Center](#) product uses a cluster of Bitbucket Server nodes to provide Active/Active failover. It is the deployment option of choice for larger enterprises that require high availability and performance at scale, and is fully supported by Atlassian. Read about [Failover for Bitbucket Data Center](#).

Please note that your feedback and comments are welcome! We very much value additional lessons learned from your experience with alternative scenarios!

- [High availability](#)
- [Understanding the availability requirements for Bitbucket Server](#)
- [Failover options](#)
- [Automatic correction](#)
- [Cold standby](#)
- [System setup](#)
- [Licensing](#)

High availability

[High availability](#) describes a set of practices aimed at delivering a specific level of "availability" by eliminating and/or mitigating failure through redundancy. Failure can result from unscheduled down-time due to network errors, hardware failures or application failures, but can also result from failed application upgrades. Setting up a highly available system involves:

Proactive Concerns

- Change management (including staging and production instances for change implementation)
- Redundancy of network, application, storage and databases
- Monitoring system(s) for both the network and applications

Reactive Concerns

- Technical [failover](#) mechanisms, either automatic or scripted semi-automatic with manual switchover
- Standard Operating Procedure for guided actions during crisis situations

This guide assumes that processes such as change management are already covered and will focus on **redundancy / replication** and **failover procedures**. When it comes to setting up your infrastructure to quickly recover from system or application failure, you have different options. These options vary in the level of uptime they can provide. In general, as the required uptime increases, the complexity of the infrastructure and the knowledge required to administer the environment increases as well (and by extension the cost goes up as well).

Understanding the availability requirements for Bitbucket Server

Central version control systems such as Subversion, CVS, ClearCase and many others require the central server to be available for any operation that involves the version control system. Committing code, fetching the latest changes from the repository, switching branches or retrieving a diff all require access to the central version control system. If that server goes down, developers are severely limited in what they can do. They can continue coding until they're ready to commit, but then they're blocked.

Git is a distributed version control system and developers have a full clone of the repository on their machines. As a result, most operations that involve the version control system don't require access to the central repository. When Bitbucket Server is unavailable developers are not blocked to the same extent as with a central version control system.

As a result, the availability requirements for Bitbucket Server *may* be less strict than the requirements for say Subversion.

Consequences of Bitbucket Server unavailability	
✔ Unaffected	✘ Affected
Developer: • Commit code • Create branch • Switch branches • Diff commits and files • ... • Fetch changes from fellow developers	Developer: • Clone repository • Fetch changes from central repository • Push changes to central repository • Access Bitbucket Server UI - create/do pull requests, browse code Build server: • Clone repository • Poll for changes Continuous Deployment: • Clone repository

Failover options

High availability and recovery solutions can be categorized as follows:

Failover option	Recovery time	Description	Possible with Bitbucket Server

Automatic correction / restart	2-10 min (application failure) hours-days (system failure)	• Single node, no secondary server available • Application and server are monitored • Upon failure of production system, automatic restarting is conducted via scripting • Disk or hardware failure may require reprovisioning of the server and restoring application data from a backup	✓
Cold standby	2-10 min	• Secondary server is available • Bitbucket Server is NOT running on secondary server • Filesystem and (optionally) database data is replicated between the 'active' server and the 'standby' server • All requests are routed to the 'active' server • On failure, Bitbucket Server is started on the 'standby' server and shut down on the 'active' server. All requests are now routed to the 'standby' server, which becomes 'active'.	✓
Warm standby	0-30 sec	• Secondary service is available • Bitbucket Server is running on both the 'active' server and the 'standby' server, but all requests are routed to the 'active' server • Filesystem and database data is replicated between the 'active' server and the 'standby' server • All requests are routed to the 'active' server • On failure, all requests are routed to the 'standby' server, which becomes 'active' • ✗ This configuration is currently not supported by Bitbucket Server, because Bitbucket Server uses in-memory caches and locking mechanisms. At this time, Bitbucket Server only supports a single application instance writing to the Bitbucket Server home directory at a time.	✗
Active/Active	< 5 sec	• Provided by Bitbucket Data Center, using a cluster of Bitbucket Server nodes and a load balancer. • Bitbucket Server is running, and serving requests, on all cluster nodes. • Filesystem and database data is shared by all cluster nodes. Clustered databases are not yet supported. • All requests are routed to the load balancer, which distributes requests to the available cluster nodes. If a cluster node goes down, the load balancer immediately detects the failure and automatically directs requests to the other nodes within seconds. • Bitbucket Data Center is the deployment option of choice for larger enterprises that require high availability and performance at scale.	✓

Automatic correction

Before implementing failover solutions for your Bitbucket Server instance consider evaluating and leveraging automatic correction measures. These can be implemented through a monitoring service that watches your application and performs scripts to start, stop, kill or restart services.

1. A Monitoring Service detects that the system has failed.
2. A correction script attempts to gracefully shut down the failed system.
 - a. If the system does not properly shut down after a defined period of time, the correction script kills the process.
3. After it is confirmed that the process is not running anymore, it is started again.
4. If this restart solved the failure, the mechanism ends.
 - a. If the correction attempts are not or only partially successful a failover mechanism should be

triggered, if one was implemented.

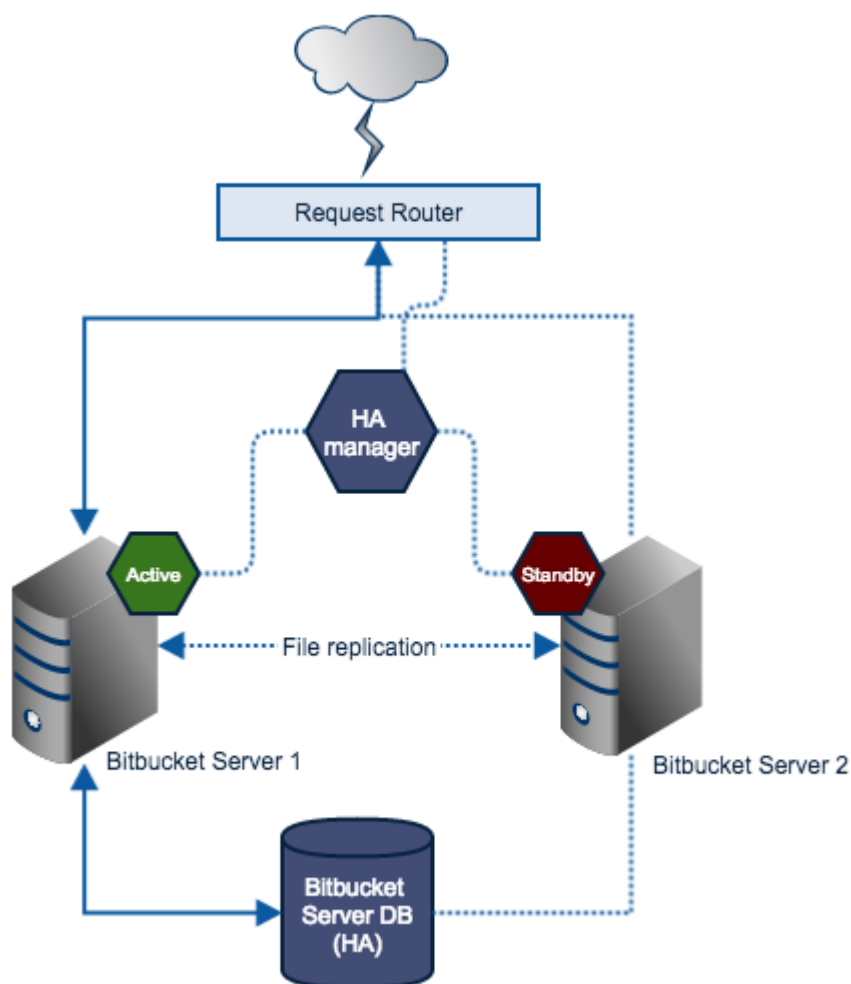
Cold standby

The cold standby (also called Active/Passive) configuration consists of two identical Bitbucket Server instances, where only one server is ever running at a time. The Bitbucket home directory on each of the servers is either a shared (and preferably highly available) network file system or is replicated from the active to the standby Bitbucket Server instance. When a system failure is detected, Bitbucket Server is restarted on the active server. If the system failure persists, a failover mechanism is started that shuts down Bitbucket Server on the active server and starts Bitbucket Server on the standby server, which is promoted to 'active'. At this time, all requests should be routed to the newly active server.

For each component in the chain of high availability measures, there are various implementation alternatives. Although Atlassian does not recommend any particular technology or product, this guide gives options for each step.

System setup

This section describes one possible configuration for how to set up a single instance of Bitbucket Server for high availability.



Component	Description
Request Router	Forwards traffic from users to the active Bitbucket Server instance.

High Availability Manager	Tracks the health of the application servers and decides when to fail over to a standby server and designate it as active. Manages failover mechanisms and sends notifications on system failure.										
Bitbucket Server instance	Each server hosts an identical Bitbucket Server installation (identical versions). Only one server is ever running a Bitbucket Server instance at any one time (known as the active server). All others are considered as standbys. The Bitbucket home directory resides on a replicated or shared file system visible to all application servers (described in more detail below). The Bitbucket home directory must never be modified when the server is in standby mode.										
Bitbucket Server DB	The production database, which should be highly available. How this is achieved is not explored in this document. See the following database vendor-specific information on the HA options available to you: <table border="1"> <thead> <tr> <th>Database</th><th>More Information</th></tr> </thead> <tbody> <tr> <td>Postgres</td><td>http://www.postgresql.org/docs/9.2/static/high-availability.html</td></tr> <tr> <td>MySQL</td><td>http://dev.mysql.com/doc/refman/5.5/en/ha-overview.html</td></tr> <tr> <td>Oracle</td><td>http://www.oracle.com/technetwork/database/features/availability/index.html</td></tr> <tr> <td>SQLServer</td><td>http://technet.microsoft.com/en-us/library/ms190202.aspx</td></tr> </tbody> </table>	Database	More Information	Postgres	http://www.postgresql.org/docs/9.2/static/high-availability.html	MySQL	http://dev.mysql.com/doc/refman/5.5/en/ha-overview.html	Oracle	http://www.oracle.com/technetwork/database/features/availability/index.html	SQLServer	http://technet.microsoft.com/en-us/library/ms190202.aspx
Database	More Information										
Postgres	http://www.postgresql.org/docs/9.2/static/high-availability.html										
MySQL	http://dev.mysql.com/doc/refman/5.5/en/ha-overview.html										
Oracle	http://www.oracle.com/technetwork/database/features/availability/index.html										
SQLServer	http://technet.microsoft.com/en-us/library/ms190202.aspx										

Licensing

Developer licenses can be used for non-production installations of Bitbucket Server deployed on a cold stand-by server. For more information see [developer licenses](#).

Clustering with Bitbucket Data Center

About Bitbucket Data Center

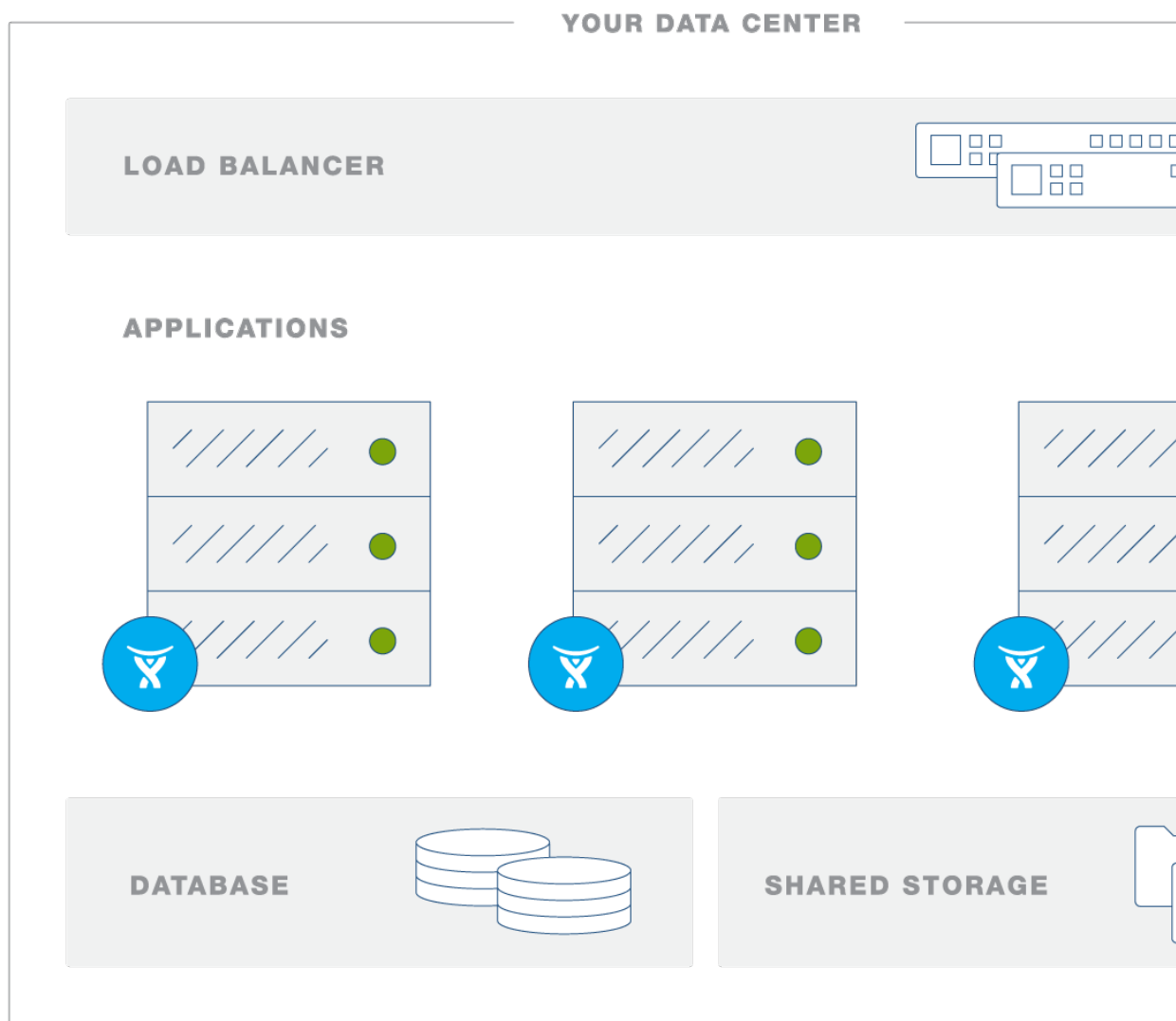
Bitbucket Data Center is the on-premises Git repository management solution for larger enterprises that require high availability and performance at scale. It allows everyone in your organization to easily collaborate on your Git repositories.

Bitbucket Data Center is designed with enterprise scaling and infrastructure flexibility in mind for when you host Bitbucket Server in your own data center. It provides enterprise teams with:

- **Performance at scale:** A cluster of many machines each running Bitbucket Server can handle a greater load than a single machine.
- **High availability:** If one cluster node goes down, then the remaining cluster nodes can continue servicing requests so that users see little or no loss of availability.
- **Instant scalability:** You can rapidly provision extra capacity with almost no downtime.

A look at the architecture

Bitbucket Data Center consists of a cluster of dedicated machines, connected like this:



Load balancer

The load balancer distributes requests from your users to the cluster nodes. If a cluster node goes down, the load balancer immediately detects the failure and automatically directs requests to the other nodes within seconds.

Application nodes

The cluster of Bitbucket Server nodes share the workload of incoming requests. Failure of a cluster node causes virtually no loss of availability for users, because requests are immediately directed to other nodes.

Shared database and storage

Bitbucket Data Center supports the same databases as Bitbucket Server (except for MySQL).

A high-performance shared file system, accessible via NFS, stores repository, attachment and avatar data.

Learn more about Bitbucket Data Center

- [Bitbucket Data Center Performance](#)
- [Failover for Bitbucket Data Center](#)
- [Installing Bitbucket Data Center](#)
- [Adding cluster nodes to Bitbucket Data Center](#)
- [Bitbucket Data Center Add-ons](#)
- [Bitbucket Data Center FAQ](#)
- [Bitbucket Enterprise Resources](#)

Installing Bitbucket Data Center

This page describes how to migrate an existing instance of Bitbucket Server to Bitbucket Data Center. For an overview, see [Bitbucket Data Center](#). If you are installing Bitbucket Server instance go straight to [Getting started](#) instead. We also recommend reading [Using Bitbucket Server in the enterprise](#). If you just want to add another node we suggest you take a look at [Adding cluster nodes to Bitbucket Data Center](#).

This guide assumes that you already have a production instance of Bitbucket Server, and that you are aiming to migrate that to a Bitbucket Data Center instance.

We recommend that you:

- Initiate the purchase of a Bitbucket Data Center license by contacting us at <https://www.atlassian.com/enterprise/contact>.
- Set up and test Bitbucket Data Center in your staging environment, before deploying to a production environment.
- Upgrade Bitbucket Server, and then make a backup of your production instance of Bitbucket Server.
- Restore a copy of this backup into your clustered staging environment.
- Test Bitbucket Data Center with identical data (repositories, users, add-ons) to your production instance.

Regardless of the process you use, please smoke test your Bitbucket Data Center instance every step of the way.

On this page

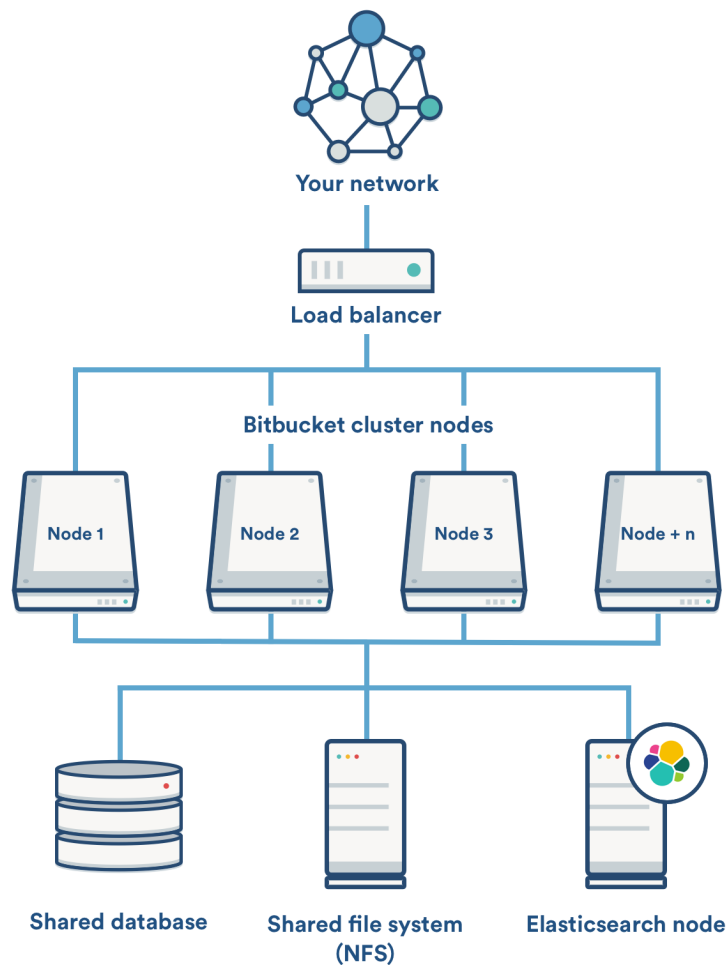
Overview and requirements

1. Upgrade your existing production instance of Bitbucket Server
2. Back up your production instance
3. Provision your shared database
4. Provision your shared file system
5. Provision your Elasticsearch node
6. Provision your cluster nodes
7. Start the first cluster node
8. Install and configure your load balancer
9. Add a new Bitbucket cluster node to the cluster
10. Connect the new Bitbucket cluster node to the load balancer
11. Repeat steps 8 and 9 for each additional cluster node
12. Congratulations!

Overview and requirements

It's worth getting a clear understanding of what you're aiming to achieve, before starting to provision your Bitbucket Data Center.

A Bitbucket Data Center instance consists of a cluster of dedicated machines connected like this:



The URL of the Bitbucket Data Center instance will be the URL of the load balancer, so this is the machine that you will need to assign the name of your Bitbucket Server instance in the DNS.

The remaining machines (Bitbucket cluster nodes, shared database, and shared file system) do not need to be publicly accessible to your users.

Bitbucket cluster nodes

The Bitbucket cluster nodes all run the Bitbucket Data Center web application.

- Each Bitbucket cluster node must be a dedicated machine.
- The machines may be physical or virtual.
- The cluster nodes must be connected in a high speed LAN (that is, they must be physically in the same data center).
- The usual Bitbucket Server [supported platforms](#) requirements, including those for Java and Git, apply to each cluster node.
- The cluster nodes do not all need to be absolutely identical, but for consistent performance we recommend they should be as similar as possible.
- All cluster nodes must run the same version of Bitbucket Data Center.
- All cluster nodes must have synchronized

Load balancer

You can use the load balancer of your choice. Bitbucket Data Center does **not** bundle a load balancer.

- Your load balancer should run on a dedicated machine.
- Your load balancer must have a high-speed LAN connection to the Bitbucket cluster nodes (that is, it must be physically in the same data center).
- Your load balancer must support **both** HTTP mode (for web traffic) **and** TCP mode (for SSH traffic).
- Terminating SSL (HTTPS) at your load balancer and running plain HTTP from the load balancer to Bitbucket Server is highly recommended for performance.
- Your load balancer should support "session

clocks (for example, using NTP) and be configured with the identical timezone.

- Ensure that only permit cluster nodes are allowed to connect to a Bitbucket cluster node's [Hazelcast](#) port, which by default is port 5701, through the use of a firewall and/or network segregation.

Shared database

You must run Bitbucket Data Center on an external database. You can **not** use Bitbucket Server's internal HSQL or H2 database with Bitbucket Data Center.

- The shared database must run on a dedicated machine.
- The shared database must be available to all cluster nodes via a high-speed LAN (it must be in the same physical data center).
- All the usual database vendors in Bitbucket Server's [supported platforms](#) are supported by Bitbucket Data Center, with one exception: we do **not** recommend MySQL at this time due to inherent deadlocks that can occur in this database engine at high load.

affinity" (also known as "sticky sessions").

If you don't have a preference for your load balancer, we provide instructions for haproxy, a popular Open Source software load balancer.

Shared file system

Bitbucket Data Center requires a high performance shared file system such as a SAN, NAS, RAID server, or high-performance file server optimized for I/O.

- The shared file system must run on a dedicated machine.
- The file system must be available to all cluster nodes via a high-speed LAN (it must be in the same physical data center).
- The shared file system should be accessible via NFS as a single mount point.

What is stored on the shared file system?

- configuration files
- data directory, which includes:
 - repositories
 - attachments
 - avatars
- plugins

What is stored locally on each node?

- caches
- logs
- temporary files

1. Upgrade your existing production instance of Bitbucket Server

Begin by upgrading your production Bitbucket Server instance to the latest public release. This is necessary for several reasons:

- The Bitbucket Server database and home directory layout often change in each release of Bitbucket Server. Upgrading first will ensure that your production Bitbucket Server instance and your Bitbucket Data Center instance share identical data format, and you can switch between them at will.
- Any add-ons in your production instance can be verified as compatible with the latest release of Bitbucket Server (or updated if not).
- Any performance or other comparisons between single-node Bitbucket Server and multi-node Bitbucket Data Center will be more meaningful.

Upgrade your Bitbucket Server by following the instructions in the [Bitbucket Server upgrade guide](#).

2. Back up your production instance

Now, take a backup of your production Bitbucket Server instance's database and home directory. For this you can:

- use the [Bitbucket Server backup client](#),
- use your own [DIY backup](#) solution, or
- just stop Bitbucket Server and manually dump your database, and zip up the home directory.

3. Provision your shared database

Set up your shared database server. Note that clustered databases are not yet supported.

See [Connecting Bitbucket Server to an external database](#) for more information.

You **must** ensure your database is configured to allow enough concurrent connections. Bitbucket Server by default uses up to 80 connections **per cluster node**, which can exceed the default connection limit of some databases.

For example, in PostgreSQL the default limit is usually 100 connections. If you use PostgreSQL, you may need to edit your `postgresql.conf` file, to increase the value of `max_connections`, and restart PostgreSQL.

We do **not** support MySQL for Bitbucket Data Center at this time due to inherent deadlocks that can occur in this database engine at high load. If you currently use MySQL, you should migrate your data to another supported database (such as PostgreSQL) before upgrading your Bitbucket Server instance to Bitbucket Data Center. You can migrate databases (on a standalone Bitbucket Server instance) using the Migrate database feature in Bitbucket Server's Administration pages, or by using the [Bitbucket Server backup client](#).

4. Provision your shared file system

Set up your shared file server.

See [Bitbucket Data Center FAQ](#) for performance guidelines when using NFS.

You **must** ensure your shared file system server is configured with enough NFS server processes.

For example, some versions of RedHat Enterprise Linux and CentOS have a default of 8 server processes. If you use one of these systems, you may need to edit your `/etc/sysconfig/nfs` file, increase the value of `RPCNFSDCOUNT`, and restart the `nfs` service.

You **must** ensure your shared file system server has the NFS lock service **enabled**. For example:

- In some versions of Ubuntu Linux you must ensure that the `portmap` and `dbus` services are enabled for the NFS `lockd` to function.
- In some versions of RedHat Enterprise Linux and CentOS, you must install the `nfs-utils` and `nfs-utils-lib` packages, and ensure the `rpcbind` and `nfslock` services are running.

Create a Bitbucket Server user account (recommended name `atlbucket`) on the shared file system server to own everything in the Bitbucket Server shared home directory. This user account must have the same UID on all cluster nodes and the shared file system server. In a fresh Linux install the UID of a newly created account is typically 1001, but in general there is no guarantee that this UID will be free on every Linux system. Choose a UID for `atlbucket` that's free on all your cluster nodes and the shared file system server, and substitute this for 1001 in the following command:

```
sudo useradd -c "Atlassian Bitbucket" -u 1001 atlbucket
```

You **must** ensure that the `atlbucket` user has the same UID on all cluster nodes and the shared file system server.

Then restore the backup you have taken in step 2 into the new shared database and shared home directory.

Only the shared directory in the [Bitbucket Server home directory](#) needs to be restored into the shared home

directory. The remaining directories (bin, caches, export, lib, log, plugins, and tmp) contain only caches and temporary files, and do not need to be restored.

You **must** ensure that the user running Bitbucket Server (usually `atlassian`) is able to read and write everything in the Bitbucket home directory, both the node-local part and the shared part (in NFS). The easiest way to do this is to ensure that:

1. `atlassian` owns all files and directories in the Bitbucket home directory,
2. `atlassian` has the recommended umask of `0027`, and
3. `atlassian` has the same UID on all machines.

Do **not** run Bitbucket Server as root. Many NFS servers squash accesses by root to another user.

5. Provision your Elasticsearch node

Elasticsearch provides search functionality for Bitbucket Server. Read more about [Elasticsearch](#).

Set up your Elasticsearch server, following the instructions in [Install and configure a remote Elasticsearch instance](#).

6. Provision your cluster nodes

1. We highly recommend provisioning cluster nodes using an automated configuration management tool such as Chef, Puppet, or Vagrant, or by spinning up identical virtual machine snapshots.
2. On each cluster node, mount the shared home directory as `${BITBUCKET_HOME}/shared`. For example, suppose your Bitbucket home directory is `/var/atlassian/application-data/bitbucket`, and your shared home directory is available as an NFS export called `bitbucket-san:/bitbucket-shared`. Add the following line to `/etc/fstab` on each cluster node:

```
/etc/fstab
bitbucket-san:/bitbucket-shared
/var/atlassian/application-data/bitbucket/shared nfs
nfsvers=3, lookupcache=pos, noatime, intr, rsize=32768, wsize=32768 0
0
```

Only the `${BITBUCKET_HOME}/shared` directory should be shared between cluster nodes. All other directories, including `${BITBUCKET_HOME}`, should be node-local (that is, private to each node).

Bitbucket Data Center checks during startup that `${BITBUCKET_HOME}` is node local and `${BITBUCKET_HOME}/shared` is shared, and will fail to form a cluster if this is not true.

Your shared file system **must** provide sufficient consistency for Bitbucket Server and Git.

Linux NFS clients require the `lookupcache=pos` mount option to be specified for proper consistency.

NFSv4 may have issues in Linux kernels from about version 3.2 to 3.8 inclusive. The issues may cause very high load average, processes hanging in "uninterruptible sleep", and in some cases may require rebooting the machine. We recommend using NFSv3 unless you are 100% sure that you know what you're doing and your operating system is free from such issues.

Linux NFS clients should use the `nfsvers=3` mount option to force NFSv3.

Then mount it:

```
mkdir -p /var/atlassian/application-data/bitbucket/shared
sudo mount -a
```

3. Ensure all your cluster nodes have synchronized clocks and identical timezone configuration. For example, in RedHat Enterprise Linux or CentOS:

```
sudo yum install ntp
sudo service ntpd start
sudo tzselect
```

In Ubuntu Linux:

```
sudo apt-get install ntp
sudo service ntp start
sudo dpkg-reconfigure tzdata
```

For other operating systems, consult your system documentation.

The system clocks on your cluster nodes **must** remain reasonably synchronized (say, to within a few seconds or less). If your system clocks drift excessively or undergo abrupt "jumps" of minutes or more, then cluster nodes may log warnings, become slow, or in extreme cases become unresponsive and require restarting. You should run the NTP service on all your cluster nodes with identical configuration, and never manually tamper with the system clock on a cluster node while Bitbucket Data Center is running.

4. Download the latest Bitbucket Data Center distribution from <https://www.atlassian.com/software/bitbucket/download>, and install Bitbucket Server as normal on all the cluster nodes. See [Getting started](#).

7. Start the first cluster node

Edit the file `${BITBUCKET_HOME}/shared/bitbucket.properties`, and add the following lines:

```
# Use multicast to discover cluster nodes (recommended).
hazelcast.network.multicast=true

# If your network does not support multicast, you may uncomment the
# following lines and substitute
# the IP addresses of some or all of your cluster nodes. (Not all of
# the cluster nodes have to be
# listed here but at least one of them has to be active when a new
# node joins.)
#hazelcast.network.tcpip=true
#hazelcast.network.tcpip.members=192.168.0.1:5701,192.168.0.2:5701,19
2.168.0.3:5701

# The following should uniquely identify your cluster on the LAN.
hazelcast.group.name=your-bitbucket-cluster
hazelcast.group.password=your-bitbucket-cluster
```

Using multicast to discover cluster nodes (`hazelcast.network.multicast=true`) is recommended, but requires all your cluster nodes to be accessible to each other via a multicast-enabled network. If your network does not support multicast then you can set `hazelcast.network.multicast=false`, `hazelcast.network.tcpip=true`, and `hazelcast.network.tcpip.members` to a comma-separated list of cluster nodes instead. Only enable **one** of `hazelcast.network.tcpip` or `hazelcast.network.multicast`, not both!

Choose a name for `hazelcast.group.name` and `hazelcast.group.password` that uniquely identifies the cluster on your LAN. If you have more than one cluster on the same LAN (for example, other Bitbucket Data Center instances or other products based on similar technology such as Confluence Data Center) then you **must** assign each cluster a distinct name, to prevent them from attempting to join together into a "super cluster".

Then start Bitbucket Server. See [Starting and stopping Bitbucket Server](#).

Then go to `http://<bitbucket>:7990/admin/license`, and install the Bitbucket Data Center license you were issued. Restart Bitbucket Server for the change to take effect. If you need a Bitbucket Data Center license, [please contact us](#)!

8. Install and configure your load balancer

You can use the load balancer of your choice, either hardware or software. Bitbucket Data Center does **not** bundle a load balancer.

Your load balancer must proxy three protocols:

Protocol	Typical port on the load balancer	Typical port on the Bitbucket cluster nodes	Notes
HTTP	80	7990	HTTP mode. Session affinity ("sticky sessions") should be enabled using the 52-character JSESSIONID cookie.
HTTPS	443	7990	HTTP mode. Terminating SSL at the load balancer and running plain HTTP to the Bitbucket cluster nodes is highly recommended.
SSH	7999	7999	TCP mode.

For best performance, your load balancer should support session affinity ("sticky sessions") using the JSESSIONID cookie. By default, Bitbucket Data Center assumes that your load balancer always directs each user's requests to the same cluster node. If it does not, users may be unexpectedly logged out or lose other information that may be stored in their HTTP session.

Bitbucket Data Center also provides a property `hazelcast.http.sessions` that can be set in `#{BITBUCKET_HOME}/shared/bitbucket.properties` that provides finer control over HTTP session management. This property can be set to one of the following values:

- `local` (the default): HTTP sessions are managed per node. When used in a cluster, the load balancer **must** have session affinity ("sticky sessions") enabled. If a node fails or is shut down, users that were assigned to that node may need to log in again.
- `sticky`: HTTP sessions are distributed across the cluster with a load balancer configured to use session affinity ("sticky sessions"). If a node fails or is shut down, users should not have to log in again. In this configuration, session management is optimized for sticky sessions and will not perform certain cleanup tasks for better performance.
- `replicated`: HTTP sessions are distributed across the cluster. If a node fails or is shut down, users should not have to log in again. The load balancer does not need to be configured for session affinity ("sticky sessions"), but performance is likely to be better if it is.

Both the `sticky` and `replicated` options come with some performance penalty, which can be substantial if session data is used heavily (for example, in custom plugins). For best performance, `local` (the default) is recommended.

When choosing a load balancer, it must support the HTTP, HTTPS, and TCP protocols. Note that:

- Apache does **not** support TCP mode load balancing.
- HAProxy versions older than 1.5.0 do **not** support HTTPS.

If your load balancer supports health checks of the cluster nodes, configure it to perform a periodic HTTP GET of `http://<bitbucket>:7990/status`, where `<bitbucket>` is the cluster node's name or IP address. This returns one of two HTTP status codes:

- 200 OK
- 500 Internal Server Error

If a cluster node does not return 200 OK within a reasonable amount of time, the load balancer should not direct any traffic to it.

You should then be able to navigate to `http://<load-balancer>/`, where `<load-balancer>` is your load balancer's name or IP address. This should take you to your Bitbucket Server front page.

Example: HAProxy load balancer

If you don't have a particular preference or policy for load balancers, you can use HAProxy which is a popular Open Source software load balancer.

If you choose HAProxy, you **must** use a minimum version of 1.5.0. Earlier versions of HAProxy do not support HTTPS.

Here is an example `haproxy.cfg` configuration file (typically found in the location `/etc/haproxy/haproxy.cfg`). This assumes:

- Your Bitbucket cluster node is at address 192.168.0.1, listening on the default ports 7990 (HTTP) and 7999 (SSH).
- You have a valid SSL certificate at `/etc/cert.pem`.

```
haproxy.cfg

global
    pidfile      /var/run/haproxy.pid
    maxconn      4000
    user         haproxy
    group        haproxy
    daemon

defaults
    log                  global
    option               dontlognull
    option               redispatch
    retries              3
    timeout http-request 10s
    timeout queue        1m
    timeout connect      10s
    timeout client       1m
    timeout server       1m
    timeout http-keep-alive 10s
    timeout check        10s
    maxconn              3000
    tune.ssl.default-dh-param 1024
    errorfile            408 /dev/null # Workaround for Chrome
35-36 bug. See
http://blog.haproxy.com/2014/05/26/haproxy-and-http-errors-408-in-chr
```

```
ome/

frontend bitbucket_http_frontend
    bind *:80
    bind *:443 ssl crt /etc/cert.pem ciphers
    RC4-SHA:AES128-SHA:AES256-SHA
    default_backend bitbucket_http_backend

backend bitbucket_http_backend
    mode http
    option httplog
    option httpchk GET /status
    option forwardfor
    option http-server-close
    appsession JSESSIONID len 52 timeout 1h
    balance roundrobin
    cookie JSESSIONID prefix
    stick-table type string len 52 size 5M expire 30m
    stick store-response set-cookie(JSESSIONID)
    stick on cookie(JSESSIONID)
    server bitbucket01 192.168.0.1:7990 check inter 10000 rise 2 fall
5
    #server bitbucket02 192.168.0.2:7990 check inter 10000 rise 2
fall 5
    # The following "backup" servers are just here to show the
startup page when all nodes are starting up
    server backup01 192.168.0.1:7990 backup
    #server backup02 192.168.0.2:7990 backup

frontend bitbucket_ssh_frontend
    bind *:7999
    default_backend bitbucket_ssh_backend
    timeout client 15m
    maxconn 50

backend bitbucket_ssh_backend
    mode tcp
    balance roundrobin
    server bitbucket01 192.168.0.1:7999 check port 7999
    #server bitbucket02 192.168.0.2:7999 check port 7999
    timeout server 15m

listen admin
    mode http
```

```
bind *:8090
stats enable
stats uri /
```

Review the contents of the `haproxy.cfg` file carefully, and customize it for your environment. See <http://www.haproxy.org/> for more information about installing and configuring haproxy.

Once you have configured the `haproxy.cfg` file, start the haproxy service.

```
sudo service haproxy start
```

You can also monitor the health of your cluster by navigating to HAProxy's statistics page at `http://<load-balancer>:8090/`. You should see a page similar to this:

HAProxy version 1.5.0, released 2014/06/19

Statistics Report for pid 28972

> General process information

pid = 28972 (process #1, nbproc = 1)
uptime = 0d 20h34m58s
system limits: memmax = unlimited; ulimit-n = 8019
maxsock = 8019; maxconn = 4000; maxpipes = 0
current conns = 3; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/17; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance

backup UP
backup UP, going down
backup DOWN, going up
not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

Scope :

[Hide 'DOWN' servers](#)
[Refresh now](#)
[CSV export](#)

External resources:

[Primary site](#)
[Updates \(v1.5\)](#)
[Online manual](#)

stash_http_frontend

	Queue	Session rate			Sessions			Bytes			Denied	Errors			Warnings	Status	Server												
		Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last		In	Out	Req			Resp	Req	Conn	Resp	Retr	Redis	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme
Frontend		0	12	-	1	22	3 000	60 503			45 043 864	381 514 405	0	0	127					OPEN									

stash_cluster_http_backend

	Queue	Session rate			Sessions			Bytes			Denied	Errors			Warnings	Status	Server										
		Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last		In	Out	Req			Resp	Req	Conn	Resp	Retr	Redis	LastChk	Wght	Act	Bck	Chk
stash-app-1	0	0	-	0	25	0	10	-	49 705	45 103	3s	24 674 486	194 031 639	0	0	0	0	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	9	0	0s	-
stash-app-2	0	0	-	0	38	0	10	-	48 129	45 103	3s	20 369 378	187 456 692	0	0	2	0	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	10	0	0s	-
Backend	0	0	-	0	36	1	22	300	137 605	90 206	3s	45 043 864	381 514 405	0	0	0	2	0	20h34m UP		2	2	0		0	0s	

stash_ssh_frontend

	Queue	Session rate			Sessions			Bytes			Denied	Errors			Warnings	Status	Server											
		Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last		In	Out	Req			Resp	Req	Conn	Resp	Retr	Redis	LastChk	Wght	Act	Bck	Chk	Dwn
Frontend		1	15	-	0	11	50	145 340			391 467 702	1 493 870 815	0	0	0					OPEN								

stash_cluster_ssh_backend

	Queue	Session rate			Sessions			Bytes			Denied	Errors			Warnings	Status	Server										
		Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last		In	Out	Req			Resp	Req	Conn	Resp	Retr	Redis	LastChk	Wght	Act	Bck	Chk
stash-app-1	0	0	-	0	7	0	7	-	54 411	54 411	9s	195 799 631	761 383 500	0	0	5	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
stash-app-2	0	0	-	0	7	0	6	-	54 410	54 410	13s	195 668 071	732 487 315	0	0	6	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0	-	0	11	5	145 340	108 821	9s	391 467 702	1 493 870 815	0	0	0	11	0	0	20h34m UP		2	2	0		0	0s		

9. Add a new Bitbucket cluster node to the cluster

Go to a new cluster node, and start Bitbucket Server. See [Starting and stopping Bitbucket Server](#).

Once Bitbucket Server has started, go to `http://<load-balancer>/admin/clustering`. You should see a page similar to this:

Clustering

A cluster of multiple Bitbucket nodes provides high availability and performance at scale. [Learn more about clustering.](#)

	Node ID	Node name	Cluster Address
	1f781654-e246-4737-84ea-6b11f59255d2	bitbucket-app-2	172.25.1.139:5701
	ca9456d1-6815-410b-b37c-3d5763a6bb88	bitbucket-app-1	172.25.1.12:5701

 New nodes can join the cluster without downtime. [Learn how to add a node.](#)

Verify that the new node you have started up has successfully joined the cluster. If it does not, please check your network configuration and the `${BITBUCKET_HOME}/log/atlassian-bitbucket.log` files on all nodes. If you are unable to find a reason for the node failing to join successfully, please contact [Atlassian Support](#).

10. Connect the new Bitbucket cluster node to the load balancer

If you are using your own hardware or software load balancer, consult your vendor's documentation on how to add the new Bitbucket cluster node to the load balancer.

If you are using HAProxy, just uncomment the lines

```
server bitbucket02 192.168.0.2:7990 check inter 10000 rise 2 fall 5
```

```
server bitbucket02 192.168.0.2:7999 check port 7999
```

in your `haproxy.cfg` file and restart haproxy:

```
sudo service haproxy restart
```

Verify that the new node is in the cluster and receiving requests by checking the logs on each node to ensure both are receiving traffic and also check that updates done on one node are visible on the other.

11. Repeat steps 8 and 9 for each additional cluster node

12. Congratulations!

You have now set up a clustered instance of Bitbucket Data Center! We are very interested in hearing your feedback on this process – please [contact us](#)!

For any issues please raise a [support ticket](#) and mention that you are following the 'Installing Bitbucket Data Center' page.

Please see [Using Bitbucket Server in the enterprise](#) for information about using Bitbucket Server in a production environment.

Adding cluster nodes to Bitbucket Data Center

This page...

... describes how to add another cluster node to an existing instance of Bitbucket Data Center.

If you are moving to Bitbucket Data Center...

... go straight to [Installing Bitbucket Data Center](#), instead.

If you are new to Bitbucket Data Center...

... we suggest you take a look at [Clustering with Bitbucket Data Center](#).

Provisioning a cluster node

You can rapidly scale the capacity of Bitbucket Data Center, with very little downtime, by provisioning extra cluster nodes.

We highly recommend provisioning cluster nodes using an automated configuration management tool such as Chef, Puppet, or Vagrant, or by spinning up identical virtual machine snapshots.

The Bitbucket cluster nodes all run the Bitbucket Data Center web application:

- Each Bitbucket cluster node must be a dedicated machine.
- The machines may be physical or virtual.
- The cluster nodes must be connected in a high speed LAN (that is, they must be physically in the same data center).
- The usual Bitbucket Server [supported platforms](#) requirements, including those for Java and Git, apply to each cluster node.
- The cluster nodes do not all need to be absolutely identical, but for consistent performance we recommend they should be as similar as possible.
- All cluster nodes must run the same version of Bitbucket Data Center.
- All cluster nodes must have synchronized clocks (for example, using NTP) and be configured with the identical timezone.

Provisioning a cluster node involves the following steps:

1. [Mount the shared home directory on the node](#)
2. [Install Bitbucket Data Center on the node](#)
3. [Add the node to the cluster](#)
4. [Connect the node to the load balancer](#)

1. Mount the shared home directory on the node

The Bitbucket Data Center makes use of a shared file system that lives on a dedicated machine and is accessible using NFS. See [Installing Bitbucket Data Center](#) for more information.

Mount the shared home directory as `${BITBUCKET_HOME}/shared`. For example, suppose your Bitbucket home directory is `/var/atlassian/application-data/bitbucket`, and your shared home directory is available as an NFS export called `bitbucket-san:/bitbucket-shared`. Add the following line to `/etc/fstab` on the cluster node:

```
/etc/fstab
bitbucket-san:/bitbucket-shared
/var/atlassian/application-data/bitbucket/shared nfs
nfsvers=3,lookupcache=pos,noatime,intr,rsiz=32768,wsiz=32768 0 0
```

Then mount it:

```
mkdir -p /var/atlassian/application-data/bitbucket/shared
sudo mount -a
```


2. Install Bitbucket Data Center on the node

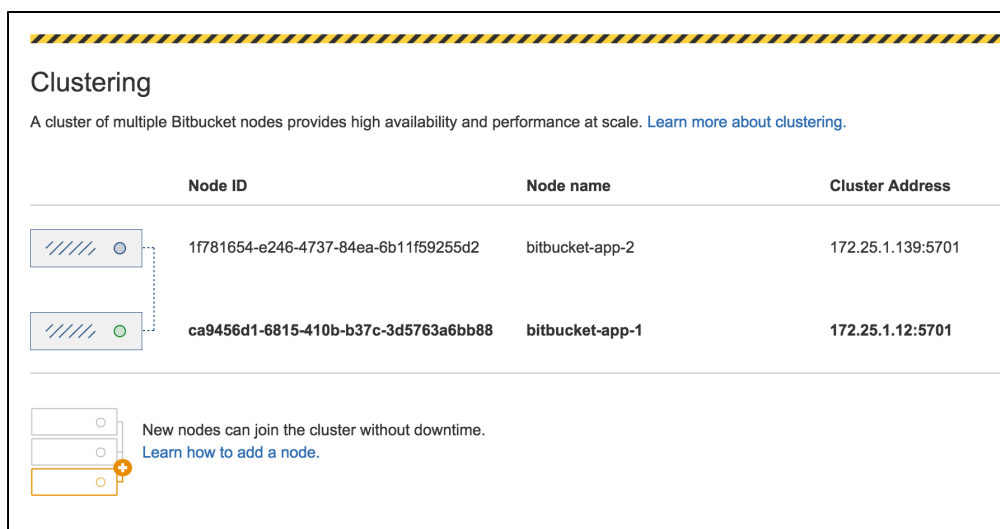
Download the latest Bitbucket Data Center distribution from <https://www.atlassian.com/software/bitbucket/download>, and install Bitbucket Server as normal on the cluster node. See [Getting started](#).

3. Add the node to the cluster

Start Bitbucket Server on the new node. See [Starting and stopping Bitbucket Server](#). You can optionally give the node a persistent, human readable name by setting a `node.name` system property under `JVM_SUPPORT_RECOMMENDED_ARGS` in `setenv.sh`. For example:

```
-Dcluster.node.name=bitbucket-1
```

Once Bitbucket Server has started, go to `http://<load-balancer>/admin/clustering`. You should see the new node listed, similarly to this:



The screenshot shows the 'Clustering' admin page. It has a header with a yellow and black striped warning bar. Below the header, there's a title 'Clustering' and a description: 'A cluster of multiple Bitbucket nodes provides high availability and performance at scale. [Learn more about clustering.](#)'. Below this is a table with three columns: 'Node ID', 'Node name', and 'Cluster Address'. There are two nodes listed: 'bitbucket-app-2' and 'bitbucket-app-1'. To the left of the table, there are two server icons with status indicators (a red 'X' and a green circle). Below the table, there's a section with a server icon and a plus sign, containing the text: 'New nodes can join the cluster without downtime. [Learn how to add a node.](#)'

Node ID	Node name	Cluster Address
1f781654-e246-4737-84ea-6b11f59255d2	bitbucket-app-2	172.25.1.139:5701
ca9456d1-6815-410b-b37c-3d5763a6bb88	bitbucket-app-1	172.25.1.12:5701

New nodes can join the cluster without downtime. [Learn how to add a node.](#)

Verify that the new node you have started up has successfully joined the cluster. If it does not, please check your network configuration and the `${BITBUCKET_HOME}/log/atlassian-bitbucket.log` files on all nodes. If you are unable to find a reason for the node failing to join successfully, please contact [Atlassian Support](#).

4. Connect the node to the load balancer

The Bitbucket Data Center makes use of a load balancer to distribute requests from your users to the cluster nodes. If a cluster node goes down, the load balancer immediately detects the failure and automatically directs requests to the other nodes within seconds. See [Installing Bitbucket Data Center](#) for more information.

If you are using a hardware or software load balancer other than HAProxy, consult your vendor's documentation on how to add the new Bitbucket cluster node to the load balancer.

If you are using HAProxy, simply add the following lines to your `haproxy.cfg` file:

```
# In the backend bitbucket_http_backend section, add:
server bitbucket<xx> 192.168.0.<x>:7990 check inter 10000 rise 2 fall
5
```



```
# In the backend bitbucket_ssh_backend section, add:
server bitbucket<xx> 192.168.0.<x>:7999 check port 7999
```

where the values for <x> and <xx> don't conflict with an existing node.

Now restart HAProxy :

```
sudo service haproxy restart
```

Verify that the new node is in the cluster and receiving requests by checking the logs on each node to ensure that all are receiving traffic. Also check that updates done on one node are visible on the other nodes.

You can monitor the health of your cluster by navigating to HAProxy's statistics page at `http://<load-balancer>:8090/`. You should see a page similar to this:

HAProxy version 1.5.0, released 2014/06/19

Statistics Report for pid 28972

> General process information

For any issues please raise a [support ticket](#) and mention that you are following the 'Adding cluster nodes to Bitbucket Data Center' page.

Enabling JMX counters for performance monitoring

This article describes how to expose JMX MBeans within Bitbucket Server for monitoring with a JMX client.

What is JMX?

JMX (Java Management eXtensions) is a technology for monitoring and managing Java applications. JMX uses objects called MBeans (Managed Beans) to expose data and resources from your application.

Why would I want to enable JMX monitoring within Bitbucket Server?

For large instances of Bitbucket Server or Bitbucket Data Center, enabling JMX allows you to more easily monitor the consumption of application resources. This enables you to make better decisions about how to maintain and optimize machine resources.

On this page

- [What is JMX?](#)
- [Expose JMX MBeans within Bitbucket Server](#)
- [Expose JMX MBeans when Bitbucket Server is run as a Windows service](#)
- [Verify JMX is configured correctly](#)

Related reading

- [Understanding JMX \(Oracle\)](#)

What can I monitor with JMX?

It is possible to monitor various statistics using JMX counters within Bitbucket Server. Below are some examples of some statistics that can be monitored.

Bitbucket Server repository statistics

- Total number of projects
- Total number of repositories
- Git pushes and pulls
- Various thread pools and attributes

Thread pools

Thread pool	Description
IoPumpThreadPool	Threads that handle external process IO
ScheduledThreadPool	Thread pool that takes care of several miscellaneous scheduled tasks
EventThreadPool	Threads that dispatch events to @EventListenermethods

Thread pool attributes

Name	Description
ActiveCount	Returns the approximate number of threads that are actively executing tasks.
MaximumPoolSize	Returns the maximum allowed number of threads.
PoolSize	Returns the current number of threads in the pool.

QueueLength	The number of tasks awaiting execution by the thread pool.
LargestPoolSize	The largest number of threads that have ever been simultaneously in the pool.
CompletedTaskCount	The approximate total number of tasks that have completed execution. Because the states of tasks and threads may change dynamically during computation, the returned value is only an approximation, but one that does not ever decrease across successive calls.

Ticket statistics

Bitbucket server uses 'tickets' as a mechanism for creating back-pressure to prevent the system from being overloaded with requests. There are two types of tickets used by Bitbucket server, hosting tickets and command tickets.

Hosting tickets: Limits the number of SCM hosting operations, meaning pushes and pulls over HTTP or SSH, which may be running concurrently.

Command tickets: Limits the number of SCM commands, such as: `git diff`, `git blame`, or `git rev-list`, which may be running concurrently.

Bitbucket server supports the following metrics for each ticket type.

Name	Description
Available	The number of tickets available for acquisition (lower number means higher load)
LastRejection	The timestamp of the last rejected ticket, or null if no tickets have been rejected
Name	The name of the ticket bucket either 'scm-command' or 'scm-hosting'
OldestQueuedRequest	The timestamp at which the oldest queued request started waiting, or null if there are no queued requests
QueuedRequests	The number of requests currently waiting for an available ticket
Total	The maximum number of tickets that can be acquired concurrently before back-pressure is applied
Used	The number of tickets that have been acquired (higher number means higher load)

Interesting 3rd party library attributes

Bitbucket Server exposes the JMX attributes from number of third party libraries. Listed below is a sample of the attributes that are particularly interesting from an operations perspective.

HikariCP - `com.zaxxer.hikari.Pool` (bitbucket)

Name	Description
ActiveConnections	Active Connections (in use)
IdleConnections	Idle Connection count
ThreadsAwaitingConnection	The number of threads waiting for a connection (when all available connections are in use)
TotalConnections	Total Connections

Hibernate - `org.hibernate.core/org.hibernate.stat.Statistics/org.hibernate.stat.internal.ConcurrentStatisticsImpl/bitbucket.core`

Name	Description

QueryCacheHitCount	Global number of cached queries successfully retrieved from cache
QueryCacheMissCount	Global number of cached queries <i>not</i> found in cache
SecondLevelCacheHitCount	Global number of cacheable entities/collections successfully retrieved from the cache
SecondLevelCacheMissCount	Global number of cacheable entities/collections <i>not</i> found in the cache and loaded from the database

Expose JMX MBeans within Bitbucket Server

To enable Bitbucket Server to publish specific statistics using JMX you need to

1. Modify the `bitbucket.properties` file.
2. Create a JMX password file for secure access to JMX monitoring.
3. Modify the `setenv.sh` file to enable Bitbucket Server to expose JMX Mbeans.

These changes will not take effect until Bitbucket Server has been restarted.

Modify the `bitbucket.properties` file

To modify (or create) the `bitbucket.properties` file

1. Create the `bitbucket.properties` file, in the shared folder of your [Bitbucket home directory](#). Take care to use the standard format for Java properties files.

The `bitbucket.properties` file is created automatically if you previously performed a [data base migration](#).

2. Add this property to the file.

```
jmx.enabled=true
```

Set up the JMX password file

To set up a JMX password file to secure access to JMX monitoring

1. Create a file named `jmx.access`.

This file will contain password information. Ensure the file is only readable by the secure user Bitbucket Server will run under. However, note that if the Bitbucket Server user cannot read the file Bitbucket Server will fail to start.

2. Edit the `jmx.access` file to include this property and save the file.

```
monitorRole=<password>
```

If you wish to use a username other than `monitorRole` or `controlRole` you will need to modify the `jmxremote.access` file that is bundled with bitbucket in the `.install4j/jre.bundle/Content s/Home/jre/lib/management/` directory.

Modify the Bitbucket Server environment file

To modify the `setenv.sh` (for Windows `setenv.bat`) files to enable JMX monitoring for Bitbucket Server

1. Within the bin directory, locate the file `setenv.sh` (for Windows `setenv.bat`) and change these properties.

```
JMX_REMOTE_AUTH=password
JMX_REMOTE_PORT=3333
RMI_SERVER_HOSTNAME=-Djava.rmi.server.hostname=<hostname>
JMX_PASSWORD_FILE=<path>/jmx.access
```

2. Restart Bitbucket Server.

Expose JMX MBeans when Bitbucket Server is run as a Windows service

To expose JMX MBeans when Bitbucket Server is run as a Windows service.

1. Stop the Bitbucket Server service.
2. Open the command line prompt and enter.

```
cmd
```

3. Navigate to the Bitbucket Server bin directory.

```
cd <Bitbucket Server Installation dir>\bin
```

4. Run this command.

```
tomcat8w //ES//AtlassianBitbucket Server
```

5. In the window that appears, click on the Java tab to see the list of current startup options. Under "Java Options:" form, input the value

```
-Dcom.sun.management.jmxremote.port=<JMX_REMOTE_PORT>
-Djava.rmi.server.hostname=<hostname>
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.password.file=<JMX_PASSWORD_FILE>
```

Ensure the owner of this password file is the secure user Bitbucket Server will run as. If the Bitbucket Server user cannot read the file, Bitbucket Server will fail to start.

6. Replace the values within the < > characters.

```
JMX_REMOTE_PORT=3333
JMX_PASSWORD_FILE=<path>\jmx.access
```

7. Restart Bitbucket Server Service.

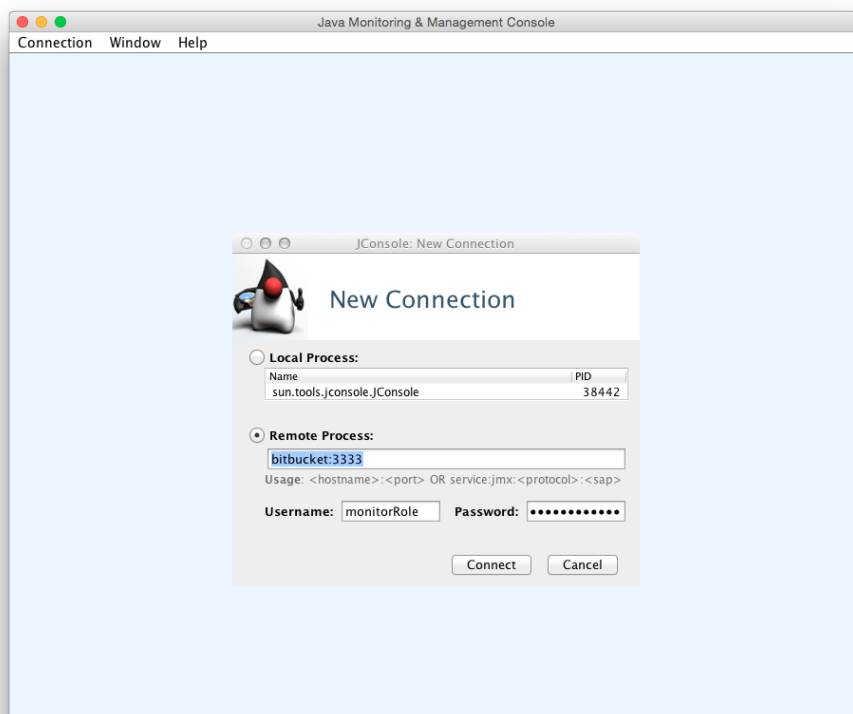
Verify JMX is configured correctly

These steps use JConsole to test that JMX has been configured correctly. JConsole is a utility that ships with

the Oracle JDK.

1. To start the jconsole utility, from a command line prompt enter

```
jconsole
```



2. Create a new JConsole connection with similar connection settings.

bitbucket	the hostname of the instance of Bitbucket Server to monitor
3333	the JMX port number previously configured.
username, password	values configured within the JMX password file <code>jmx.access</code> .

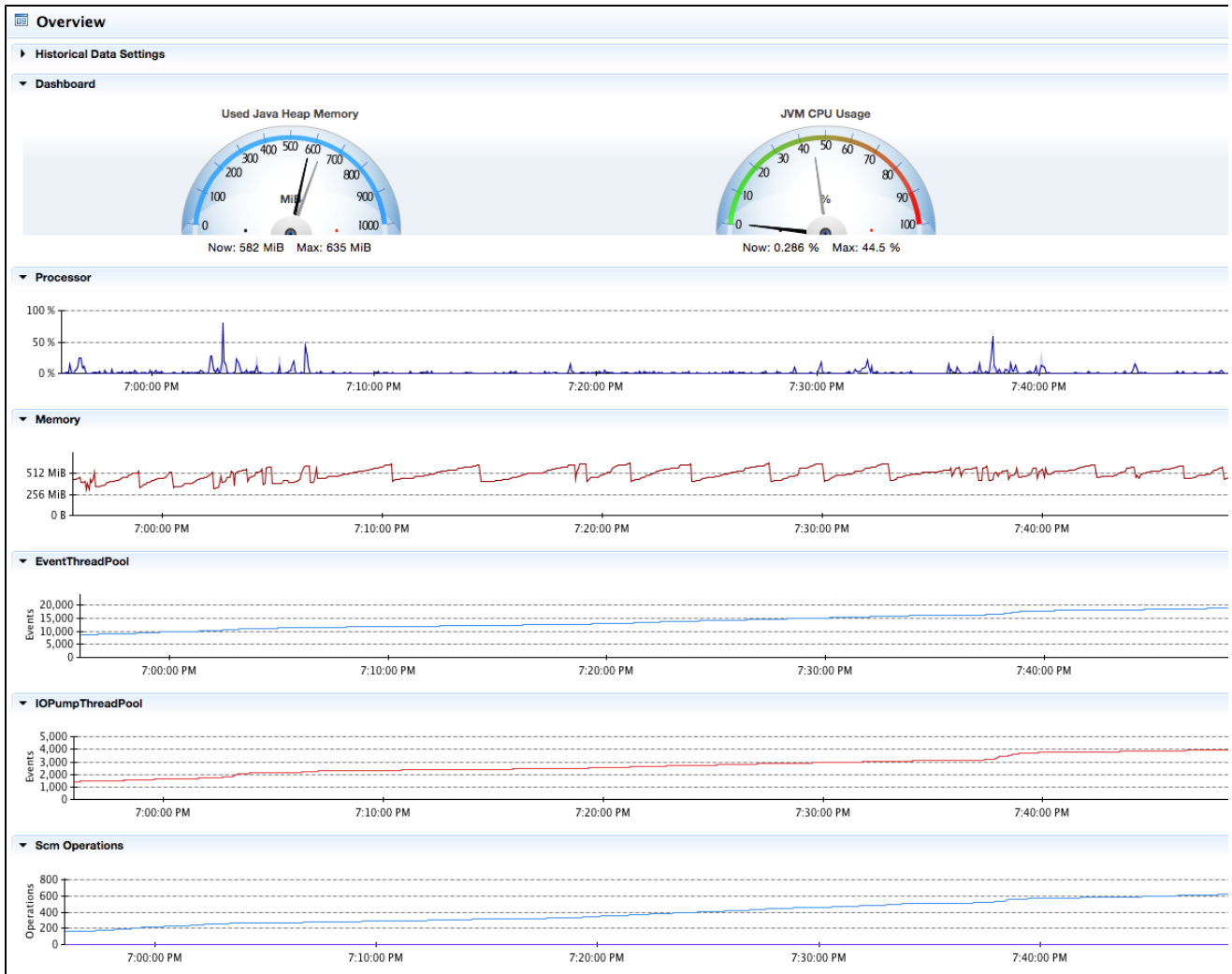
3. Click **Connect**.

When configured correctly, you will see these properties.

com.atlassian.bitbucket	• CommandTickets • HostingTickets • Projects • Repositories • ScmStatistics • Tickets
com.atlassian.bitbucket.thread-pools	• EventThreadPool • IoPumpThreadPool • ScheduledThreadPool

Example performance dashboard

This dashboard was generated using [Java Mission Control](#) that ships with the Oracle JDK (since 1.7u40). See the documentation that comes with your JMX client of choice for more information.



Configuring JMX to use SSL

You can find information about the options for configuring JMX to use SSL in the setenv files. Comprehensive documentation is [available from Oracle](#).

Getting started with Bitbucket Server and AWS

Running Bitbucket Server in the [Amazon Web Services \(AWS\)](#) cloud can give you scalable computing capacity without the need to invest in hardware up front. To this end, Atlassian provides:

- an Amazon Machine Image (AMI) that you can launch in AWS as a "turnkey" deployment of Bitbucket Server, or use as the starting point for customizing your own more complex deployments,
- an Amazon CloudFormation template that automates the process of spinning up a Bitbucket Server instance in EC2, and
- tools and guidelines for backing up, restoring, sizing, and administering your Bitbucket Server instances in AWS.

Running Bitbucket Data Center in AWS is not supported at this time.

Quick Start guide

The simplest way to launch Bitbucket Server in AWS is to use Atlassian's public Amazon CloudFormation template. See [Quick Start with Bitbucket Server and AWS](#).

[Get Started!](#)

Launching Bitbucket Server in AWS manually

For more precise control over the components enabled within the Atlassian Bitbucket Server AMI, including AWS-specific configuration, network and security settings, [Launching Bitbucket Server in AWS manually](#) describes how to launch the AMI by running the EC2 launch wizard.

Performance guidelines

To get the best performance out of your Bitbucket Server deployment in AWS, it's important not to under-provision your instance's CPU, memory, or I/O resources. We provide specific recommendations on choosing AWS EC2 and EBS settings for best performance when running Bitbucket Server in AWS. See [Recommendations for running Bitbucket Server in AWS](#).

Backing up Bitbucket Server in AWS

Atlassian also provides Bitbucket Server DIY Backup utilities that back up and restore your Bitbucket Server instance in AWS using native AWS snapshots. This provides a number of advantages:

- **Performance:** AWS snapshots occur asynchronously resulting in shorter backup downtime for your instances.
- **Durability:** The underlying storage of AWS snapshots is in Amazon S3, which is stored redundantly and with high durability.
- **Availability:** AWS snapshots are available across an entire AWS region, and are available for restore even in the event of an outage affecting an entire Availability Zone (AZ).

To learn more about how to back up and restore a Bitbucket Server instance in AWS, see [Using Bitbucket Server DIY Backup in AWS](#).

The Atlassian Bitbucket Server AMI

The Atlassian Bitbucket Server AMI provides a typical Bitbucket Server deployment in AWS, pre-configured and ready to launch. See [Launching Bitbucket Server in AWS manually](#).

The components bundled in the Atlassian Bitbucket Server AMI are

- Bitbucket Server (either the latest version or version of your choice),
- an external PostgreSQL database,
- nginx as a reverse proxy,
- the Bitbucket Server DIY Backup utilities pre-configured for native AWS snapshots,
- an EBS Volume and Instance Store to hold the data.

Administering Bitbucket Server in AWS

See [Administering Bitbucket Server in AWS](#) for information about performing administration tasks on a Bitbucket Server instance within AWS, including

- configuring variables when launching Bitbucket Server in AWS
- maintaining, resizing, upgrading, migrating, and customizing your Bitbucket Server deployment in AWS
- additional details about the components within the Atlassian Bitbucket Server AMI

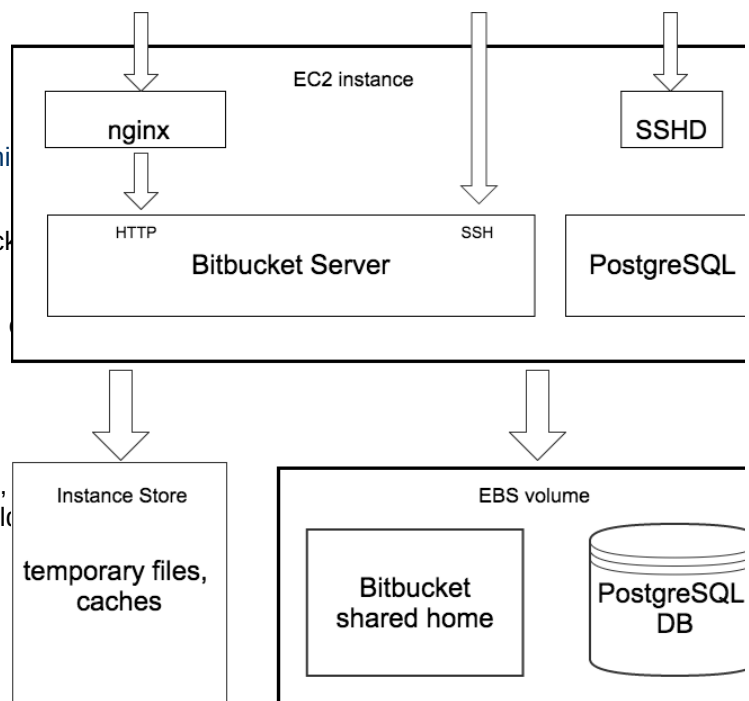
Securing Bitbucket Server within AWS

AWS is accessed over the public Internet, so it is important to apply appropriate security measures when running Bitbucket Server there. See [Best practices for securing Bitbucket Server in AWS](#) for security guidance on a range of security topics, including Amazon Virtual Private Cloud (VPC), Security Groups, and SSL.

Quick Start with Bitbucket Server and AWS

Set up a CloudFormation stack using the Atlassian Bitbucket Server template.

To get started using Bitbucket Server in AWS



1. [Sign in to the AWS Console](#), and go to **Services > CloudFormation**.
2. Click **Create Stack**.
3. Select **Specify an Amazon S3 template URL** field and paste in

```
http://atlassian-software.s3.amazonaws.com/templates/bitbucket/BitbucketServer.template
```

4. When filling in Parameters, make sure you fill in the parameters appropriately for your organization.

Parameters

CidrBlock The IP address range that is permitted to access the instance

EbsOptimized ☒ Enable EBS Optimizations

HomeIops
Home directory IOPS (100 - 20000, only used with Provisioned IOPS). Note: The ratio of IOPS provisioned to the volume size requested can be a maximum of 30; for example, a volume with 3000 IOPS must be at least 100 GiB

HomeSize Home directory storage size, in gibibytes (GiB) (100 - 16384)

HomeVolumeType Home directory volume type

InstanceType EC2 Instance type

KeyName (REQUIRED) The EC2 Key Pair to allow SSH access to the instance

SSLCertificate ☒ Generate a self-signed certificate ☐ Certificate

StashVersion Version of Stash, or 'latest' for the latest public release

Subnet (REQUIRED) Subnet within the selected VPC

VPC (REQUIRED) Virtual Private Cloud

CidrBlock: You can optionally restrict access to your instance to an IP address range in CIDR notation. NOTE: using 0.0.0.0/0 means unrestricted access.

KeyName (REQUIRED): Make sure you have access to the private key file for the EC2 Key Pair you have selected. Without this file, you won't be able to SSH into your instance. See [Creating an EC2 Key Pair](#).

SSLCertificate: You can optionally generate a self-signed SSL certificate, forcing all Web access to your instance to use HTTPS. See [Installing an SSL certificate in your Bitbucket Server instance](#).

VPC and Subnet (REQUIRED): Choose the right public or private VPC for your account, and make sure the Subnet is within the VPC.

See [Securing Bitbucket Server in AWS](#) for more information about these options.

5. Once your CloudFormation stack has finished, select the **Outputs** tab and click the **URL**.

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy
Key					Value		
URL					https://ec2-52-16-79-135.eu-west-1.compute.amazonaws.com		
PublicIp					52.16.79.135		
PrivateIp					172.31.26.245		

What's next?

Now you're ready to configure your Bitbucket Server instance in AWS.

- Complete the [Bitbucket Server Setup Wizard](#), and begin using this like any other instance of Bitbucket Server.
- Review and update your [security settings for AWS](#).
- [Migrate your existing Bitbucket Server instance into AWS](#).
- Be sure to see the rest of the [Administering Bitbucket Server](#) documentation.

Launching Bitbucket Server in AWS manually

This page describes how to launch the Atlassian Bitbucket Server AMI manually, giving you complete control over the components enabled in the AMI and over AWS-specific configuration, network and security settings. If you are just looking for an automated way to spin up Bitbucket Server in AWS, see [Quick Start with Bitbucket Server and AWS](#).

You can launch the Atlassian Bitbucket Server AMI directly from the [AWS Console](#), and running the EC2 launch wizard. See [Launching EC2 Instances](#) for detailed instructions.

On this page

- Finding the Atlassian Bitbucket Server AMI
- Choosing an instance type
- Configure instance details
- Add storage
- Configure your Security Group
- What's next?

Finding the Atlassian Bitbucket Server AMI

You can find the Atlassian Bitbucket Server AMI by clicking **Community AMIs** and searching for **Atlassian Bitbucket 2016.03.30_0432 (HVM)**.

Be sure to use the correct AMI ID for your specific region. The following table lists the AMI ID of the Atlassian Bitbucket Server AMI in each region.

Region Code	Region Name	AMI ID
ap-northeast-1	Asia Pacific (Tokyo)	ami-1026307e
ap-northeast-2	Asia Pacific (Seoul)	ami-18d41d76
ap-southeast-1	Asia Pacific (Singapore)	ami-a131fbc2
ap-southeast-2	Asia Pacific (Sydney)	ami-2e0b284d
eu-central-1	EU (Frankfurt)	ami-7abd5c15
eu-west-1	EU (Ireland)	ami-fb0a8c88
sa-east-1	South America (São Paulo)	ami-c1c24dad
us-east-1	US East (N. Virginia)	ami-e8e6ed82
us-west-1	US West (N. California)	ami-a78ef3c7
us-west-2	US West (Oregon)	ami-a401ebc4

Choosing an instance type

When choosing an EC2 Instance type, see [Recommendations for running Bitbucket Server in AWS](#) for recommended instance sizing.

**Minimum hardware requirements**

The default t2.micro (Free tier eligible), small, and medium instance types do not meet Bitbucket Server's [minimum hardware requirements](#), and are not supported for production deployments. See [Recommendations for running Bitbucket Server in AWS](#) for the EC2 instance types supported by Bitbucket Server.

Configure instance details

When configuring your EC2 instance these are some important details to consider.

IAM Role

It is recommended to launch your instance with an Identity and Access Management (IAM) Role that allows native AWS DIY Backup to run without explicit credentials. See [IAM Roles for Amazon EC2](#) for more information.

From **Step 3: Configure Instance Details** of the EC2 Launch wizard, you can create a new IAM Role by clicking **Create new IAM role**. The role should contain at least the following policy:

```
{
  "Statement": [
    {
      "Resource": [
        "*"
      ],
      "Action": [
        "ec2:AttachVolume",
        "ec2:CreateSnapshot",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:DescribeSnapshots",
        "ec2:DescribeVolumes",
        "ec2:DetachVolume"
      ],
      "Effect": "Allow"
    }
  ],
  "Version": "2012-10-17"
}
```

**IAM Role must be configured at launch time**

An IAM Role can only be configured for your EC2 instance **during initial launch**. You cannot associate an IAM role with a running EC2 instance **after** launch. See [IAM Roles](#) for more information.

Advanced Details

The Atlassian Bitbucket Server AMI can be configured in a number of different ways at launch time:

- The built-in PostgreSQL and Nginx components (enabled by default) can be disabled,
- Self-signed SSL certificate generation (disabled by default) can be enabled.

You can control these options supplying User Data to your instance under **Advanced Details** in **Step 3: Configure Instance Details** of the EC2 launch wizard. All user-configurable behavior in the Atlassian

Bitbucket Server AMI can be controlled by creating a file `/etc/atlassian` containing shell variable definitions. On first boot, the Atlassian Bitbucket Server AMI will source the file `/etc/atlassian` (if it exists), allowing its built-in default variable definitions to be overridden.

For example, to enable self-signed SSL certificate generation (and force all Web access to Bitbucket Server to use HTTPS), you can add User Data (**As text**) as follows:

```
#!/bin/bash
echo "ATL_SSL_SELF_CERT_ENABLED=true" >>/etc/atlassian
```

For a complete list of variables that can be overridden in User Data at launch time, see [Launching your Bitbucket Server instance](#).

User Data is flexible and allows you to run arbitrary BASH commands on your instance at launch time, in addition to overriding variables in `/etc/atlassian`. See [Running Commands on Your Linux Instance at Launch](#) for more information.



Security considerations

See [Securing Bitbucket Server in AWS](#) for more details about enabling HTTPS and self-signed certificates in the Atlassian Bitbucket Server AMI.

Add storage

When attaching EBS volumes, use these storage device settings for your instance.

Type	Device	Purpose	Size (GiB)	Volume Type	IOPS	Delete on Termination
Root	<code>/dev/xvda</code>	Linux root volume	10	General Purpose (SSD)	30	No
EBS	<code>/dev/xvdf</code>	Bitbucket Server data: repositories, attachments, avatars, etc.	100+	General Purpose (SSD) / Provisioned IOPS *	300+ *	No
Instance Store	<code>/dev/xvdb</code>	Bitbucket Server temporary files and caches	N/A	N/A	N/A	N/A

* Provisioned IOPS with at least 500 – 1000 IOPS is recommended for instances with more than 500 active users. See [Recommendations for running Bitbucket Server in AWS](#) for more information.

The Atlassian Bitbucket Server AMI will not use any other block devices attached to the instance. The EBS volume for `/dev/xvdf` will be initialized and formatted at launch time, unless a snapshot id is provided (see the capture below in the page), in which case it will only format it if it's not already formatted. See [Managing EBS Volumes](#) for more information about storage options in Amazon EC2.

Attach an existing EBS snapshot

You can also attach an existing EBS volume based on a snapshot during launch. To attach an existing EBS volume, within the *Device* field, change the EBS volume device to `/dev/sdf` and enter the Snapshot ID of the snapshot.

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Type	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted
Root						
EBS	snap-1b4bed30	10	General Purpose (SSD)	30 / 3000	☐	Not Encrypted
EBS	snap-735d3027	100	General Purpose (SSD)	300 / 3000	☐	Not Encrypted
Instance Store 0	N/A	N/A	N/A	N/A	N/A	Not Encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

See [Administering Bitbucket Server in AWS - Moving your Bitbucket Server data volume between instances](#) for more details.

Configure your Security Group

When configuring your Security Group, you must allow incoming traffic to all the following ports. For more information, see [Using Security Groups](#).

Type	Protocol	Port	Description
SSH	TCP	22	SSH port, allowing access to administrative functions
HTTP	TCP	80	
HTTPS	TCP	443	
Custom TCP Rule	TCP	7999	Bitbucket Server SSH port for Git hosting operations

What's next?

Now you're ready to configure your AWS version of Bitbucket Server.

View your new instance

Once your new EC2 instance has launched, find it within the EC2 console and navigate to the URL provided so you can continue to configuring Bitbucket Server.

To find the URL of your new EC2 instance

1. From within the EC2 Console, in the Description tab of your new instance, copy the **Public DNS**.

The screenshot shows the AWS Management Console interface for an EC2 instance. The instance name is i-0bedadfb, and it is in the 'running' state. The Public DNS is displayed as ec2-54-152-199-123.compute-1.amazonaws.com. The instance is located in the us-east-1b Availability Zone. The instance type is m3.xlarge. The instance is currently initializing.

2. Paste the URL into a browser window to view start using Bitbucket Server.

Set up your AWS instance of Bitbucket Server

Once you've followed the URL of the EC2 instance you are presented with the [Bitbucket Server Setup Wizard](#).

Once you have launched Bitbucket Server within AWS you can use it like any other Bitbucket Server instance. So be sure to check out the rest of the [Getting Started with Bitbucket Server](#) documentation.

Administering Bitbucket Server in AWS

This page describes the Atlassian Bitbucket Server Amazon Machine Image (AMI), what's inside it, how to launch it, and how to perform administration tasks on your Bitbucket Server instance in the Amazon Web Services (AWS) environment.

The Bitbucket Server AMI

The Atlassian Bitbucket Server AMI provides a typical deployment of Bitbucket Server in AWS. It bundles all the components used in a typical Bitbucket Server deployment (reverse proxy, external database, backup tools, data volume, and temporary storage), pre-configured and ready to launch.

You can use the Atlassian Bitbucket Server AMI as a "turnkey" deployment of a Bitbucket Server instance in AWS, or use it as the starting point for customizing your own, more complex Bitbucket Server deployments.

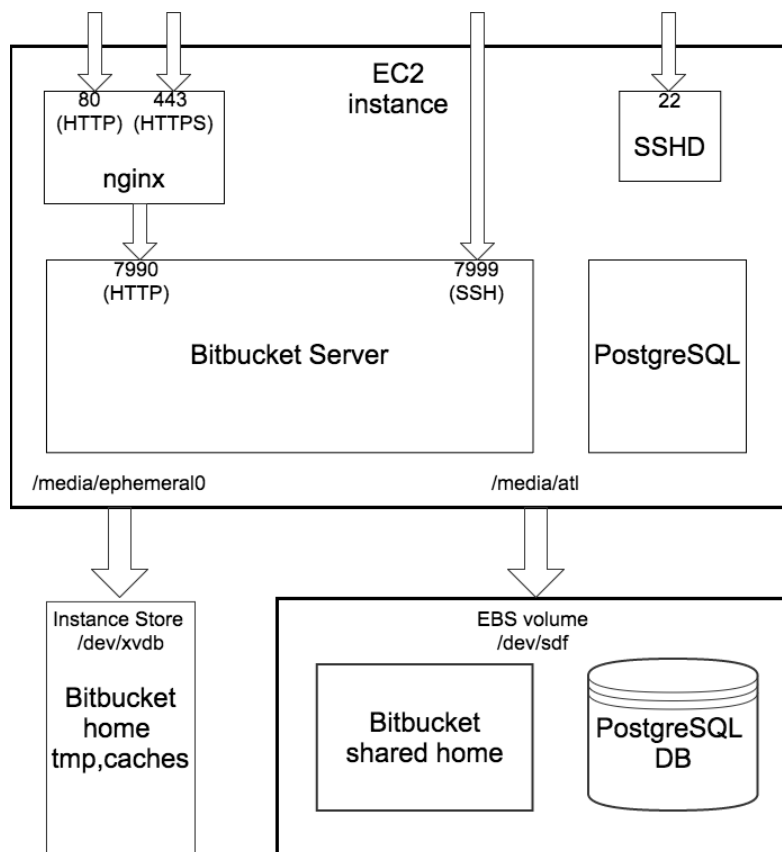
Components of the Bitbucket Server AMI

An instance launched from the Atlassian Bitbucket Server AMI contains the following components:

- Bitbucket Server (either the latest version or a version of your choice),
- an external PostgreSQL database,
- nginx as a reverse proxy,
- the Bitbucket Server DIY Backup utilities pre-configured for native AWS snapshots,
- an EBS Volume and Instance Store to hold the data.

On this page:

- [The Bitbucket Server AMI](#)
- [Components of the Bitbucket Server AMI](#)
- [Launching your Bitbucket Server instance](#)
- [Connecting to your Bitbucket Server instance using SSH](#)
- [Installing an SSL certificate in your Bitbucket Server instance](#)
- [Backing up your Bitbucket Server instance](#)
- [Upgrading your Bitbucket Server instance](#)
- [Stopping and starting your EC2 instance](#)
- [Migrating your existing Bitbucket Server instance into AWS](#)
- [Resizing the data volume in your Bitbucket Server instance](#)
- [Moving your Bitbucket Server data volume between instances](#)



Operating system	Amazon Linux 64-bit, 2014.09.1
Bitbucket Server	Bitbucket Server (latest public version or a version of your choice) is downloaded and installed on launch.
Administrative tools	atlassian-bitbucket-diy-backup pre-installed and configured for AWS native backup, accessible over SSH.
Reverse proxy	nginx , configured as follows: • listens on port 80 and (optionally) 443, • (optionally) terminates SSL (HTTPS) and passes through plain HTTP to Bitbucket Server, • displays a static HTML page when the Bitbucket Server service is not running.
Database	PostgreSQL 9.3
Block devices	1. An EBS volume (/dev/xvdf, mounted as /media/atl), that stores: • the Bitbucket Server shared home directory, containing all of Bitbucket Server's repository, attachment, and other data, • PostgreSQL's data directory. 2. An EC2 Instance Store (/dev/xvdb, mounted on /media/ephemeral0) to store Bitbucket Server's temporary and cache files.

Launching your Bitbucket Server instance

The Atlassian Bitbucket Server AMI can be launched by either

- Using a CloudFormation template which automates creation of the associated Security Group and IAM Role, see [Quick Start with Bitbucket Server and AWS](#).
- Manually using the AWS Console which gives finer control over the optional components to enable in the instance and AWS-specific network, security, and block device settings, see [Launching Bitbucket Server in AWS manually](#).

On first boot, the Atlassian Bitbucket Server AMI reads the file `/etc/atlassian` (if any), which can override variables that enable each of the installed components. So for example to enable a self-signed SSL certificate, you can supply user data to the instance at launch time like this:

```
#!/bin/bash
echo "ATL_SSL_SELF_CERT_ENABLED=true" >>/etc/atlassian
```

The following variables can be configured:

Variable name	Default value	Description
ATL_NGINX_ENABLED	true	Set to false to disable the Nginx reverse proxy, and leave Bitbucket Server's <code>server.xml</code> configured to listen on port 7990 with no proxy.
ATL_POSTGRES_ENABLED	true	Set to false to disable the PostgreSQL service, and leave Bitbucket Server configured with its internal H2 database.
ATL_SSL_SELF_CERT_ENABLED	false	Set to true to enable a self-signed SSL certificate to be generated at launch time, and for Bitbucket Server's <code>server.xml</code> and Nginx's <code>nginx.conf</code> to be configured for HTTPS. Requires ATL_NGINX_ENABLED also to be true.

See [Proxying and securing Bitbucket Server](#) for more information about Bitbucket Server's `server.xml` configuration file.

Connecting to your Bitbucket Server instance using SSH

When connecting to your instance over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem
ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

The `ec2-user` has sudo access. The Atlassian Bitbucket Server AMI does not allow SSH access by root.

Installing an SSL certificate in your Bitbucket Server instance

If launched with a self-signed SSL certificate (you selected **SSLCertificate** Generate a self-signed certificate in [Quick Start with Bitbucket Server and AWS](#) or you set `ATL_SSL_SELF_CERT_ENABLED=true` in [Launching Bitbucket Server in AWS manually](#)), Bitbucket Server will be configured to force HTTPS and redirect all plain HTTP requests to the equivalent `https://` URL.

It is highly recommended to replace this self-signed SSL certificate with a proper one for your domain, obtained from a Certification Authority (CA), at the earliest opportunity. See [Securing Bitbucket Server in AWS](#). Once you have a true SSL certificate, install it as soon as possible.

To replace the self-signed SSL certificate with a true SSL certificate

1. Place your certificate file at (for example) `/etc/nginx/ssl/my-ssl.crt`
2. Place your *password-less* certificate key file at `/etc/nginx/ssl/my-ssl.key`
3. Edit `/etc/nginx/nginx.conf` as follows:
 - a. Replace references to `/etc/nginx/ssl/self-ssl.crt` with `/etc/nginx/ssl/my-ssl.crt`
 - b. Replace references to `/etc/nginx/ssl/self-ssl.key` with `/etc/nginx/ssl/my-ssl.key`
4. Append the contents of `/etc/nginx/ssl/my-ssl.crt` to the default system PKI bundle (`/etc/pki/tls/certs/ca-bundle.crt`) to ensure scripts on the instance (such as DIY backup) can cur

- l successfully.
5. Restart nginx.

Backing up your Bitbucket Server instance

The Atlassian Bitbucket Server AMI includes a complete set of Bitbucket Server DIY Backup scripts which has been built specifically for AWS. For instructions on how to backup and restore your instance please refer to [Using Bitbucket Server DIY Backup in AWS](#).

Upgrading your Bitbucket Server instance

To upgrade to a later version of Bitbucket Server in AWS you first must [connect to your instance using SSH](#), then follow the steps in the [Bitbucket Server upgrade guide](#).

Stopping and starting your EC2 instance

An EC2 instance launched from the Atlassian Bitbucket Server AMI can be stopped and started just as any machine can be powered off and on again.

When stopping your EC2 instance, it is important to first

1. Stop the `atlbitbucket` and `postgresql93`
2. Unmount the `/media/atl` filesystem.

If your EC2 instance becomes unavailable after stopping and restarting

When starting your EC2 instance back up again, if you rely on Amazon's automatically assigned [public IP address](#) (rather than a fixed private IP address or Elastic IP address) to access your instance, your IP address may have changed. When this happens, your instance can become inaccessible and display a "The host name for your Atlassian instance has changed" page. To fix this you need to update the hostname for your Bitbucket Server instance.

To update the hostname for your Bitbucket Server instance

1. Connect to your instance over SSH and run
`sudo /opt/atlassian/bin/atl-update-host-name.sh`
2. Wait for Bitbucket Server to restart.
3. If you have also set up Bitbucket Server's Base URL to be the public DNS name or IP address you should also [update Bitbucket Server's base URL in the administration screen](#) to reflect the change.

Migrating your existing Bitbucket Server instance into AWS

Migrating an existing Bitbucket Server instance to AWS involves moving consistent backups of your `${BITBUCKET_HOME}` and your database to the AWS instance.

To migrate your existing Bitbucket Server instance into AWS

1. Check for any known migration issues in the [Bitbucket Server Knowledge Base](#).
2. Alert users to the forthcoming Bitbucket Server service outage.
3. [Create a user](#) in the Bitbucket Server Internal User Directory with `SYSADMIN` permissions to the instance so you don't get locked out if the new server is unable to connect to your User Directory.
4. Take a backup of your instance with either the [Bitbucket Server Backup Client](#) or the [Bitbucket Server DIY Backup](#).
5. Launch Bitbucket Server in AWS using the [Quick Start instructions](#), which uses a CloudFormation template.
6. Connect to your AWS EC2 instance with SSH and upload the backup file.
7. Restore the backup with the same tool used to generate it.
8. If necessary, update the JDBC configuration in the `${BITBUCKET_HOME}/shared/bitbucket.properties` file.

Resizing the data volume in your Bitbucket Server instance

By default, the application data volume in an instance launched from the Atlassian Bitbucket Server AMI is a standard Linux ext4 filesystem, and can be resized using the standard Linux command line tools.

To resize the data volume in your Bitbucket Server instance

1. Stop the `atlbitbucket` and `postgresql93` services.

2. Unmount the `/media/atl` filesystem.
3. Create a snapshot of the volume to resize.
4. Create a new volume from the snapshot with the desired size, in the same availability zone as your EC2 instance.
5. Detach the old volume and attach the newly resized volume as `/dev/sdf`.
6. Resize `/dev/sdf` using `resize2fs`, verify that its size has changed, and remount it on `/media/atl`.
7. Start the `postgresql93` and `atlbitbucket` services.

For more information, see [Expanding the Storage Space of an EBS Volume on Linux](#), [Expanding a Linux Partition](#), and the Linux manual pages for `resize2fs` and related commands.

Moving your Bitbucket Server data volume between instances

Occasionally, you may need to move your Bitbucket Server data volume to another instance—for example, when setting up staging or production instances, or when moving to an instance to a different availability zone.

There are two approaches to move your Bitbucket Server data volume to another instance

1. Take a backup of your data volume with Bitbucket Server DIY Backup, and restore it on your new instance. See [Using Bitbucket Server DIY Backup in AWS](#) for this option.
2. Launch a new instance from the Atlassian Bitbucket Server AMI with a snapshot of your existing data volume.

A Bitbucket Server data volume may only be moved to a Bitbucket Server instance of the same or higher version than the original.

To launch a new instance from the Bitbucket Server AMI using a snapshot of your existing Bitbucket Server data volume

1. Stop the `atlbitbucket` and `postgresql93` services on your existing Bitbucket Server instance.
2. Unmount the `/media/atl` filesystem.
3. Create a snapshot of the Bitbucket Server data volume (the one attached to the instance as `/dev/sdf`).
4. Once the snapshot generation has completed, launch a new instance from the Atlassian Bitbucket Server AMI as described in [Launching Bitbucket Server in AWS manually](#). When adding storage, change the EBS volume device to `/dev/sdf` as seen below and enter the id of the created snapshot.

Step 4: Add Storage
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Type	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted
Root	snap-1b4bed30	10	General Purpose (SSD)	30 / 3000	☐	Not Encrypted
EBS	snap-735d3027	100	General Purpose (SSD)	300 / 3000	☐	Not Encrypted
Instance Store 0	N/A	N/A	N/A	N/A	N/A	Not Encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

5. If the host name (private or public) that users use to reach your Bitbucket Server instance has changed as a result of moving availability zones (or as a result of stopping an instance and starting a new one) you will need to SSH in and run `sudo /opt/atlassian/bin/atl-update-host-name.sh <newhostname>` where `<newhostname>` is the new host name.
6. Once Bitbucket Server has restarted your new instance should be fully available.
7. If the host name has changed you should also update the JDBC URL configuration in the `bitbucket.properties` file (typically located in `/var/atlassian/application-data/bitbucket/shared/`), as well as Bitbucket Server's base URL in the [administration screen](#) to reflect this.

Recommendations for running Bitbucket Server in AWS

To get the best performance out of your Bitbucket Server deployment in AWS, it's important not to under-provision your instance's CPU, memory, or I/O resources. Note that the very smallest instance types provided by AWS do not meet Bitbucket Server's [minimum hardware requirements](#) and are not recommended in production environments. If you do not provision sufficient resources for your workload, Bitbucket Server may exhibit slow response times, display a [Bitbucket Server is reaching resource limits](#) banner, or fail to start altogether.

Recommended EC2 and EBS instance sizes

The following table lists the recommended EC2 and EBS configurations for Bitbucket Server in AWS under typical workloads.

On this page

- [Recommended EC2 and EBS instance sizes](#)
- [Other supported instance sizes](#)
- [Advanced: Monitoring your Bitbucket Server instance to tune instance sizing](#)

Active Users	EC2 instance type	EBS Optimized	EBS Volume type	IOPS
0 – 250	c3.large	No	General Purpose (SSD)	N/A
250 – 500	c3.xlarge	Yes	General Purpose (SSD)	N/A
500 – 1000	c3.2xlarge	Yes	Provisioned IOPS	500 – 1000

In Bitbucket Server instances with high hosting workload, I/O performance is often the limiting factor. It is recommended to pay particular attention to EBS volume options, especially the following:

- The size of an EBS volume also influences I/O performance. Larger EBS volumes generally have a larger slice of the available bandwidth and I/O operations per second (IOPS). A minimum of 100 GiB is recommended in production environments.
- The IOPS that can be sustained by General Purpose (SSD) volumes is limited by Amazon's I/O credits. If you exhaust your I/O credit balance, your IOPS will be limited to the baseline level. You should consider using a larger General Purpose (SSD) volume or switching to a Provisioned IOPS (SSD) volume. See [Amazon EBS Volume Types](#) for more information.
- New EBS volumes in particular have reduced performance the first time each block is accessed. See [Pre-Warming Amazon EBS Volumes](#) for more information.

The above recommendations are based on a **typical** workload with the specified number of active users. The resource requirements of an actual Bitbucket Server instance may vary with a number of factors, including:

- The number of continuous integration servers cloning or fetching from Bitbucket Server: Bitbucket Server will use more resources if you have many build servers set to clone or fetch frequently from Bitbucket Server.
- Whether continuous integration servers are using push mode notifications or polling repositories regularly to watch for updates.
- Whether continuous integration servers are set to do full clones or shallow clones.
- Whether the majority of traffic to Bitbucket Server is over HTTP, HTTPS, or SSH, and the encryption ciphers used.
- The number and size of repositories: Bitbucket Server will use more resources when you work on many very large repositories.
- The activity of your users: Bitbucket Server will use more resources if your users are actively using the Bitbucket Server web interface to browse, clone and push, and manipulate Pull Requests.
- The number of open Pull Requests: Bitbucket Server will use more resources when there are many open Pull Requests, especially if they all target the same branch in a large, busy repository.

See [Scaling Bitbucket Server](#) and [Scaling Bitbucket Server for Continuous Integration performance](#) for more detailed information on Bitbucket Server resource requirements.

Other supported instance sizes

The following [Amazon EC2 instances](#) also meet or exceed Bitbucket Server's [minimum hardware requirements](#). These instances provide different balances of CPU, memory, and I/O performance, and can cater for workloads that are more CPU-, memory-, or I/O-intensive than the typical.

Model	vCPU	Mem (GiB)	Instance Store (GB)	EBS optimizations available	Dedicated EBS Throughput (Mbps)
c3.large	2	3.75	2 x 16 SSD	-	-
c3.xlarge	4	7.5	2 x 40 SSD	Yes	-
c3.2xlarge	8	15	2 x 80 SSD	Yes	-
c3.4xlarge	16	30	2 x 160 SSD	Yes	-
c3.8xlarge	32	60	2 x 320 SSD	-	-
c4.large	2	3.75	-	Yes	500
c4.xlarge	4	7.5	-	Yes	750
c4.2xlarge	8	15	-	Yes	1,000
c4.4xlarge	16	30	-	Yes	2,000
c4.8xlarge	36	60	-	Yes	4,000
hs1.8xlarge	16	117	24 x 2000	-	-
i2.xlarge	4	30.5	1 x 800 SSD	Yes	-
i2.2xlarge	8	61	2 x 800 SSD	Yes	-
i2.4xlarge	16	122	4 x 800 SSD	Yes	-
i2.8xlarge	32	244	8 x 800 SSD	-	-
m3.large	2	7.5	1 x 32 SSD	-	-
m3.xlarge	4	15	2 x 40 SSD	Yes	-
m3.2xlarge	8	30	2 x 80 SSD	Yes	-
r3.large	2	15.25	1 x 32 SSD	-	-
r3.xlarge	4	30.5	1 x 80 SSD	Yes	-
r3.2xlarge	8	61	1 x 160 SSD	Yes	-
r3.4xlarge	16	122	1 x 320 SSD	Yes	-
r3.8xlarge	32	244	2 x 320 SSD	-	-

In all AWS instance types, Bitbucket Server only supports "large" and higher instances. "Micro", "small", and "medium" sized instances do not meet Bitbucket Server's [minimum hardware requirements](#) and are not recommended in production environments.

Bitbucket Server does not support [D2 instances](#), [Burstable Performance \(T2\) Instances](#), or [Previous Generation Instances](#).

In any instance type with available Instance Store device(s), a Bitbucket Server instance launched from the Atlassian Bitbucket Server AMI will configure one Instance Store to contain Bitbucket Server's temporary files

and caches. Instance Store can be faster than an EBS volume but the data does not persist if the instance is stopped or rebooted. Use of Instance Store can improve performance and reduce the load on EBS volumes. See [Amazon EC2 Instance Store](#) for more information.

Advanced: Monitoring your Bitbucket Server instance to tune instance sizing

This section is for advanced users who wish to monitor the resource consumption of their instance and use this information to guide instance sizing.

The above recommendations provide guidance for **typical** workloads. The resource consumption of every Bitbucket Server instance, though, will vary with the mix of workload. The most reliable way to determine if your Bitbucket Server instance is under- or over-provisioned in AWS is to monitor its resource usage regularly with [Amazon CloudWatch](#). This provides statistics on the actual amount of CPU, I/O, and network resources consumed by your Bitbucket Server instance.

The following simple example BASH script uses

- the [AWS CLI](#),
- [gnuplot](#),
- [jq](#)

to gather CPU, I/O, and network statistics and display them in a simple chart that can be used to guide your instance sizing decisions.

▼ [Click here to expand...](#)

```
#!/bin/bash
# Example AWS CloudWatch monitoring script
# Usage:
#   (1) Install gnuplot and jq (minimum version 1.4)
#   (2) Install AWS CLI
#       (http://docs.aws.amazon.com/cli/latest/userguide/installing.html)
#       and configure it with
#       credentials allowing cloudwatch get-metric-statistics
#   (3) Replace "xxxxxxx" in volume_ids and instance_ids below with
#       the ID's of your real instance
#   (4) Run this script

export start_time=$(date -v-14d +%Y-%m-%dT%H:%M:%S)
export end_time=$(date +%Y-%m-%dT%H:%M:%S)
export period=1800
export volume_ids="vol-xxxxxxx"      # REPLACE THIS WITH THE VOLUME
ID OF YOUR REAL EBS VOLUME
export instance_ids="i-xxxxxxx"      # REPLACE THIS WITH THE
INSTANCE ID OF YOUR REAL EC2 INSTANCE

# Build lists of metrics and datafiles that we're interested in
ebs_metrics=""
ec2_metrics=""
cpu_datafiles=""
iops_datafiles=""
queue_datafiles=""
net_datafiles=""
for volume_id in ${volume_ids}; do
  for metric in VolumeWriteOps VolumeReadOps; do
    ebs_metrics="${ebs_metrics} ${metric}"
    iops_datafiles="${iops_datafiles} ${volume_id}-${metric}"
  done
done
for volume_id in ${volume_ids}; do
```

```

    for metric in VolumeQueueLength; do
        ebs_metrics="${ebs_metrics} ${metric}"
        queue_datafiles="${queue_datafiles} ${volume_id}-${metric}"
    done
done
for instance_id in ${instance_ids}; do
    for metric in DiskWriteOps DiskReadOps; do
        ec2_metrics="${ec2_metrics} ${metric}"
        iops_datafiles="${iops_datafiles} ${instance_id}-${metric}"
    done
done
for instance_id in ${instance_ids}; do
    for metric in CPUUtilization; do
        ec2_metrics="${ec2_metrics} ${metric}"
        cpu_datafiles="${cpu_datafiles} ${instance_id}-${metric}"
    done
done
for instance_id in ${instance_ids}; do
    for metric in NetworkIn NetworkOut; do
        ec2_metrics="${ec2_metrics} ${metric}"
        net_datafiles="${net_datafiles} ${instance_id}-${metric}"
    done
done

# Gather the metrics using AWS CLI
for volume_id in ${volume_ids}; do
    for metric in ${ebs_metrics}; do
        aws cloudwatch get-metric-statistics --metric-name ${metric} \
                                           --start-time ${start_time} \
                                           \
                                           --end-time ${end_time} \
                                           --period ${period} \
                                           --namespace AWS/EBS \
                                           --statistics Sum \
                                           --dimensions

        Name=VolumeId,Value=${volume_id} | \
        jq -r '.Datapoints | sort_by(.Timestamp) | map(.Timestamp + "
" + (.Sum | tostring)) | join("\n")' >${volume_id}-${metric}.data
    done
done

for metric in ${ec2_metrics}; do
    for instance_id in ${instance_ids}; do
        aws cloudwatch get-metric-statistics --metric-name ${metric} \
                                           --start-time ${start_time} \
                                           \
                                           --end-time ${end_time} \
                                           --period ${period} \
                                           --namespace AWS/EC2 \
                                           --statistics Sum \
                                           --dimensions

        Name=InstanceId,Value=${instance_id} | \
        jq -r '.Datapoints | sort_by(.Timestamp) | map(.Timestamp + "
" + (.Sum | tostring)) | join("\n")' >${instance_id}-${metric}.data
    done
done

```



```

cat >aws-monitor.gnuplot <<EOF
set term pngcairo font "Arial,30" size 1600,900
set title "IOPS usage"
set datafile separator whitespace
set xdata time
set timefmt "%Y-%m-%dT%H:%M:%SZ"
set grid
set ylabel "IOPS"
set xrange ["${start_time}Z":"${end_time}Z"]
set xtics "${start_time}Z",86400*2 format "%d-%b"
set output "aws-monitor-iops.png"
plot \
EOF
for datafile in ${iops_datafiles}; do
    echo " \`${datafile}.data\" using 1:(\`$2/${period}) with lines
title \`${datafile}\`, \`${datafile}\`, \`${datafile}\`" >>aws-monitor.gnuplot
done

cat >>aws-monitor.gnuplot <<EOF

set term pngcairo font "Arial,30" size 1600,900
set title "IO Queue Length"
set datafile separator whitespace
set xdata time
set timefmt "%Y-%m-%dT%H:%M:%SZ"
set grid
set ylabel "Queue Length"
set xrange ["${start_time}Z":"${end_time}Z"]
set xtics "${start_time}Z",86400*2 format "%d-%b"
set output "aws-monitor-queue.png"
plot \
EOF
for datafile in ${queue_datafiles}; do
    echo " \`${datafile}.data\" using 1:2 with lines title
\`${datafile}\`, \`${datafile}\`, \`${datafile}\`" >>aws-monitor.gnuplot
done

cat >>aws-monitor.gnuplot <<EOF

set term pngcairo font "Arial,30" size 1600,900
set title "CPU Utilization"
set datafile separator whitespace
set xdata time
set timefmt "%Y-%m-%dT%H:%M:%SZ"
set grid
set ylabel "%"
set xrange ["${start_time}Z":"${end_time}Z"]
set xtics "${start_time}Z",86400*2 format "%d-%b"
set output "aws-monitor-cpu.png"
plot \
EOF
for datafile in $cpu_datafiles; do
    echo " \`${datafile}.data\" using 1:2 with lines title
\`${datafile}\`, \`${datafile}\`, \`${datafile}\`" >>aws-monitor.gnuplot
done

cat >>aws-monitor.gnuplot <<EOF

```

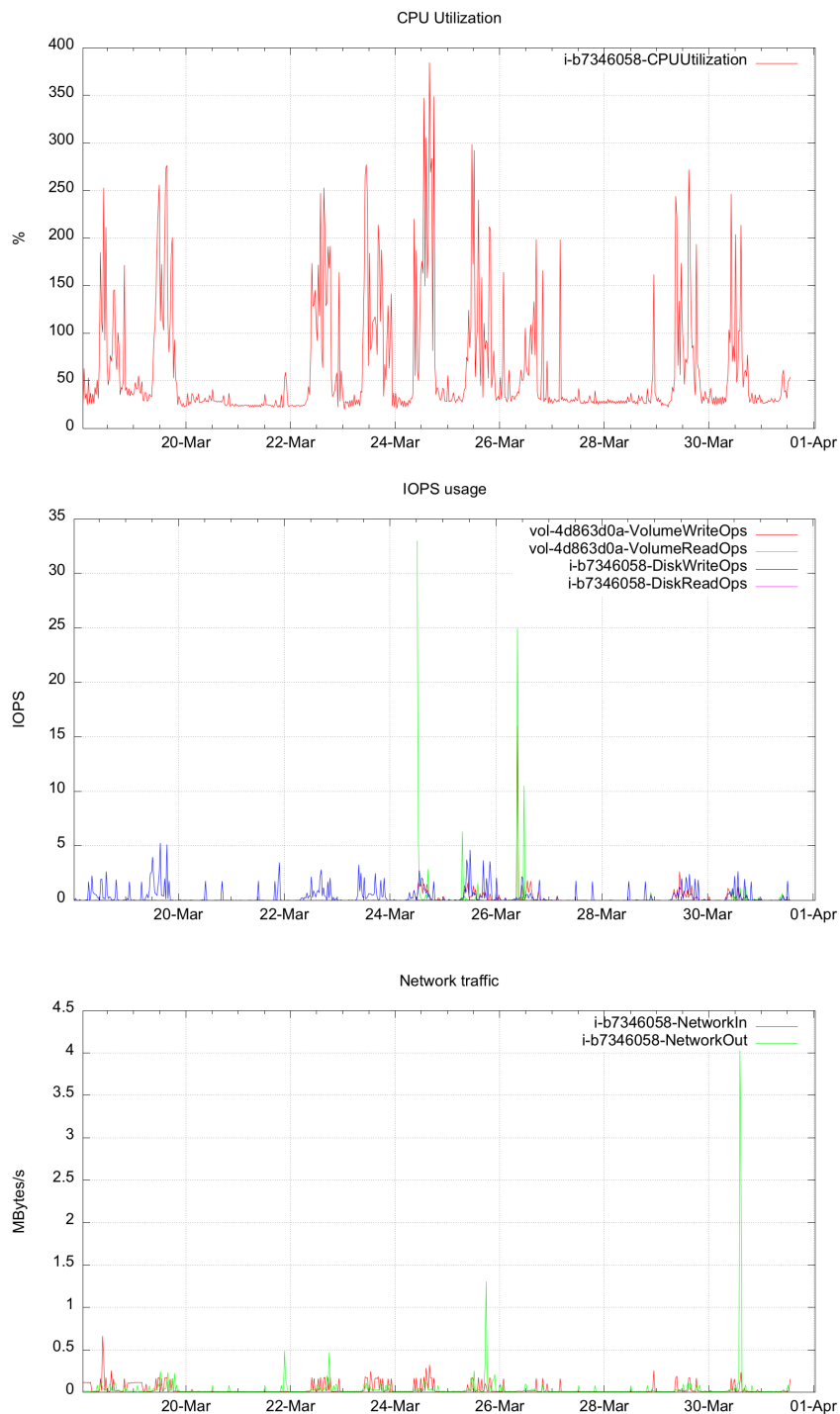
```
set term pngcairo font "Arial,30" size 1600,900
set title "Network traffic"
set datafile separator whitespace
set xdata time
set timefmt "%Y-%m-%dT%H:%M:%SZ"
set grid
set ylabel "MBytes/s"
set xrange ["${start_time}Z":"${end_time}Z"]
set xtics "${start_time}Z",86400*2 format "%d-%b"
set output "aws-monitor-net.png"
plot \
EOF
for datafile in $net_datafiles; do
    echo "  \"${datafile}.data\" using 1:(\${2}/${period}/1000000) with
lines title \"${datafile}\", \"\" >>aws-monitor.gnuplot
```

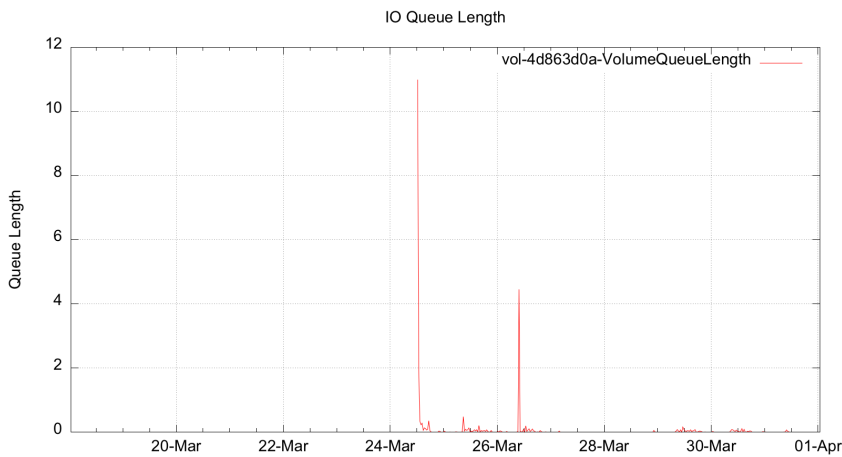


```
done
```

```
gnuplot <aws-monitor.gnuplot
```

When run on a typical Bitbucket Server instance, this script produces charts such as the following:





You can use the information in charts such as this to decide whether CPU, network, or I/O resources are over- or under-provisioned in your instance.

If your instance is frequently saturating the maximum available CPU (taking into account the number of cores in your instance size), then this may indicate you need an EC2 instance with a larger CPU count. (Note that the CPU utilization reported by Amazon CloudWatch for smaller EC2 instance sizes may be influenced to some extent by the "noisy neighbor" phenomenon, if other tenants of the Amazon environment consume CPU cycles from the same physical hardware that your instance is running on.)

If your instance is frequently exceeding the IOPS available to your EBS volume and/or is frequently queuing I/O requests, then this may indicate you need to upgrade to an EBS optimized instance and/or increase the Provisioned IOPS on your EBS volume. See [EBS Volume Types](#) for more information.

If your instance is frequently limited by network traffic, then this may indicate you need to choose an EC2 instance with a larger available slice of network bandwidth.

Securing Bitbucket Server in AWS

This page describes security best practices for running and maintaining Bitbucket Server in AWS.

Amazon Virtual Private Cloud (VPC) and Subnets

Amazon VPC enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. See [Amazon EC2 and Amazon Virtual Private Cloud](#) for more information.

A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a subnet that you select. Use a public subnet for resources that must be connected to the Internet, and a private subnet for resources that won't be connected to the Internet.

See Amazon's article called [Your VPC and Subnets](#) for a general overview of VPCs and subnets.

To bolster the security of your VPC you may wish to enable one or more of the following:

- Secure your VPC with a firewall virtual appliance / AMI to defend against unauthorized network activity
- Configure a site-to-site VPN to ensure information is transferred securely between Bitbucket Server and its users
- Configure an intrusion prevention or intrusion detection virtual appliance to detect when unauthorized network activity has occurred
- Enable [Amazon CloudTrail](#) to log VPC API operations and keep an audit trail of network changes

Security Groups

A security group acts as a virtual firewall that controls the traffic for one or more instances. When you launch

On this page:

- [Amazon Virtual Private Cloud \(VPC\) and Subnets](#)
- [Security Groups](#)
- [HTTPS](#)
- [Self-signed SSL certificates](#)
- [Domain name](#)
- [Trusted CA-issued certificates](#)
- [Keeping your system up-to-date](#)

an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances. You can modify the rules for a security group at any time; the new rules are automatically applied to all instances that are associated with the security group. See [Amazon EC2 Security Groups for Linux Instances](#) for more information.

We recommend you restrict the security groups that apply to the Bitbucket Server instance to the [absolute minimum required](#). As an instance can have up to a hundred security groups applied to it, it can be difficult to understand which restrictions are in effect. It is for this reason we recommend you condense the applied security groups to as few as possible.

HTTPS

By default, the Bitbucket Server AMI configures Bitbucket Server to serve requests over HTTP not HTTPS. If you are not connected to the AWS VPC your Bitbucket Server resides in via a [Virtual Private Gateway](#), then all passwords and data will be sent unencrypted over the public Internet. If you intend for your Bitbucket Server instance to be Internet facing, setting `ATL_SSL_SELF_CERT_ENABLED=true` is recommended to enable HTTPS to your instance at launch time.

Self-signed SSL certificates

If HTTPS is enabled via `ATL_SSL_SELF_CERT_ENABLED=true` a self-signed certificate will be generated for your Statsh instance.

If you continue to use the self-signed certificate:

- most browsers will display security warnings that must be ignored before proceeding to the Bitbucket Server Web interface,
- Git clients will refuse to connect to Bitbucket Server over HTTPS unless configured to ignore the self-signed certificate with `git config --global http.sslVerify false`, and
- Application links and/or integrations with other applications that use Bitbucket Server's REST API and do not accept self-signed certificates may fail.

The self-signed certificate should be replaced with a certificate obtained from a trusted certificate authority (CA) at the earliest opportunity to improve your security and improve the experience of your users.

Domain name

In order to use a trusted CA-issued certificate with your Bitbucket Server instance and to avoid the problems outlined above with self-signed certificates you will first need a static public domain name associated with your instance. [Amazon Route 53](#) and other DNS providers can provide you with this. You will need to ensure you update your DNS record every time your EC2 instance's IP address changes. Using Amazon's [Elastic IP Address](#) helps minimize the IP address changes of your instance and thus minimize its day-to-day administration.

Trusted CA-issued certificates

Once you have a static domain name for your EC2 instance you can request a trusted certificate authority issue a certificate for use with this domain / instance. Installing the certificate is a straight-forward process as long as you [first set up your instance to use a self-signed certificate](#).

Keeping your system up-to-date

It is essential to keep your Bitbucket Server instance up-to-date with patches and updates to maximize security and minimize opportunity for exploits and misadventure. On first boot a Bitbucket Server AMI instance will download the latest official release of Bitbucket Server at that time so you are assured of having the very latest version of Bitbucket Server when you first start using Bitbucket Server in AWS.

Please be sure to always perform a backup of your instance before attempting any update.

Amazon Linux Security Updates

The Bitbucket Server AMI is based on Amazon Linux and the latest version of this is used whenever we cut a new release of the Bitbucket Server AMI. Occasionally vulnerabilities in libraries and utilities used in Amazon Linux will be detected and updates posted in the Amazon Linux AMI yum repository. Atlassian will issue new versions of the Bitbucket Server AMI where necessary to ensure new Bitbucket Server AWS instances start with these updates but if you are managing an existing instance you may need to apply these updates

yourself. By default, Amazon Linux applies all security updates on reboot. Alternatively you can run "yum update --security".

From time-to-time you may also wish to apply other updates from the Amazon Linux AMI yum repository to your Bitbucket Server instance. You must ensure that any updated packages are supported by the version of Bitbucket Server you are running. Bitbucket Server version requirements can always be found on the [Supported platforms](#) page.

Bitbucket Server Updates

The Atlassian Bitbucket Server team have a strong release cadence and routinely issue releases including new features, performance and security fixes. It is strongly recommended you keep Bitbucket Server as up to date as possible. To update Bitbucket Server in an existing instance please follow the [Bitbucket Server Upgrade Guide](#).

Using Bitbucket Server DIY Backup in AWS

This page describes how to execute a DIY Backup and Restore of a Bitbucket Server instance deployed in AWS.

About the Bitbucket Server DIY Backup for AWS

[Bitbucket Server DIY Backup for AWS](#) leverages AWS infrastructure to backup and restore the application. The provided scripts take EBS snapshots of the volume containing the Bitbucket Server shared home directory and the database data directory. These snapshots can later be used to create a new volume and attach it to your instance thus restoring Bitbucket Server to a specific point in time.

Other approaches include using the [Bitbucket Server Backup Client](#) or manually running the same steps as the Bitbucket Server DIY Backup for AWS yourself. The benefits of using the Bitbucket Server DIY Backup over these approaches are:

- taking AWS native snapshots are faster than filesystem level copying
- downtime is kept to an absolute minimum
- snapshots are stored with the redundancy and durability of S3
- it makes it easy to relocate an instance to a different Availability Zone in the future

The scripts use the [AWS CLI](#) toolset, which is included in all instances launched from the AMI, regardless of launch method. The template creates an [IAM role](#) with a policy that grants the instance the permissions required to backup and restore the EBS volume. See the [Advanced configuration](#) section below for an example policy with similar permissions.

Configure the Bitbucket Server DIY Backup script

The provided script, `bitbucket.diy-aws-backup.vars.sh`, comes with sensible defaults for your AWS environment, but you need to modify variables in the script for your specific setup. These variables indicate how to lock Bitbucket Server for backup, to ensure consistency by preventing writes to the volume.

To locate the Bitbucket Server DIY Backup scripts available in the default installation directory:

1. Connect to your Bitbucket Server instance on AWS over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem  
ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

2. Go to the `bitbucket-diy-backup` directory.

On this page

- [About the Bitbucket Server DIY Backup for AWS](#)
- [Configure the Bitbucket Server DIY Backup script](#)
- [Back up your instance](#)
- [Restore your instance](#)
- [Advanced configuration](#)

```
cd /opt/atlassian/bitbucket-diy-backup
```

3. Pull the latest version of the scripts. The installation directory contains a clone of the [Bitbucket Server DIY Backup scripts repository](#). You can update to the latest changes at any time.

```
git pull
```

4. Locate the variables script.

```
bitbucket.diy-aws-backup.vars.sh
```

5. Modify the variables within the script for your specific instance.

Variables

These variables must be customized.

Variable	Explanation
BITBUCKET_BACKUP_USER	The username of a Bitbucket Server user with the SYSADMIN role.
BITBUCKET_BACKUP_PASS	The password to the BITBUCKET_BACKUP_USER account.

Users who have attached their home directory volume to a device name other than `/dev/xvdf` may also need to update the `HOME_DIRECTORY_DEVICE_NAME` variable. See the [Advanced configuration](#) section for more information.

Back up your instance

To back up your Bitbucket Server instance within AWS you need to run the `bitbucket.diy-aws-backup.sh` script. This will take the values you [configured above](#) to perform the backup.

To run the Bitbucket Server DIY Backup scripts

1. Connect to your Bitbucket Server instance on AWS over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem
ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

2. Go to the `bitbucket-diy-backup` directory.

```
cd /opt/atlassian/bitbucket-diy-backup
```

3. Run the `bitbucket.diy-aws-backup.sh`.

```
./bitbucket.diy-aws-backup.sh
```

Running the scripts will produce an output similar to this:

▼ [Click here to see example output...](#)

```
[http://localhost:7990] INFO: Using IAM instance role
Bitbucket-DIY-Backup-BitbucketBackupRole-1GJ4BL2XHPDBG
[http://localhost:7990] INFO: Looking up volume for device name
/dev/sdf
[http://localhost:7990] SUCC: Found volume vol-aae66abd for device
name /dev/sdf
[http://localhost:7990] INFO: locked with
'603c1d2c41121d1b6f42c0b03d4e03ee8a22577b'
[http://localhost:7990] INFO: backup started with
'a62d9b002747877e57d7ff32cdaedc92ba66db79'
[http://localhost:7990] INFO: Waiting for DRAINED state.. done
[http://localhost:7990] INFO: db state 'DRAINED'
[http://localhost:7990] INFO: scm state 'DRAINED'
[http://localhost:7990] INFO: Backup progress updated to 50
[http://localhost:7990] INFO: Freezing filesystem at mount point
/media/atl
[http://localhost:7990] INFO: Performing backup of home directory
[http://localhost:7990] SUCC: Taken snapshot snap-2f133304 of
volume vol-aae66abd
[http://localhost:7990] INFO: Tagged snap-2f133304 with
Name=bitbucket-20150326-013036-405
[http://localhost:7990] INFO: Unfreezing filesystem at mount point
/media/atl
[http://localhost:7990] INFO: Backup progress updated to 100
[http://localhost:7990] INFO: Bitbucket instance unlocked
[http://localhost:7990] SUCC: Successfully completed the backup of
your Bitbucket instance
[http://localhost:7990] INFO: Cleaning up...
[http://localhost:7990] INFO: Unfreezing filesystem at mount point
/media/atl
```

You can review the newly created snapshot in the [AWS EC2 console](#).

Restore your instance

Restoring your instance is done by replacing the existing volume with a new one created using an existing EBS snapshot.

To restore your instance using the Bitbucket Server DIY Restore scripts

1. Connect to your Bitbucket Server instance on AWS over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem
ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

2. Go to the `bitbucket-diy-backup` directory.

```
cd /opt/atlassian/bitbucket-diy-backup
```

3. Stop your Bitbucket Server instance.

```
sudo /etc/init.d/atlbucket stop
```

4. Stop your database (if available).

```
sudo /etc/init.d/postgresql93 stop
```

5. Unmount any volumes using the configured mount point.

```
sudo umount /media/atl
```

6. Detach any volumes using the `/dev/sdf` device name. These commands export the instance region for AWS CLI tools, retrieve the ID for the volume attached to the device name, and detach the volume from the instance.

```
export AWS_DEFAULT_REGION=`curl -s -f
http://169.254.169.254/latest/meta-data/placement/availability-z
one | sed -e 's:\([0-9][0-9]*\)[a-z]*\$:\\1:'`
DETACH_VOLUME_ID=$(aws ec2 describe-volumes --filter
Name=attachment.instance-id,Values=`curl -s -f
http://169.254.169.254/latest/meta-data/instance-id`
Name=attachment.device,Values=/dev/xvdf | jq -r
'.Volumes[0].VolumeId')
aws ec2 detach-volume --volume-id ${DETACH_VOLUME_ID}
```

The previous commands assume the home directory volume is attached to device name `/dev/xvdf`

7. All snapshot taken using the DIY backup script ([see above](#)) will be tagged with key "Name" and value which includes the configured `INSTANCE_NAME` and a timestamp for when the backup was taken. To see which snapshots are available for your `INSTANCE_NAME`, run the script without any arguments.

```
./bitbucket.diy-aws-restore.sh
```

The snapshots are sorted alphabetically to keep the latest snapshots at the top of the list.

▼ [Click here to see example output...](#)


```
[http://localhost:7990] INFO: Using IAM instance role
Bitbucket-Server-DIY-Backup-Bitbucket-ServerBackupRole-1GJ4BL2
XHPDBG
[http://localhost:7990] INFO: Usage:
./bitbucket.diy-aws-restore.sh <snapshot-tag>
Available snapshot tags:
bitbucket-20150327-013036-405
bitbucket-20150326-234633-252
bitbucket-20150326-231048-106
bitbucket-20150326-231038-587
bitbucket-20150326-224449-846
bitbucket-20150326-224012-288
bitbucket-20150326-222700-117https://confluence.atlassian.com/
pages/editpage.action?pageId=776640203#
bitbucket-20150326-222522-962
bitbucket-20150326-220256-274
bitbucket-20150326-083409-384
```

8. After you select a tag, run the script again with the instance tag as an argument.

```
./bitbucket.diy-aws-restore.sh bitbucket-20150326-013036-405
```

▼ [Click here to see example output...](#)

```
[http://localhost:7990] INFO: Using IAM instance role
Bitbucket-Server-DIY-Backup-Bitbucket-ServerBackupRole-1GJ4BL2
XHPDBG
[http://localhost:7990] INFO: Restoring from tag
bitbucket-20150326-013036-405
[http://localhost:7990] INFO: Checking for existing volumes
using device name /dev/sdf
[http://localhost:7990] INFO: Found EBS snapshot
snap-2f133304 for tag bitbucket-20150326-013036-405
[http://localhost:7990] INFO: Restoring home directory from
snapshot snap-2f133304 into a gp2 volume
[http://localhost:7990] SUCC: Restored snapshot snap-2f133304
into volume vol-dae16dcd
[http://localhost:7990] INFO: Waiting for volume vol-dae16dcd
to be attached. This could take some time
[http://localhost:7990] INFO: Volume vol-dae16dcd state:
attaching
[http://localhost:7990] INFO: Volume vol-dae16dcd state:
attached
[http://localhost:7990] SUCC: Attached volume vol-dae16dcd to
device /dev/sdf at instance i-6acb8a8e
[http://localhost:7990] SUCC: Mounted device /dev/sdf to
/media/atl
[http://localhost:7990] INFO: Performed restore of home
directory snapshot
[http://localhost:7990] SUCC: Successfully completed the
restore of your Bitbucket instance
```

You can review the newly created volume in the [AWS EC2 console](#).

9. Start your instance.

```
sudo /etc/init.d/postgresql93 start
sudo /etc/init.d/atlbucket start
```

Advanced configuration

Variable	Explanation
INSTANCE_NAME	A name for your instance. It cannot contain spaces. It must be under 100 characters in length. This will be used as a prefix when tagging your instance snapshot.
BITBUCKET_URL	The Bitbucket Server base URL. It must not end with '/'
BITBUCKET_HOME	The path to the Bitbucket Server home directory. For example /var/atlassian/application-data/bitbucket
BACKUP_DATABASE_TYPE	'ebs-collocated' means the database data directory is located in the same volume as the BITBUCKET_HOME.
BACKUP_HOME_TYPE	'ebs-home' means the Bitbucket Server home (and the database data directory if ebs-collocated has been specified) will be backed up by taking a snapshot of an EBS volume.
HOME_DIRECTORY_MOUNT_POINT	The mount point for the volume holding the BITBUCKET_HOME directory as it appears in /etc/fstab (for example /media/atl)
HOME_DIRECTORY_DEVICE_NAME	The device name on to which the BITBUCKET_HOME volume is attached as it appears in the Amazon console (for example /dev/sdf)
AWS_AVAILABILITY_ZONE	The availability zone for your AWS instance. If left unchanged from the template it will be retrieved from the metadata endpoint
AWS_REGION	The region for your AWS instance. If left unchanged from the template it will be derived from the AWS_AVAILABILITY_ZONE
RESTORE_HOME_DIRECTORY_VOLUME_TYPE	The type of volume to create from the snapshot (one of io1, gp2, and standard)
RESTORE_HOME_DIRECTORY_IOPS	The provisioned IOPS for the new volume. Only necessary if RESTORE_HOME_DIRECTORY_VOLUME_TYPE is 'io1'
HIPCHAT_URL	The url for the HipChat API
HIPCHAT_ROOM	The HipChat room to which notifications should be delivered
HIPCHAT_TOKEN	The authentication token for the HipChat API

CURL_OPTIONS	A set of options to pass to the <code>curl</code> commands executed by the scripts
BITBUCKET_VERBOSE_BACKUP	If <code>FALSE</code> <code>info</code> and <code>print</code> level logging will be skipped

Setting up the instance role

The DIY backup and restore scripts use the [AWS CLI](#) toolset to do their job. These tools need to authenticate with AWS in order to gain access to your resources (EBS volumes, snapshots, etc). The recommended way of providing credentials to the instance is by launching it with an instance role that has a suitable policy attached. If you are using the [Bitbucket Server CloudFormation template](#), it will take care of creating a policy for you and attach it to the instance at launch time.

If you need to create your own policy, you can use this JSON object as an example of the minimum permissions required for an instance:

```
{
  "Statement": [
    {
      "Resource": [
        "*"
      ],
      "Action": [
        "ec2:AttachVolume",
        "ec2:CreateSnapshot",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:DescribeSnapshots",
        "ec2:DescribeVolumes",
        "ec2:DetachVolume"
      ],
      "Effect": "Allow"
    }
  ],
  "Version": "2012-10-17"
}
```

For other ways of configuring the AWS CLI toolset, please refer to the [documentation](#).

Disabling HTTP(S) access to Git repositories in Bitbucket Server

A Bitbucket Server administrator can disable HTTP(S) access to Git repositories in Bitbucket Server. This removes the ability to push to or clone Git repositories in Bitbucket Server over HTTP(S).

Related pages:

- [Enabling SSH access to Git repositories in Bitbucket Server](#)

Disabling HTTP(S) access

To disable HTTP(S) access:

1. Go to the Bitbucket Server administration area and click **Server settings** (under 'Settings').
2. Under 'HTTP(S) access', uncheck **HTTP(S) enabled**.
3. Click **Save**.

Smart Mirroring

Smart Mirroring can greatly improve Git clone speeds for distributed teams working with large repositories. Large repositories that take hours to

clone from a Bitbucket instance over the Internet from the other side of the world can take minutes when cloned from a local mirror on a fast network.

If you're ready to start mirroring your repositories you can jump straight to the [Set up a mirror](#) article and follow the steps there, or read on to learn more about the benefits of using a mirror. If you have mirrors already configured, you might be searching for instructions on how to [Clone a mirror repository](#).

This feature is only for customers with an active Bitbucket Data Center license. [Contact us](#) to obtain a Bitbucket Data Center evaluation license.

About Smart Mirroring

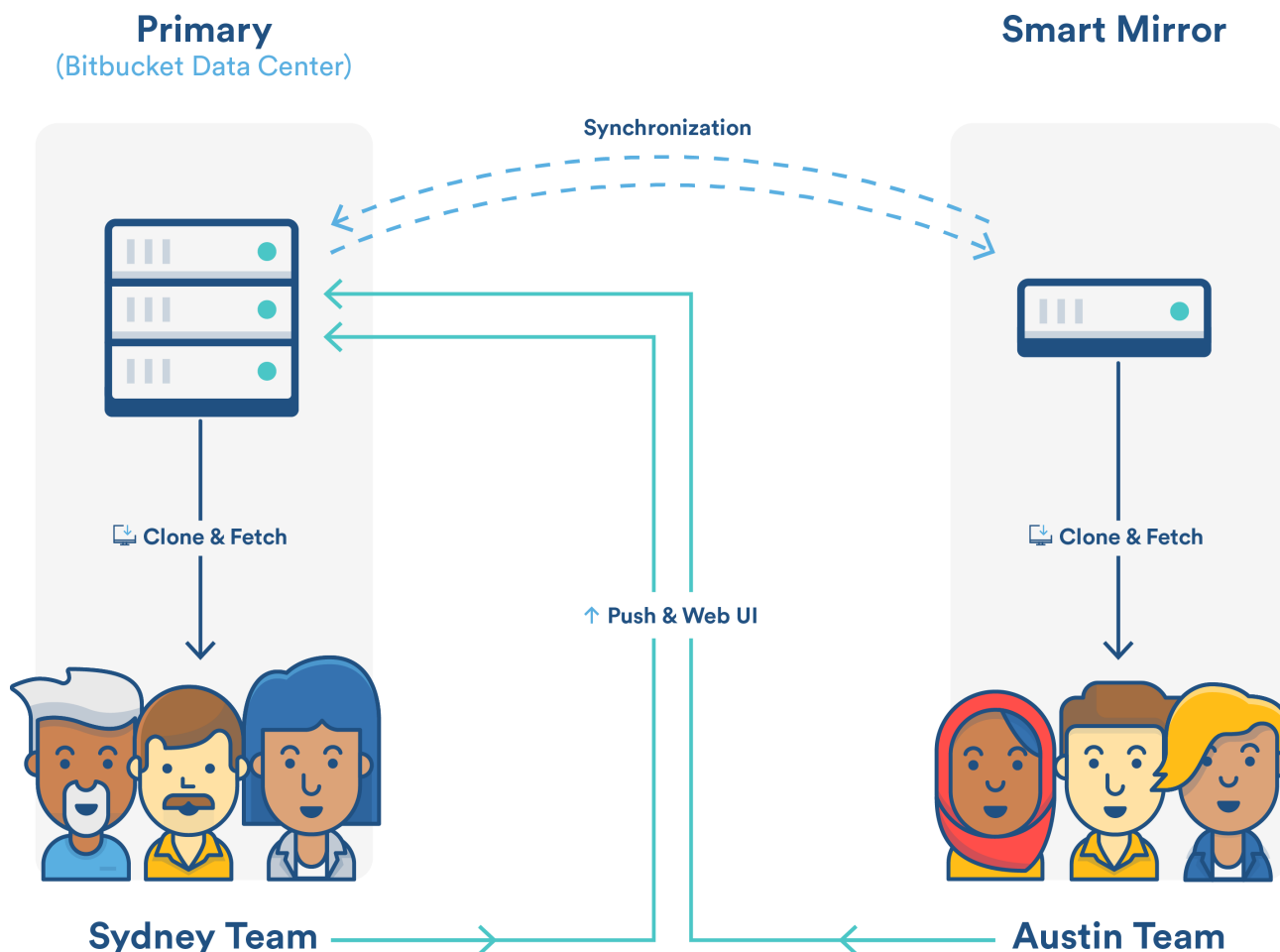
Many software development teams using Git have large repositories as a result of storing lots of historical information, using monolithic repositories, or storing large binary files (or all three). Companies with distributed software development teams often have little control over the network performance available to them between sites. In combination, this leads to lost development time when developers have to wait long periods, often hours, to clone a large repository from across the world.

Smart Mirroring gives you back this lost development time by allowing you to set up live mirror nodes with read-only copies of repositories in remote locations. The mirrors automatically keep all repositories hosted on them in sync with the primary Bitbucket Data Center instance. Users in those remote locations may clone and fetch repositories from the mirror and get identical content, only faster. Mirrors can be configured to mirror all repositories in all projects from their primary Bitbucket instance, or a selection of projects configured by an administrator.

How it works

Mirrors run the same Bitbucket application as a full Bitbucket Server instance, but configured to mirror an upstream Bitbucket Data Center instance where the primary copy of all your repositories is hosted.

Mirrors have no user interface and do not provide repository browsing, pull requests, or other such functionality, which is all provided by the primary Bitbucket server. The web interface of Bitbucket remain on the primary server.



Mirrors are *read-only* copies of a primary Bitbucket Server instance – they don't accept pushes. Fortunately you can easily configure Git to fetch from a mirror but send pushes directly to the primary instance, see [Update your push URL](#) for details about how to do this.

When a user clones or fetches from a mirror, the mirror automatically delegates the authentication and authorization of their credentials back to the primary server. So no additional user management is required on mirrors, and all the users, groups, and permissions of the primary Bitbucket instance (whether provided by the built-in user directory and permission system or by your own user directories and/or custom extensions) are always replicated *exactly* on all mirrors.

Ready to get started setting up a mirror?

Be sure to read [Set up a mirror](#) for detailed instructions on installing a mirror.

Don't have Bitbucket Data Center yet? [Request an evaluation license to get started!](#)

Set up a mirror

This feature is only for customers with an active Bitbucket Data Center license. [Contact us](#) to obtain a Bitbucket Data Center evaluation license.

Smart Mirroring can drastically improve Git clone speeds for distributed teams working with large repositories. This page describes how to install a mirror and troubleshoot any issues you might encounter.

For an overview of the benefits to mirroring, see [Smart Mirroring](#). For answers to frequently asked questions regarding mirroring, see [Bitbucket Data Center FAQ](#). These instructions assume you already

On this page

- [Before you start](#)
- [1. Install Bitbucket on the mirror](#)
- [2. Set up SSL on the mirror](#)
- [3. Start the mirror](#)
- [4. Set up the mirror](#)
- [5. Approve a mirror request](#)
- [6. Decide which projects to mirror](#)
- [Troubleshooting](#)

have a fully licensed Bitbucket Data Center instance up and running.

Before you start

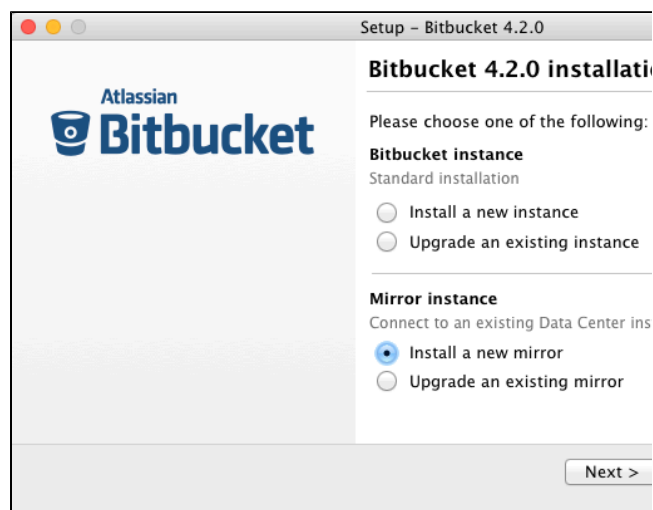
You must also meet the following requirements:

- **Your primary Bitbucket instance *must* be a fully licensed Bitbucket Data Center instance** - You do not actually have to run your Bitbucket Data Center instance as a multi-node cluster to use Mirroring, but you must have an up-to-date Data Center license.
- **The primary instance and all mirror(s) *must* have HTTPS with a valid (i.e., signed by a Certificate Authority anchored to the root and not expired) SSL certificate** - This is a strict requirement of Smart Mirroring on both the primary instance and all mirror(s), and cannot be bypassed. The mirror setup wizard will not proceed if either the mirror or the primary instance does not have a valid SSL certificate.
- **The primary Bitbucket Server instance *must* have SSH enabled** - Mirrors keep their repositories synchronized with the primary instance over SSH and can not use HTTP or HTTPS for this. See [Enabling SSH access to Git repositories in Bitbucket Server](#) for instructions on enabling SSH access on your primary instance.
- **The platform the mirror is running on *must* meet the same minimum requirements for Bitbucket Data Center** - Check the [Supported platforms](#) for detailed requirements, including those for Java and Git, apply to each mirror.
- **The platform the mirror is running on *should not* be under-provisioned** - Just as with your primary Bitbucket Data Center instance, your mirrors need to be provisioned with enough CPU, memory, and I/O resources to handle their peak workload. See [Scaling Bitbucket Server](#) for more information.
- **Minimum version Bitbucket 4.2** - Both the primary instance and mirror(s) must be running Bitbucket 4.2.0 or higher. (The primary and mirror(s) do not need to be identical versions, provided they are both 4.2.0 or higher.)
- **Running your mirror behind a reverse proxy server and terminating SSL at the proxy is highly recommended** - See [Proxying and securing Bitbucket Server](#).

1. Install Bitbucket on the mirror

The easy way to install a mirror is to download and run the Bitbucket installer and select the **Install a new mirror** option. (Note this is the same installer that you use for standard Bitbucket Server and Data Center instances, the mirroring functionality is included in the same distribution.)

1. Go to www.atlassian.com/software/bitbucket/download and download the latest version of the Bitbucket installer onto the server where the mirror will reside.
2. Run the installer, and be sure to select **Install a new mirror**.
3. Complete the rest of the installation wizard.



Installing from an archive file

Alternatively, If you prefer not to use the Bitbucket installer, you can:

1. [Install Bitbucket Server from an archive file](#) but **do not start it**.
2. Add the following line to your `${BITBUCKET_HOME}/shared/bitbucket.properties`:

```
application.mode=mirror
```

2. Set up SSL on the mirror

You *must* configure your mirror to use HTTPS with a valid SSL certificate, and make secure access

mandatory.

SSL certificates are issued by a trusted third party Certificate Authority (CA), such as [VeriSign](#), [DigiCert](#) or [Thawte](#), which provide such services on a commercial basis. Atlassian does not provide such services or support their use.

Once you have your SSL certificate, you can configure your mirror to use it by following the steps described in [Proxying and securing Bitbucket Server: Securing access to Bitbucket Server using HTTPS](#).

You can choose whether to install your SSL certificate in a reverse proxy server or directly into Tomcat, either approach will work. But for performance, running your mirror behind a reverse proxy server such as Nginx or Haproxy and terminating SSL at the proxy is highly recommended. See [Proxying and securing Bitbucket Server](#) for more information.

Both your mirror and your primary (upstream) instance must be configured for HTTPS. The mirror setup wizard will not proceed if either the mirror or the primary instance does not have a valid SSL certificate.

3. Start the mirror

Then start your mirror.

```
sudo service atlbitbucket start
```

See [Starting and stopping Bitbucket Server](#) for more information on starting and stopping mirrors with the `service` command.

4. Set up the mirror

Once your mirror has started, you can go to the setup wizard by navigating to `https://<mirror-full-name>` in your browser, where `<mirror-full-name>` is your mirror's fully qualified name in the DNS. This name will be pre-filled as the **Mirror base URL** and cannot be changed after you click **Submit configuration**, so make sure your primary Bitbucket instance is able to reach the mirror with this name.

Configure mirror settings

Mirror name *

Users see this name when cloning a repository

Mirror base URL *

URL where users will access this mirror

Primary server URL *

The Bitbucket instance you want to mirror

Submit configuration

Descriptions of the fields:

Mirror name	The human-readable name of the instance. Users will see this name when selecting a mirror to clone from. Choose a name that your users will recognize and understand which name is the closest and fastest one for them.
Mirror base URL	The URL where the mirror can be accessed.
Primary server URL	The URL of the primary Bitbucket Data Center instance.

After configuring your details, click **Submit configuration**.

The setup wizard fields cannot be changed after you click **Submit configuration**, so fill them in carefully. If you do make a mistake and need to change a mirror's setup after it has been submitted, you need to stop the mirror, delete its home directory completely, and go back to step 2, [Set up SSL on the mirror](#).

Advanced: Automated setup

As an alternative to the setup wizard, you can automate the setup process without needing to navigate to the mirror in a browser. You can supply the <Mirror name>, <Mirror base URL>, and <Primary server URL> to your new mirror by adding the following properties to your `ITBUCKET_HOME}/shared/bitbucket.properties` file. These properties can only be applied on the initial start of a new mirror, not after the mirror has already been set up.

```
setup.displayName=<Mirror name>
setup.baseUrl=<Mirror base URL>
plugin.mirroring.upstream.url=<Primary server URL>
```

This can be useful if you deploy new mirrors using an automated process such as Puppet. See [Automated setup for Bitbucket Server](#) for more information.

Once you have successfully set up your mirror, it will guide you to log in to the primary instance as an administrator to approve it.

5. Approve a mirror request

Once a mirror has been installed and configured a request is sent to the primary Bitbucket Data Center instance.

An approved mirror will have access to all projects and repositories in the primary Bitbucket Data Center instance. Smart Mirroring has been designed to require secure communication throughout and restrict all functionality to the appropriate privilege level (e.g., system administrator) to ensure the security of all your sensitive information. It is still *your* responsibility to set up mirrors with the same stringent security practices as your primary Bitbucket Data Center instance. See the FAQ [How secure is Smart Mirroring?](#) for more information.

To approve a mirror request

1. Within the primary Bitbucket Data Center instance, go to **Admin > Mirrors**, and you will see the name of the mirror and that approval is required.
2. Click **Approve** to approve the mirror request and start syncing the projects and repositories of the primary Bitbucket Data Center instance.

6. Decide which projects to mirror

Once a mirror instance is approved you need to decide which projects to mirror. Go to **Admin > Mirrors** and

type in the name of a project in the search box. Do this for each projects you want to mirror. You can also choose to mirror all projects.

Mirroring all projects cannot be undone later

If you decide you do not want to mirror a project in the future you will have to completely remove and reinstall the mirror instance.

Troubleshooting

This section contains guidance on troubleshooting problems with installing a mirror. For further reference, read the [Bitbucket Data Center FAQ](#), which covers the most frequently asked questions about Bitbucket Data Center.

No valid HTTPS configuration

- Your primary Bitbucket Data Center instance and the mirror server must be configured for HTTPS access in order to use a mirror instance. See [Proxying and securing Bitbucket Server](#) for more information.
- The value for the `Primary server URL` field must include the "https://" prefix.

Synchronizing projects stays stuck on "Retrieving project information..."

- Ensure [SSH is enabled](#) and usable on the primary instance. See [Enabling SSH access to Git repositories in Bitbucket Server](#).
- Ensure your SSH base URL is set correctly in Server settings on the primary instance. See [Enabling SSH access to Git repositories in Bitbucket Server#SSHbaseURL](#).
- If appropriate, see [Setting up SSH port forwarding](#).

Incorrectly set Mirror base URL

- The property `Mirror base URL` can only be set once. If configured incorrectly you will need to delete your mirror and set it up again.

Git Large File Storage

Git Large File Storage (or LFS) is a new, open-source extension to Git that aims to improve handling of large files. It does this by replacing large files in your repository—such as graphics and videos—with simple text pointers.

Bitbucket Server includes an embedded LFS object store, allowing storage of large files without the need for an external object store.

Git LFS is disabled by default, on a per-repository basis, within Bitbucket Server.

On this page

- [Important Notices](#)
- [Enabling Git LFS for a repository](#)
- [Installing and using the Git LFS command line client](#)
- [Disabling Git LFS for Bitbucket Server](#)
- [Embedded Object Store](#)
- [Repository Forking](#)
- [Smart Mirroring](#)
- [Limiting Disk Space Usage](#)
- [Limiting Network Connections](#)

Important Notices

- SSL should be configured and the Base URL should be configured to a "https" URL. While Git LFS does support SSH remotes, the actual download & upload of LFS objects is via HTTP(S). SSL configuration is not a requirement but is *strongly recommended*.
- Please refer to [Bitbucket Server Supported Platforms](#) to ensure use of a compatible version of the Git LFS client.
- The user running the Bitbucket Server instance should not have the Git LFS client executable installed.

Enabling Git LFS for a repository

Git LFS is disabled by default within Bitbucket Server.

To enable Git LFS for Bitbucket Server

1. Go to **Repository settings > Large file storage (LFS)**
2. Tick the **Allow LFS** option.
3. Save your settings.

If the **Allow LFS** tick box is un-selectable (greyed out) this is because the instance administrator has disabled LFS support on the instance.

Installing and using the Git LFS command line client

Git LFS aims to integrate with the standard Git workflow as seamlessly as possible.

To push your first Git LFS files to an existing repository

1. Enable Git LFS support in Bitbucket Server for repository you wish to use Git LFS ([see above](#))
2. Download and install the [git-lfs command line client](#)
3. Install the Git LFS filters:

```
git lfs install
```

This adds the following lines to the `.gitconfig` file located in your home directory:

```
[filter "lfs"]
  clean = git-lfs clean %f
  smudge = git-lfs smudge %f
  required = true
```

The above change applies globally, so it is not necessary to run this for each repository you work with.

4. Choose the file types you would like LFS to handle by executing the `git lfs track` command. The `git lfs track` command creates or updates the `.gitattributes` file in your repository. Change to your cloned repository, then execute `git add` to ensure updates to the `.gitattributes` are later committed:

```
git lfs track "*.jpg"
git add .gitattributes
```

5. Add, commit, and push your changes as you normally would:

```
git add image.jpg
git commit -m "Added an image"
git push
```

When pushed, the Git tree updates to include a pointer to the actual file content. This pointer will include the SHA256 hash of the object and its size in bytes. For example:

```
oid
sha256:4fa32d6f9b1461c4a53618a47324e243e36ce7ceae72ad440cc811a7e
6881be1
size 1580060
```

The object itself will be uploaded to a separate location via the [Git LFS Batch API](#). Specifically, the object will be uploaded to an embedded LFS object store within your Bitbucket Server instance.

Disabling Git LFS for Bitbucket Server

Bitbucket Server is shipped with Git LFS support enabled on the instance by default, however it can be disabled. This will prevent upload or download of LFS objects for all repositories. It however does not delete existing LFS objects stored within Bitbucket Server.

To disable Git LFS on the entire Bitbucket Server instance

1. Log in to Bitbucket Server with sysadmin permissions.
2. Go to Admin > Server settings.
3. Untick the **Git LFS enabled** option.
4. Save your settings.

Embedded Object Store

Bitbucket Server includes an embedded LFS object store. Uploaded Git LFS objects are stored in the directory `$BITBUCKET_HOME/shared/data/git-lfs/storage`. This storage location cannot be changed, as for clustered deployments it is critical that LFS object storage reside on the shared filesystem so it is globally visible to all cluster nodes.

Repository Forking

LFS enabled repositories support forking and a full fork-based workflow including [pull requests](#) between forks and [fork synchronization](#). Forking is implemented through sharing of objects between fork and origin repositories and as such forks are lightweight. This means creating a fork is faster because objects are not duplicated when a fork is created.

Resolution of merge conflicts is slightly different from the non-LFS case, specifically where a merge conflict exists between two LFS objects. Since the Git repository simply contains an LFS pointer, conflict resolution must be performed with the conflicted pointer file.

Such a conflict might look like this:



Conflict: Added file to Source, Added same file to Target

This file is in a conflicted state. You will need to resolve the conflict manually before you can merge this pull request.

```
1 1 | version https://git-lfs.github.com/spec/v1
   2 + | <<<<<<<
2 3 | oid sha256:4fa32d6f9b1461c4a53618a47324e243e36ce7ceae72ad440cc811a7e6881be3
3 4 | size 1580067
   5 + | =====
   6 + | oid sha256:6bb99acfa40felc259397afea7fb18055cb25a8d4f0ae7dee9924fb76ec5ca72
   7 + | size 1083955
   8 + | >>>>>>>
```

Users are still required to resolve the merge conflict manually using the comment line Git client, as they normally would. The merge strategy, since this is only a pointer, can only be an *ours* or *theirs* based strategy,

keeping one or the two pairs (oid/size). When resolving the conflict, if the pointer file becomes corrupted it will not be recognized as an LFS file.

Smart Mirroring

The [Smart Mirroring](#) feature in Bitbucket Server supports mirroring of Git LFS objects as of Bitbucket Server 4.5. Mirroring of Git LFS objects is performed on-demand; that is, when a client requests download of a Git LFS object from the mirror node, the object will be streamed from the upstream node if it is not already available on the mirror. Subsequent downloads of the same object will be downloaded directly from the copy stored on the mirror.

Limiting Disk Space Usage

By default Bitbucket Server will not permit any further Git LFS uploads if the amount of free disk space in the filesystem that hosts the storage directory has less than 100 megabytes of free disk space. When a client attempts to upload (via a Git push) an LFS file that would result in the free disk-space threshold being exceeded, an error message indicating as such is sent to the client:

```
$ git push
Git LFS: (0 of 1 files, 1 skipped) 0 B / 348.59 MB, 348.59 MB skipped
[a39717817507d0ae3434a36347159e4970aec061c8c506f197c0eeadd2e8efe2]
Insufficient free space in store
error: failed to push some refs to
'https://bitbucket.example.com/bitbucket/scm/myproject/myrepo.git'
```

A warning will also be logged in the `atlassian-bitbucket.log` file. For example:

```
2016-01-01 18:00:00 WARN [http-nio-7990-exec-2] user
@G6I7R6x969x556x0 0:0:0:0:0:0:1 "POST
/scm/myproject/myrepo.git/info/lfs/objects/batch HTTP/1.1"
c.a.b.i.s.g.l.s.e.EmbeddedStoreAccessor Upload rejected due to
insufficient free space in store - Required: 1073741824 Free:
10485760
```

The free disk-space threshold can be tuned by adding the following property to the [Bitbucket Server config properties](#) file (this example tunes the threshold to one gigabyte, and note that this value is in bytes):

```
plugin.bitbucket-git-lfs.minimum.free.space=1073741824
```

A Bitbucket Server restart is required for the change to take effect.

Limiting Network Connections

Large Git LFS download or upload operations over very slow network links could take many minutes, or even hours. Bitbucket Server supports a finite number of HTTP connections via Tomcat (by default 150). If Git LFS were permitted to exhaust this connection pool then the user interface, Git hosting, and REST API access could be impacted. For this reason a default limit of 80 concurrent Git LFS connections are permitted. This limit can be increased or decreased if necessary by overriding the `throttle.resource.git-lfs=` setting in the [Bitbucket Server config properties](#) file. A Bitbucket Server restart is required for the change to take effect.

When a client makes a request to upload or download an object, and the request would exceed the maximum number of concurrent connections, a HTTP status 503 is returned, and this error is sent to the client:

The requested resource is busy and cannot service your request.
Please try again later

The `atlassian-bitbucket.log` file will also contain an associated warning:

```
2016-01-01 18:00:00 WARN [http-nio-7990-exec-3] username
@1HJ10F1x1115x417x1 0:0:0:0:0:0:1 "GET
/rest/git-lfs/storage/myproject/myrepo/3a9219fde5bc436a2fc37cdd38bdb8
478a210c7a49405dd9603ccdc95ed39613 HTTP/1.1"
c.a.s.i.t.SemaphoreThrottleService A [git-lfs] ticket could not be
acquired (0/80)
```

Copy of Bitbucket Server config properties

This page describes the Bitbucket Server system properties that can be used to control aspects of the behavior in Bitbucket Server. Create the `bitbucket.properties` file, in the shared folder of your [Bitbucket Server home directory](#), and add the system properties you need, use the standard format for Java properties files.

Note that the `bitbucket.properties` file is created automatically when you perform a [database migration](#).

Bitbucket Server must be restarted for changes to become effective.

Default values for system properties, where applicable, are specified in the tables below.

On this page:

- [Audit](#)
- [Authentication](#)
- [Avatars](#)
- [Backup](#)
- [Changesets](#)
- [Changeset indexing](#)
- [Commit graph cache](#)
- [Database](#)
- [Database pool](#)
- [Display](#)
- [Downloads](#)
- [Events](#)
- [Executor](#)
- [Features](#)
- [Hibernate](#)
- [JIRA Applications](#)
- [JMX](#)
- [Large File Storage](#)
- [Liquibase](#)
- [Logging](#)
- [Notifications](#)
- [Paging](#)
- [Password reset](#)
- [Process execution](#)
- [Pull requests](#)
- [Readme parsing](#)
- [Ref metadata](#)
- [Resource throttling](#)
- [SCM – Cache](#)
- [SCM – Git](#)
- [Server busy banners](#)
- [Setup automation](#)
- [SMTP](#)

- [SSH command execution](#)
- [SSH security](#)
- [Syntax highlighting](#)
- [Webhooks](#)

Audit

Property	Description
<code>audit.highest.priority.to.log=HIGH</code>	Defines the lowest priority audit events that will be logged. Accepted values are: HIGH, MEDIUM, LOW and NONE. Setting the value to HIGH will result in only HIGH level events being logged. NONE will cause no events to be logged. MEDIUM will only allow events with a priority of MEDIUM and HIGH to be logged. Refer to the levels for the various events. This does not affect events displayed in the Audit log screens for projects and repositories.
<code>audit.details.max.length=1024</code>	Defines the number of characters that can be stored as details for a single audit entry.
<code>plugin.bitbucket-audit.max.entity.rows=500</code>	The maximum number of entries a project or repository can have in the audit tables. This does not affect the data stored in the logs.
<code>plugin.bitbucket-audit.cleanup.batch.size=1000</code>	When trimming the audit entries table this is the maximum number of rows that will be trimmed in one transaction. Reduce this size if you are having issues with long running transactions. This does not affect the data stored in the logs.
<code>plugin.bitbucket-audit.cleanup.run.interval=24</code>	How often the audit tables will be checked to see if they need to be trimmed (in hours). This does not affect the data stored in the logs.

Authentication

See also [Connecting Bitbucket Server to Crowd](#).

Property	Description

<code>plugin.auth-crowd.sso.enabled=false</code>	Whether SSO support should be enabled or not. Regardless of this setting, SSO authentication will only be activated if a Crowd directory is configured in Bitbucket Server that is configured for SSO.
<code>plugin.auth-crowd.sso.session.lastvalidation=atl.crowd.sso.lastvalidation</code>	The session key to use when storing the Date value of the user's authentication.
<code>plugin.auth-crowd.sso.session.tokenkey=atl.crowd.sso.tokenkey</code>	The session key to use when storing the String value of the user's authentication token.
<code>plugin.auth-crowd.sso.session.validationinterval=3</code>	The number of minutes to cache authentication validation. If the session is expired, this value is used. If set to 0, the SSO session will be validated with the Crowd server for every HTTP request.
<code>plugin.auth-crowd.sso.http.max.connections=20</code>	The maximum number of HTTP connections in the connection pool for communicating with the Crowd server.

<code>plugin.auth-crowd.sso.http.proxy.host</code>	The name the proxy server use transport SOAP traf the Crowd server.
<code>plugin.auth-crowd.sso.http.proxy.port</code>	The connection port of the proxy serv (must be specified i proxy host specified).
<code>plugin.auth-crowd.sso.http.proxy.username</code>	The usern used to authentic with the pr server (if t proxy serv requires authentic
<code>plugin.auth-crowd.sso.http.proxy.password</code>	The passv used to authentic with the pr server (if t proxy serv requires authentic
<code>plugin.auth-crowd.sso.http.timeout=5000</code>	The HTTP connection timeout in millisecon used for communic with the C server. A \ of zero indicates t there is no connection timeout.
<code>plugin.auth-crowd.sso.socket.timeout=20000</code>	The socke timeout in millisecon You may v to override default val the latency the Crowd server is h

Avatars

Property	Description
<code>avatar.gravatar.default=mm</code>	The fallback URL for Gravatar avatars when a user does not have an acceptable avatar configured. This may be a URL resource, or a Gravatar provided default set. This configuration setting is DEPRECATED. It will be removed in Bitbucket Server 3.0. Use <code>avatar.url.default</code> instead.
<code>avatar.max.dimension=1024</code>	Controls the max height <i>and</i> width for an avatar image. Even if the avatar is with the acceptable file size, if its dimension exceed this value for height <i>or</i> width, it will be rejected. When an avatar is loaded by the server for processing, images with large dimensions may expand from as small a few kilobytes on disk to consume a substantially larger amount of memory, depending on how well the image data was compressed. Increasing this limit <i>substantially</i> increase the amount of memory used while processing avatars and may result in OutOfMemoryErrors. Value is in PIXELS.
<code>avatar.max.size=1048576</code>	Controls how large an avatar is allowed to be. Avatars larger than this are rejected and cannot be uploaded to the server, to prevent excessive disk usage. Value is in BYTES.
<code>avatar.temporary.cleanup.interval=1800000</code>	Controls how frequently temporary avatars are cleaned up. Any temporary avatars that have been uploaded are checked against their configured max age and removed from the file system if they are "too old". Value is in MILLISECONDS.
<code>avatar.temporary.max.age=30</code>	Controls how long a temporary avatar that has been uploaded is retained before it is automatically deleted. Value is in MINUTES.

<code>avatar.url.default=\${avatar.gravatar.default}</code>	Defines the fallback URL to be formatted into the <code>avatar.url.format.http</code> or <code>avatar.url.format.https</code> URL format for use when a user does not have an acceptable avatar configured. This value may be a URL or, if using Gravatar, it may be the identifier for one of Gravatar's default avatars. The default here falls back on the now-deprecated <code>avatar.gravatar.default</code> setting, which should ensure the value, if set, continues to work until it is removed in Bitbucket Server 3.0. At that time, this default will become "mm".
<code>avatar.url.format.http=http://www.gravatar.com/avatar/%1\$s.jpg?s=%2\$d&d=%3\$s</code>	Defines the default URL format for retrieving user avatars over HTTP. This default uses any G-rated avatar provided by the Gravatar service [http://www.gravatar.com] The following format parameters are available: %1\$s – the user's e-mail address, MD5 hashed, or "00000000000000000000000000000000" if the user has no e-mail. %2\$d – the requested avatar size. %3\$s – the fallback URL, URL-encoded which may be defined using "avatar.url.default". %4\$s – the user's e-mail address, not hashed, or an empty string if the user has no e-mail.
<code>avatar.url.format.https=https://secure.gravatar.com/avatar/%1\$s.jpg?s=%2\$d&d=%3\$s</code>	Defines the default URL format for retrieving user avatars over HTTPS. This default uses any G-rated avatar provided by the Gravatar service [http://www.gravatar.com] The following format parameters are available: %1\$s – the user's e-mail address, MD5 hashed, or "00000000000000000000000000000000" if the user has no e-mail. %2\$d – the requested avatar size. %3\$s – the fallback URL, URL-encoded which may be defined using "avatar.url.default". %4\$s – the user's e-mail address, not hashed, or an empty string if the user has no e-mail.

Backup

Property	Description
<code>backup.drain.database.timeout=60</code>	Defines the number of seconds Bitbucket Server will wait for connections to the database to drain and latch in preparation for a backup. Value is in SECONDS.

Changesets

Property	Description
<code>changeset.diff.context=10</code>	Defines the number of context lines to include around diff segments in changeset diffs.

Changeset indexing

These properties control how changesets are indexed when new commits are pushed to Bitbucket Server.

Property	Description
<code>indexing.max.threads=2</code>	Controls the maximum number of threads which are used to perform indexing. The resource limits configured below are not applied to these threads, so using a high number may negatively impact server performance.
<code>indexing.job.batch.size=250</code>	Defines the number of changesets which will be indexed in a single database transaction.
<code>indexing.job.queue.size=150</code>	Defines the maximum number of pending indexing requests. When this limit is reached, attempts to queue another indexing operation will be rejected.
<code>indexing.process.timeout.execution=3600</code>	Controls how long indexing processes are allowed to execute before they are interrupted, even if they are producing output or consuming input. Value is in SECONDS.

Commit graph cache

Property	Description
<code>commit.graph.cache.min.free.space=1073741824</code>	Controls how much space needs to be available on disk (specifically under <code><Bitbucket home directory>/caches</code>) for caching to be enabled. This setting ensures that the cache plugin does not fill up the disk. Value is in BYTES.

<code>commit.graph.cache.max.threads=2</code>	Defines the number of threads that will be used to create commit graph cache entries.
<code>commit.graph.cache.max.job.queue=1000</code>	Defines the maximum number of pending cache creation jobs.

Database

Database properties allow very specific configuration for your database connection parameters, which are set by Bitbucket Server during database setup and migration, and allow you to configure a database of your own. We don't expect that you will edit these, except in collaboration with Atlassian Support.

If none of the properties below are specified in `bitbucket.properties`, then the internal HSQL database will be used.

If the `jdbc.driver`, `jdbc.url`, `jdbc.password` and `jdbc.user` properties are specified in `bitbucket.properties` when the Setup Wizard runs after installing Bitbucket Server, then those values will be used, and the Setup Wizard will not display the database configuration screen.

Any other driver must be placed in `WEB-INF/lib` in order to use the associated database.

Warning: `jdbc.driver` and `jdbc.url` are available to plugins via the `ApplicationPropertiesService`. Some JDBC drivers allow the username and password to be defined in the URL. Because that property is available throughout the system (and will be included in STP support requests), that approach should not be used. The `jdbc.username` and `jdbc.password` properties should be used for these values instead.

Property	Description
<code>jdbc.driver=org.hsqldb.jdbcDriver</code>	The JDBC driver class that is used to connect to the database. The internal Bitbucket Server uses <code>org.hsqldb.jdbcDriver</code>. It is located in the Bitbucket Server home directory. Bitbucket Server bundles the following JDBC drivers: <code>org.postgresql.Driver</code> <code>com.microsoft.sqlserver.jdbc.Driver</code> (more info) <code>oracle.jdbc.driver.OracleDriver</code> The JDBC drivers for MySQL and Microsoft SQL Server (due to licensing restrictions) are not bundled. You must download and install the drivers yourself. See Bitbucket Server to MySQL for more information.
<code>jdbc.url=jdbc:hsqldb:\${bitbucket.home}/data/db;shutdown=true</code>	This is the JDBC url that Bitbucket Server uses to connect to the database. This should include the protocol (e.g. <code>jdbc:postgresql:</code>), the hostname, port, and database name. This string may vary depending on the database you are connecting to. Please refer to the documentation of the database you are connecting to for more information.
<code>jdbc.user=bitbucket</code>	This is the user that Bitbucket Server uses to connect to the database with. The user will have the necessary permissions to create tables and indexes, as well as to manage the entire database schema.

<code>jdbc.password=bitbucket</code>	The password that the user connect with.
<code>jdbc.ignoreunsupported=false</code>	Allows using a given database UNSUPPORTED . This is not documented, nor to be used as a mechanism to override the security event that it incorrectly blocks.

Database pool

These properties control the database pool. The pool implementation used is HikariCP. Documentation for these settings can be found at: <https://github.com/brettwooldridge/HikariCP/wiki/Configuration>

To get a feel for how these settings really work in practice, the most relevant classes in HikariCP are:

- `com.zaxxer.hikari.HikariConfig` Holds the configuration for the database pool and has documentation for the available settings.
- `com.zaxxer.hikari.pool.HikariPool` Provides the database pool and manages connections.
- `com.zaxxer.hikari.util.ConnectionBag` Holds references to open connections, whether in-use or idle.

Property	Description
<code>db.pool.size.idle=0</code>	Defines the number of connections the pool tries to keep idle. <i>The system can have more idle connections than the value configured here.</i> As connections are borrowed from the pool, this value is used to control whether the pool will eagerly open new connections to try and keep some number idle, which can help smooth ramp-up for load spikes. By default, the system does not eagerly open new idle connections. Connections will be opened as needed. Once opened, connections may <i>become</i> idle and will be retained for <code>db.pool.timeout.idle</code> seconds.
<code>db.pool.size.max=80</code>	Defines the maximum number of connections the pool can have open at once.
<code>db.pool.timeout.connect=15</code>	Defines the amount of time the system will wait when attempting to open a new connection before throwing an exception. The system may hang, during startup, for the configured number of seconds if the database is unavailable. As a result, the timeout configured here should not be generous. This value is in SECONDS.

<code>db.pool.timeout.idle=1750</code>	Defines the maximum period of time a connection may be idle before it is closed. In general, generous values should be used here to prevent creating and destroying many short-lived database connections (which defeats the purpose of pooling). Note: If an aggressive timeout is configured on the database server, a <i>more aggressive</i> timeout must be used here to avoid issues caused by the database server closing connections from its end. The value applied here should ensure the system closes idle connections before the database server does. This value needs to be less than <code>db.pool.timeout.lifetime</code> otherwise the idle timeout will be ignored. This value is in SECONDS.
<code>db.pool.timeout.leak=0</code>	Defines the maximum period of time a connection may be checked out before it is reported as a potential leak. <i>By default, leak detection is not enabled.</i> Long-running tasks, such as taking a backup or migrating databases, can easily exceed this threshold and trigger a false positive detection. This value is in MINUTES.
<code>db.pool.timeout.lifetime=30</code>	Defines the maximum <i>lifetime</i> for a connection. Connections which exceed this threshold are closed the first time they become idle and fresh connections are opened. This value is in MINUTES.

Display

Property	Description
<code>display.max.source.lines=20000</code>	Controls how many lines of a source file will be retrieved before a warning banner is shown that encourages the user is to download the raw file for further inspection. This property relates to <code>page.max.source.lines</code> (see Paging below) in that up to <code>(display.max.source.lines / page.max.source.lines)</code> requests will be made to view the page.

Downloads

Property	Description
<code>http.download.raw.policy=Smart</code>	Controls the download policy for raw content. Possible values are: - Insecure – allows all file types to be viewed in the browser. - Secure – requires all file types to be downloaded rather than viewed in the browser. - Smart – forces "dangerous" file types to be downloaded, rather than allowing them to be viewed in the browser. These options are case-sensitive and defined in <code>com.atlassian.http.mime.DownloadPolicy</code>.

Events

These properties control the number of threads that are used for dispatching asynchronous events. Setting this number too high can decrease overall throughput when the system is under high load because of the additional overhead of context switching. Configuring too few threads for event dispatching can lead to events being queued up, thereby reducing throughput. These defaults scale the number of dispatcher threads with the number of available CPU cores.

Property	Description
<code>event.dispatcher.core.threads=0.8*\${scaling.concurrency}</code>	The minimum number of threads that is available to the event dispatcher. The <code>\${scaling.concurrency}</code> variable is resolved to the number of CPUs that are available.
<code>event.dispatcher.max.threads=\${scaling.concurrency}</code>	The maximum number of event dispatcher threads. The number of dispatcher threads will only be increased when the event queue is full and <code>max.threads</code> has not been reached yet.
<code>event.dispatcher.queue.size=4096</code>	The number of events that can be queued. When the queue is full and no more threads can be created to handle the events, events will be discarded.
<code>event.dispatcher.keepAlive=60</code>	The time a dispatcher thread will be kept alive when the queue is empty and more than <code>core.threads</code> threads are running. Value is in SECONDS.

Executor

Controls the thread pool that is made available to plugins for asynchronous processing.

Property	Description
<code>executor.max.threads=\${scaling.concurrency}</code>	Specifies the maximum number of threads in the thread pool. When more threads are required than the configured maximum, the thread attempting to schedule an asynchronous task to be executed will block until a thread in the pool becomes available. The <code>\${scaling.concurrency}</code> variable is resolved to the number of CPUs that are available.

Features

Feature properties control high-level system features, allowing them to be disabled for the entire instance. Features that are disabled at this level are disabled *completely*. This means that instance-level configuration for a feature is overridden. It also means that a user's permissions are irrelevant; a feature is still disabled even if the user has the `SYS_ADMIN` permission.

Property	Description
<code>attachment.upload.max.size=10</code>	Controls the file size limit for individual attachments to pull request comments and descriptions. Value is in MB.
<code>feature.attachments=true</code>	Controls whether attachments can be added to pull request comments and descriptions.
<code>feature.auth.captcha=true</code>	Controls whether to require CAPTCHA verification when the number of failed logins is exceeded. If enabled, any client who has exceeded the number of failed logins allowed using either the Bitbucket Server web interface or the Git hosting interface will be required to authenticate in the Bitbucket Server web interface and successfully submit a CAPTCHA before continuing. Setting this to <code>false</code> will remove this restriction and allow users to incorrectly authenticate as many times as they like without penalty. ⚠ Warning: It is STRONGLY recommended you keep this setting enabled. Disabling it will have the following ramifications: - Your users may lock themselves out of any underlying user directory service (LDAP, Active Directory etc) because Bitbucket Server will pass through all authentication requests (regardless of the number of previous failures) to the underlying directory service. - For Bitbucket Server installations where you use Bitbucket Server for user management or where you use a directory service with no limit on the number of failed logins before locking out users, you will open Bitbucket Server or the directory service up to brute-force password attacks.
<code>feature.forks=true</code>	Controls whether repositories can be forked. This setting <i>supersedes and overrides</i> instance-level configuration. If this is set to <code>false</code> , even repositories which are marked as forkable cannot be forked.
<code>feature.personal.repos=true</code>	Controls whether personal repositories can be created. When set to <code>false</code> , personal repository creation is disabled globally in Bitbucket Server.
<code>feature.public.access=true</code>	Public access to Bitbucket Server allows unauthenticated users to be granted access to projects and repositories for specific read operations including cloning and browsing repositories. This is normally controlled by project and repository administrators but can be switched off system wide by setting this property to <code>false</code> . This can be useful in highly sensitive environments.

Hibernate

Property	Description
<code>hibernate.format_sql=false</code>	When <code>hibernate.show_sql</code> is enabled, this flag controls whether Hibernate will format the output SQL to make it easier to read.

<code>hibernate.jdbc.batch_size=20</code>	Controls Hibernate's JDBC batching limit, which is used to make bulk processing more efficient (both for processing and for memory usage).
<code>hibernate.show_sql=false</code>	Used to enable Hibernate SQL logging, which may be useful in debugging database issues. This value should generally only be set by developers.

JIRA Applications

Property	Description
<code>plugin.jira-integration.pullrequest.attribute.changesets.max=100</code>	Controls the maximum number of changesets to retrieve when retrieving attributes associated with changesets of a pull-request. This value should be between 50 and 1000 as Bitbucket Server will enforce an lower bound of 50 issues and an upper bound of 1000 issues.
<code>plugin.jira-integration.remote.page.max.issues=20</code>	Controls the maximum number of issues to request from a JIRA application. This value should be between 5 and 50 as Bitbucket Server will enforce a lower bound of 5 issues and an upper bound of 50 issues.

<code>plugin.jira-integration.remote.timeout.connection=5000</code>	The connection timeout duration in milliseconds for requests to JIRA applications. This timeout occurs if a JIRA application server does not answer. e.g. the server has been shut down. This value should be between 2000 and 60000 as Bitbucket Server will enforce a lower bound of 2000ms and an upper bound of 60000ms. Value is in MILLISECONDS.
<code>plugin.jira-integration.remote.timeout.socket=10000</code>	The socket timeout duration in milliseconds for requests to JIRA applications. This timeout occurs if the connection to a JIRA application has been stalled or broken. This value should be between 2000 and 60000 as Bitbucket Server will enforce a lower bound of 2000ms and an upper bound of 60000ms. Value is in MILLISECONDS.

JMX

Property	Description
<code>jmx.enabled=true</code>	Controls the publishing of Bitbucket Server specific statistics via JMX. See Enabling JMX counters for performance monitoring.

Large File Storage

Property	Description

<code>throttle.resource.git-lfs=80</code>	Limits the number of Git LFS file transfer operations which may be running concurrently. This is primarily intended to prevent Git LFS requests from consuming all available connections thereby degrading UI and Git hosting operation. This should be a fraction of the maximum number of concurrent connections permitted by Tomcat.
<code>throttle.resource.git-lfs.timeout=0</code>	Controls how long threads will wait for Git LFS uploads/downloads to complete when the system is already running the maximum number of concurrent transfers. It is recommended this be set to zero (i.e. don't block) or a few seconds at most. Since waiters still hold a connection, a non-zero wait time defeats the purpose of this throttle. This value is in seconds .

Liquibase

Property	Description
<code>liquibase.commit.block.size=10000</code>	The maximum number of changes executed against a particular Liquibase database before a commit operation is performed. Very large values may cause DBMS to use excessive amounts of memory when operating within transaction boundaries. If the value of this property is less than one, then changes will not be committed until the end of the change set.

Logging

Logging levels for any number of loggers can be set in the `bitbucket.properties` file using the following format:

```
logging.logger.<name>=<level>
```

For example, to configure all classes in the `com.atlassian.bitbucket` package to `DEBUG` level:

```
logging.logger.com.atlassian.bitbucket=DEBUG
```

To adjust the `ROOT` logger, you use the special name `ROOT` (case-sensitive):

```
logging.logger.ROOT=INFO
```

Notifications

Property	Description
<code>plugin.bitbucket-notification.batch.min.wait.minutes=10</code>	Controls the minimum time to wait for new notifications before sending the batch. This is the inactivity timeout. Value is in MINUTES.
<code>plugin.bitbucket-notification.batch.max.wait.minutes=30</code>	Controls the maximum time to wait for new notifications before sending the batch. This is the staleness timeout. Value is in MINUTES.
<code>plugin.bitbucket-notification.mail.max.comment.size=2048</code>	Controls the maximum allowed size of a single comment in characters (not bytes). Extra characters will be truncated.
<code>plugin.bitbucket-notification.mail.max.description.size=2048</code>	Controls the maximum allowed size of a single description in characters (not bytes). Extra characters will be truncated.
<code>plugin.bitbucket-notification.mentions.enabled=true</code>	Controls whether notifications for mentions are enabled.
<code>plugin.bitbucket-notification.max.mentions=200</code>	Controls the maximum number of allowed mentions in a single comment.
<code>plugin.bitbucket-notification.sendmode.default=BATCHED</code>	Controls the system default for notifications batching for users who have not set an explicit preference. Value is either BATCHED or IMMEDIATE.

Paging

These properties control the maximum number of objects which may be returned on a page, regardless of how many were actually requested by the user. For example, if a user requests `Integer.MAX_INT` branches on a page, their request will be limited to the value set for `page.max.branches`.

This is intended as a safeguard to prevent enormous requests from tying up the server for extended periods of time and then generating responses whose payload is prohibitively large. The defaults configured here represent a sane baseline, but may be overridden if necessary.

Property	Description
<code>page.max.branches=1000</code>	Maximum number of branches per page.
<code>page.max.changes=1000</code>	Maximum number of changes per page. Unlike other page limits, this is a hard limit; subsequent pages cannot be requested when the number of changes in a changeset exceeds this size.
<code>page.max.commits=100</code>	Maximum number of commits per page.
<code>page.max.diff.lines=10000</code>	Maximum number of segment lines (of any type, total) which may be returned for a single diff. Unlike other page limits, this is a hard limit; subsequent pages cannot be requested when a diff exceeds this size.
<code>page.max.directory.children=500</code>	Maximum number of directory entries which may be returned for a given directory.
<code>page.max.directory.recursive.children=100000</code>	Maximum number of file entries which may be returned for a recursive listing of a directory. A relatively high number as this is used by the file finder which needs to load the tree of files upfront.
<code>page.max.groups=1000</code>	Maximum number of groups per page.
<code>page.max.index.results=50</code>	Maximum number of changesets which may be returned from the index when querying by an indexed attribute. For example, this limits the number of changesets which may be returned when looking up commits against a JIRA application issue.
<code>page.max.projects=1000</code>	Maximum number of projects per page.
<code>page.max.repositories=1000</code>	Maximum number of repositories per page.
<code>page.max.source.length=5000</code>	Maximum length for any line returned from a given file when viewing source. This value truncates long lines. There is no mechanism for retrieving the truncated part short of downloading the entire file.
<code>page.max.source.lines=5000</code>	Maximum number of lines which may be returned from a given file when viewing source. This value breaks large files into multiple pages. See also Display above.
<code>page.max.tags=1000</code>	Maximum number of tags per page.
<code>page.max.users=1000</code>	Maximum number of users per page.
<code>page.max.pullrequests=100</code>	Maximum number of pull requests per page.
<code>page.scan.pullrequest.activity.size=500</code>	The size of the page Bitbucket Server should use when scanning activities.
<code>page.scan.pullrequest.activity.count=4</code>	The number of pages of activities Bitbucket Server should scan before giving up.

Password reset

Property	Description
<code>password.reset.validity.period=4320</code>	Controls how long a password reset token remains valid for. Default period is 72 hours. Value is in MINUTES.

Process execution

Property	Description
<code>process.timeout.execution=120</code> <code>process.timeout.idle=60</code>	Controls timeouts for external processes, such as Git and Hg. The idle timeout configures how long the command is allowed to run without producing any output. The execution timeout configures a hard upper limit on how long the command is allowed to run even if it is producing output. Values are in SECONDS. Using 0, or a negative value, disables the timeout completely. USE AT YOUR OWN RISK!

Pull requests

Property	Description
<code>plugin.bitbucket-git.pullrequest.merge.strategy.KEY.slug=no-ff</code>	Control the merge strategy for a repository (where KEY slug is the repository slug). Note that the URL for the repository is of the following form: <code>http://<bitbucketdomain>/projects/<PROJECTKEY>/re</code> Overrides project and global settings. Possible values are: <code>no-ff</code> – no fast-forward; the default setting. <code>ff</code> – allow fast-forward; will merge when necessary <code>ff-only</code> – require fast-forward; will never create a merge if a merge is required. <code>squash</code> – collapse all incoming commits into a single commit on the target branch; never create a merge. <code>squash-ff-only</code> – collapse all the incoming commits into a single commit directly to the target branch, never creating a merge if the source branch is fast-forward.

<code>plugin.bitbucket-git.pullrequest.merge.strategy.KEY=no-ff</code>	Control the merge strategy for a project (where KEY is overridden by repository settings). Overrides global settings. Is overridden by repository settings. Possible values are listed above.
<code>plugin.bitbucket-git.pullrequest.merge.strategy=no-ff</code>	Control the merge strategy globally. Is overridden by repository settings. Possible values are listed above.
<code>pullrequest.diff.context=10</code>	Defines the number of context lines to include around pull request diffs. By default, Git only includes 3 lines. The more context you include a bit more useful context around changes, until the context is implemented.
<code>pullrequest.rescope.changesets.display=5</code>	Defines the maximum number of changesets per type (added, removed) to display in a rescope activity.
<code>pullrequest.rescope.changesets.max=1000</code>	Defines the absolute maximum number of changesets to display when attempting to determine, for a given rescope activity, how many were added to or removed from a pull request. Adjusting this value has a significant memory footprint impact on the system. It is not recommended to be changed, but the option is provided here to support debugging.
<code>pullrequest.rescope.detail.threads=2</code>	Defines the maximum number of threads to use for processing pull request details. These threads perform the requisite processing for commits added and removed when a pull request is rescoped. Most rescopes do not add or remove any commits. Some rescopes delete commits during processing. The primary goal is to ensure that the diff is already been calculated when users try to view a pull request.
<code>pullrequest.rescope.drift.threads=4</code>	Defines the maximum number of threads to use when performing comment drift for a pull request during rescope. Higher numbers of threads mean higher throughput! Performing comment drift with a large number of merges to be created, which can be very I/O intensive. The number of merges running at the same time can significantly impact the speed of performing comment drift.

Readme parsing

Property	Description
<code>plugin.bitbucket-readme.max.size=65536</code>	Controls the maximum allowed size of a readme file to parse. Value is in BYTES.

Ref metadata

Property	Description
<code>ref.metadata.timeout=2</code>	Controls timeouts for retrieving metadata associated with a collection of refs from all metadata providers collectively. This value is in SECONDS.

<code>ref.metadata.max.request.count=100</code>	Controls the maximum number of refs that can be used in a metadata query.

Resource throttling

These properties define concurrent task limits for the ThrottleService, limiting the number of concurrent Git operations of a given type that may be run at once. This is intended to help prevent Bitbucket Server from overwhelming a server machine with running processes. Bitbucket Server has two settings to control the number of Git processes that are allowed to process in parallel: one for the web UI and one for the 'hosting' operations (pushing and pulling commits, and cloning a repository).

When the limit is reached for the given resource, the request will wait until a currently running request has completed. If no request completes within a configurable timeout, the request will be rejected.

When requests while accessing the Bitbucket Server UI are rejected, users will see either a 501 error page indicating the server is under load, or a popup indicating part of the current page failed.

When Git client 'hosting' commands (pull/push/clone) are rejected, Bitbucket Server does a number of things:

- Bitbucket Server will return an error message to the client which the user will see on the command line: "Bitbucket Server is currently under heavy load and is not able to service your request. Please wait briefly and try your request again"
- A warning message will be logged for every time a request is rejected due to the resource limits, using the following format:
"A [scm-hosting] ticket could not be acquired (12/12)"
- For five minutes after a request is rejected, Bitbucket Server will display a red banner in the UI to warn that the server is under load.

The hard, machine-level limits these are intended to prevent hitting are very OS- and hardware-dependent, so you may need to tune them for your instance of Bitbucket Server. When hyperthreading is enabled for the server CPU, for example, it is likely that the server will allow sufficient concurrent Git operations to completely bury the I/O on the machine. In such cases, we recommend starting off with a less aggressive default on multi-cored machines – the value can be increased later if hosting operations begin to back up. These defaults are finger-in-the-wind guesstimates (which so far have worked well).

Additional resource types may be configured by defining a key with the format `throttle.resource.<resource-name>`.

When adding new types, it is strongly recommended to configure their ticket counts explicitly using this approach.

Property	Description
<code>throttle.resource.scm-command=25</code>	Limits the number of operations that support the UI , such as <code>git diff</code> , <code>git blame</code> , or <code>git rev-list</code> , that can run concurrently. This is intended to prevent these SCM commands from competing with the running of push and pull operations.
<code>throttle.resource.scm-command.timeout=2</code>	Controls how long threads will wait for SCM commands to complete when the system is already running the maximum number of SCM commands. Value is in SECONDS.

<code>throttle.resource.scm-hosting=1.5*\${scaling.concurrency}</code>	Limits the number of SCM 'hosting' operations, such as <code>git clone</code> , <code>git push</code> and <code>git pull</code> over HTTP or SSH that may be running concurrently. This is intended primarily to prevent pulls, which can be very memory-intensive, from pinning a server's resources. There is limited support for mathematical expressions; <code>+</code> , <code>-</code> , <code>*</code> , <code>\</code> and <code>()</code> are supported. You can also use the <code>\${scaling.concurrency}</code> variable which is resolved to the number of CPUs that are available.
<code>throttle.resource.scm-hosting.timeout=300</code>	Controls how long threads will wait for SCM hosting operations to complete when the system is already running the maximum number of SCM hosting operations. Value is in SECONDS.
<code>throttle.resource.busy.message.timeout=5</code>	Controls how long a warning banner is displayed in the UI after a request is rejected due to excessive load. Value is in MINUTES. Using 0, or a negative value, disables displaying the banner. This is deprecated and replaced by <code>server.busy.on.ticket.rejected.within</code> . It is due to be removed in Bitbucket Server 3.0.

SCM – Cache

See [Scaling Bitbucket Server for Continuous Integration performance](#) for more information about using the SCM Cache Plugin for Bitbucket Server.

Property	Description
<code>plugin.bitbucket-scm-cache.expiry.check.interval=300</code>	Controls how frequently expired caches are checked and deleted from disk. Value is in SECONDS.
<code>plugin.bitbucket-scm-cache.minimum.free.space=1073741824</code>	Controls how much space needs to be available on disk (specifically under <code><Bitbucket home directory>/cache s</code>) for caching to be enabled. This setting ensures that the cache plugin does not fill up the disk. Value is in BYTES.
<code>plugin.bitbucket-scm-cache.protocols=HTTP,SSH</code>	Controls which protocols caching is applied to. The HTTP value encapsulates both <code>http</code> and <code>https</code> .

<code>plugin.bitbucket-scm-cache.refs.enabled=false</code>	Controls whether ref advertisement operations are cached.
<code>plugin.bitbucket-scm-cache.refs.ttl=60</code>	Controls how long the caches for ref advertisements are kept around when there no changes to the repository. Caches are automatically invalidated when someone pushes to a repository or when a pull request is merged. Time is in SECONDS.
<code>plugin.bitbucket-scm-cache.upload-pack.enabled=true</code>	Controls whether clone operations are cached.
<code>plugin.bitbucket-scm-cache.upload-pack.ttl=14400</code>	Controls how long the caches for clone operations are kept around when there no changes to the repository. Caches are automatically invalidated when someone pushes to a repository or when a pull request is merged. Time is in SECONDS.

SCM – Git

Property	Description
<code>plugin.bitbucket-git.path.executable=git</code>	Defines the default path to the Windows machines, the .exe s configured value automatically general, "git" should be an acc platform, here, assuming that i runtime user's PATH. With the new path searching p DefaultGitBinaryHelper, setting unnecessary, as the plugin will This is left here purely for docu explicit path.
<code>plugin.bitbucket-git.path.libexec=</code>	Defines the path to the Git libe the git-core directory). This pat executable and is used for fork git-http-backend. If this value is directly fork out those process unnecessary fork (git -> git-http improve scalability.
<code>plugin.bitbucket-git.backend.http.buffer.size=32768</code>	Defines the buffer size in bytes marshaling data between the C socket.

<code>plugin.bitbucket-git.backend.ssh.buffer.size=32768</code>	Defines the buffer size in bytes marshaling data between the C socket.
<code>plugin.bitbucket-git.backend.timeout.idle=1800</code>	Defines the idle timeout for push, a limit to how long the operation can execute without either producing output or receiving input. The default value is 30 minutes. This value is in SECONDS.
<code>plugin.bitbucket-git.backend.timeout.execution=86400</code>	Defines the execution timeout for push, applying a hard limit to how long the operation can run even if it is producing output. The default value is 1 day. This value is in SECONDS.
<code>plugin.bitbucket-git.diff.renames=copies</code>	Defines whether copy and/or rename are performed. By default, both are performed. Only files modified in the source are considered as rename or copy to avoid overhead. The possible settings are: • <code>copy</code> or <code>copies</code> – applies • <code>rename</code> or <code>renames</code> – applies • <code>off</code> – disables rename and copy When using <code>copy</code> or <code>copies</code> , the setting should be suffixed with a "+" to use <code>--diff-flags</code> . This setting should be used with caution as it is expensive. It considers every file as a copy even if it is not modified in the source. This is not the case for origins for copies. When copy and/or rename detection is disabled, <code>plugin.bitbucket-git.diff.renames</code> is used to control the similarity index required for a copy or rename.
<code>plugin.bitbucket-git.diff.renames.threshold=50</code>	Defines the threshold, as a percentage, for a file to be detected as a rename or a copy. The default threshold applied is 50% (0.5). Git diff and Git diff-tree <i>do not</i> use the threshold (only) for the threshold. They ignore the threshold and apply the default 50% threshold. If the threshold of 100 will be applied, the configured threshold that is 0, 1, or as 1.

<code>plugin.bitbucket-git.environment.variablesize=2000</code>	Defines the maximum number added to a single environment operating systems (and even c same operating system) have they apply to environment var intended to be low enough to v platforms out of the box, but st usable. It is configurable in cas on some platform.
<code>plugin.bitbucket-git.pullrequest.merge.auto.forceadd=false</code>	Defines whether conflicted file: index using Git add --force By default, this behavior is off. However, when merging across .Gitignore settings, enabling system to create a conflicted d the common ancestor will be s Note: This value has <i>no effect</i> merges. It is <i>only</i> applied durin for producing a pull request's c
<code>plugin.bitbucket-git.pullrequest.merge.auto.timeout=120</code>	Defines the maximum amount to perform a merge to support allowed to execute or idle. Bec generally do not produce output timeout. This value is in SECONDS.
<code>plugin.bitbucket-git.pullrequest.merge.real.timeout=300</code>	Defines the maximum amount to merge a pull request is allow Because the commands used output, there is no separate idl This value is in SECONDS.
<code>plugin.bitbucket-git.repository.size.timeout=75</code>	Defines the maximum amount the size of a single repository. repositories and/or remote stor value. This value is in MILLISECOND

Server busy banners

Property	Description
<code>server.busy.on.ticket.rejected.within=5</code>	Controls how long a warning banner is displayed in the UI after a request is rejected due to excessive load. Value is in MINUTES. Using 0, or a negative value, disables displaying the banner.
<code>server.busy.on.queue.time=60</code>	Controls how long requests need to be queued before they cause a warning banner to appear. Value is in SECONDS. Using 0, or a negative value, disables displaying the banner.

Setup automation

If these properties are specified in `bitbucket.properties` when the Setup Wizard runs after installing Bitbucket Server, then those values will be used, and the Setup Wizard will not display the corresponding configuration screens.

You can use these properties to automate Bitbucket Server setup and remove the need to interact with the Bitbucket Server Setup Wizard when provisioning Bitbucket Server. See [Automated setup for Bitbucket Server](#).

Property	Description
<code>setup.displayName=displayName</code>	The display name for the Bitbucket Server instance.
<code>setup.baseUrl= https://bitbucket.yourcompany.com</code>	The base URL to use for the Bitbucket Server instance.
<code>setup.license=AAAB...\u000a1ev...\u000aA4N...</code>	The Bitbucket Server license. Use the the <code>\u000a</code> character to <i>not</i> break the license over multiple lines.
<code>setup.sysadmin.username=username</code>	Credentials for the system admin account.
<code>setup.sysadmin.password=password</code>	
<code>setup.sysadmin.displayName=John Doe</code>	The display name for the system admin account.
<code>setup.sysadmin.emailAddress=sysadmin@yourcompany.com</code>	The email address for the system admin account.

SMTP

Property	Description
<code>mail.timeout.connect=60</code> <code>mail.timeout.send=60</code> <code>mail.test.timeout.connect=30</code> <code>mail.test.timeout.send=30</code>	Controls timeouts for establishing an SMTP connection and sending an e-mail. Shorter timeouts should be applied for when sending test e-mails, as the test occurs in user time. Values are in SECONDS.
<code>mail.error.pause.log=300</code>	Controls how frequently logs will go to the standard log file about mail sending errors. All errors are logged to the <code>atlassian-bitbucket-mail.log</code> file, but Bitbucket Server will periodically log a warning to the standard log file if there are errors sending messages. Value is in SECONDS.
<code>mail.error.pause.retry=5</code>	Controls how long Bitbucket Server will wait before retrying to send a message if an error occurs. Value is in SECONDS.

<code>mail.threads=1</code>	Controls the number of threads to use for sending emails. Setting this to a higher value will put greater load on your mail server when Bitbucket Server generates a lot of emails, but will make Bitbucket Server clear its internal queue faster.
<code>mail.max.message.size=1048576</code>	Controls the maximum allowed size of a single mail message, which is the sum of the subject and body sizes. Value is in BYTES.
<code>mail.max.queue.size=157286400</code>	Controls the maximum allowed size for the mail queue (any new message will be rejected if the mail queue reaches that size). Value is in BYTES.

SSH command execution

Property	Description
<code>plugin.ssh.command.timeout.idle=86400</code>	Controls timeouts for all SSH commands, such as those that service Git and hg operations over SSH. The idle timeout configures how long the command is allowed to run without writing any output to the client. For SCM commands, the <code>plugin.*.backend.timeout.idle</code> properties defined above will be applied to the underlying process. The default value is 1 day. Value is in SECONDS.

SSH security

Property	Description
<code>plugin.ssh.disabled.ciphers</code>	Controls which default ciphers are disabled when executing all SSH commands. Non existent ciphers are ignored. Names are case sensitive. Example value: <code>arcfour128,3des-cbc</code> To enable additional ciphers see the KB article Disable default SSH algorithms .
<code>plugin.ssh.disabled.key.exchanges</code>	Controls which default key exchange algorithms are disabled when executing all SSH commands. Non existent key exchange algorithms are ignored. Names are case sensitive. Example value: <code>ecdh-sha2-nistp256,ecdh-sha2-nistp384</code> To enable additional key exchange algorithms see the KB article Disable default SSH algorithms .

<code>plugin.ssh.disabled.macs</code>	Controls which default macs are disabled when executing all SSH commands. Non existent macs are ignored. Names are case sensitive. Example value: <code>hmac-sha1-96, hmac-md5-96, hmac-md5</code> To enable additional macs see the KB article Disable default SSH algorithms.

Syntax highlighting

See [Configuring syntax highlighting for file extensions](#) for more information.

Bitbucket Server applies syntax highlighting to diffs as well as source files.

Property	Description
<code>syntax.highlighter.<MIME type>.executables=exe1, exe2</code>	Controls the language highlighter used for a given set of hashbang executables. The <MIME type> refers to the MIME type CodeMirror uses.
<code>syntax.highlighter.<MIME type>.extensions=ext1, ext2</code>	Controls the language highlighter used for a given set of file extensions. The <MIME type> refers to the MIME types CodeMirror uses.

Webhooks

See [POST service webhook for Bitbucket Server](#) for more information.

Property	Description
<code>plugin.com.atlassian.bitbucket.plugin.hook.threadPoolCoreSize=2</code>	Core size of thread pool – the default number of concurrent hooks notifications.
<code>plugin.com.atlassian.bitbucket.plugin.hook.threadPoolMaxSize=3</code>	Maximal size of thread pool – the maximum number of concurrent hooks notifications.

<code>plugin.com.atlassian.bitbucket.plugin.hook.queueSize=1024</code>	The maximum size of the queue which holds queued requests that are yet to be sent. When this size is exceeded the oldest unsent message will be dropped and a warning message logged.
<code>plugin.com.atlassian.bitbucket.plugin.hook.connectionTimeout=10000</code>	Connection timeout for hook request in MILLISECONDS. When the connection times out a warning message will be logged.
<code>plugin.com.atlassian.bitbucket.plugin.hook.changesetsLimit=500</code>	Limit of maximum count of changesets that will be sent in the POST data for a single ref change.
<code>plugin.com.atlassian.bitbucket.plugin.hook.changesLimit=100</code>	Limit of maximum count of changes for a single changeset in the POST data.

Updating your Bitbucket Server license details

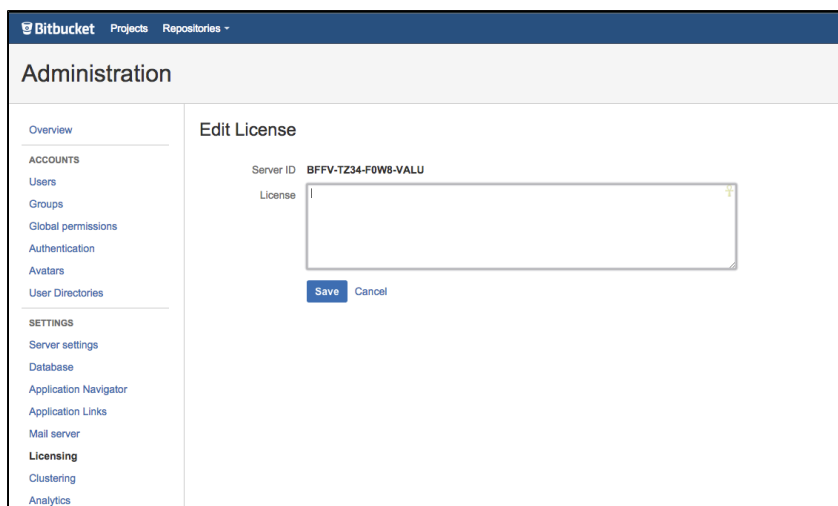
When you upgrade or renew your Bitbucket Server license, you will receive a new license key. You will need to update your Bitbucket Server server with the new license key.

 You can access your license key at <http://my.atlassian.com>

To update your Bitbucket Server license key:

1. Log in to Bitbucket Server as an **Admin**.
2. Go to **Administration > Settings > Licensing**.
3. Click on **License** and paste your new license into this box.

 You can retrieve existing licenses or generate an evaluation one by clicking the **My Account** link.
4. Click **Add** to update the Bitbucket Server installation with the new license.



i Need more information about licensing or want to find out more about starter licenses? See [Licensing FAQ](#) and [Starter Licenses](#).

Git resources

Get Git

See [Installing and upgrading Git](#)

On this page:

- [Get Git](#)
- [Learning Git](#)
- [Getting started](#)
- [Git cheat sheets and other resources](#)
- [Git .mailmap](#)

Learning Git

[Git Tutorials and Training](#)

[Basic Git commands](#)

Getting started

One "gotcha" when starting with Git is the way in which it pushes branches by default. On older versions of Git, pushing without arguments would push *all* branches that have the same name both locally and remotely. This can result in unexpected behavior if you have old branches that complain when the remote branch is updated. It can even be quite dangerous if you do a force push and it reverts changes on the server. You can see the current value by running:

```
git config push.default
```

If this value is blank or 'matching', it is our recommendation that you reconfigure it to use 'upstream'.

```
git config --global push.default upstream
```

There has been some [discussion](#) around changing the default behavior of Git.

Git cheat sheets and other resources

<http://rogerdudler.github.com/git-guide/>

<http://byte.kde.org/~zrusin/git/git-cheat-sheet-medium.png>

<http://nvie.com/posts/a-successful-git-branching-model/>

<http://zrusin.blogspot.com.au/2007/09/git-cheat-sheet.html>

<http://ndpsoftware.com/git-cheatsheet.html#loc=workspace;>

<http://blog.fournova.com/2011/06/git-cheat-sheet/>

<http://jan-krueger.net/development/git-cheat-sheet-extended-edition>

Git .mailmap

The Git `.mailmap` feature is useful locally, and in Bitbucket Server repositories, to map multiple commit identities to the one Bitbucket Server user – this can be used to tidy up your Git histories.

The [Git documentation](#) for `.mailmap` has configuration details (see the "MAPPING AUTHORS" section).

Basic Git commands

Here is a list of some basic Git commands to get you going with Git.

For more detail, check out the [Atlassian Git Tutorials](#) for a visual introduction to Git commands and workflows, including examples.

Git task	Notes	Git commands
Tell Git who you are	Configure the author name and email address to be used with your commits. Note that Git strips some characters (for example trailing periods) from <code>user.name</code> .	<code>git config --global user.name "Sam Smith"</code> <code>git config --global user.email sam@example.com</code>
Create a new local repository		<code>git init</code>
Check out a repository	Create a working copy of a local repository:	<code>git clone /path/to/repository</code>
	For a remote server, use:	<code>git clone username@host:/path/to/repository</code>
Add files	Add one or more files to staging (index):	<code>git add <filename></code> <code>git add *</code>
Commit	Commit changes to head (but not yet to the remote repository):	<code>git commit -m "Commit message"</code>
	Commit any files you've added with <code>git add</code> , and also commit any files you've changed since then:	<code>git commit -a</code>

Push	Send changes to the master branch of your remote repository:	<code>git push origin master</code>
Status	List the files you've changed and those you still need to add or commit:	<code>git status</code>
Connect to a remote repository	If you haven't connected your local repository to a remote server, add the server to be able to push to it:	<code>git remote add origin <server></code>
	List all currently configured remote repositories:	<code>git remote -v</code>
Branches	Create a new branch and switch to it:	<code>git checkout -b <branchname></code>
	Switch from one branch to another:	<code>git checkout <branchname></code>
	List all the branches in your repo, and also tell you what branch you're currently in:	<code>git branch</code>
	Delete the feature branch:	<code>git branch -d <branchname></code>
	Push the branch to your remote repository, so others can use it:	<code>git push origin <branchname></code>
	Push all branches to your remote repository:	<code>git push --all origin</code>
	Delete a branch on your remote repository:	<code>git push origin :<branchname></code>
Update from the remote repository	Fetch and merge changes on the remote server to your working directory:	<code>git pull</code>
	To merge a different branch into your active branch:	<code>git merge <branchname></code>
	View all the merge conflicts: View the conflicts against the base file:	<code>git diff</code> <code>git diff --base <filename></code> <code>git diff <sourcebranch> <targetbranch></code>
	Preview changes, before merging:	
	After you have manually resolved any conflicts, you mark the changed file:	<code>git add <filename></code>

Tags	You can use tagging to mark a significant changeset, such as a release:	<code>git tag 1.0.0 <commitID></code>
	CommitID is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:	<code>git log</code>
	Push all tags to remote repository:	<code>git push --tags origin</code>
Undo local changes	If you mess up, you can replace the changes in your working tree with the last content in head: Changes already added to the index, as well as new files, will be kept.	<code>git checkout -- <filename></code>
	Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this:	<code>git fetch origin</code> <code>git reset --hard origin/master</code>
Search	Search the working directory for <code>foo()</code> :	<code>git grep "foo()"</code>

Bitbucket Server FAQ

On this page:

- The Bitbucket Server product
 - Q: Why did you create a new product for Git repository management? Couldn't you build this into FishEye?
 - Q: What about Git repository management in FishEye and Crucible?
 - Q: Does FishEye require Bitbucket Server? Does Bitbucket Server require FishEye? Can they be used together?
 - Q: Will Bitbucket Server be available for Atlassian Cloud?
 - Q: What is the difference between Bitbucket Server and Bitbucket Data Center?
- Repositories
 - Q: Does Bitbucket Server support Mercurial (Hg)? What about other version control systems?
- Integration
 - Q: Does Bitbucket Server work with JIRA Software? If so, what version of JIRA Software do I need to run Bitbucket Server?
 - Q: Will Bitbucket Server integrate with any other Atlassian Tools? Crowd? Bitbucket? SourceTree?
- Licensing
 - Q: Does my Bitbucket Server license have to match the number of users in my external directory (LDAP, Active Directory, Crowd or JIRA Software)?
 - Q: The number of users in my instance has exceeded my license count. Will Bitbucket Server still work properly?
- Data recovery and backups
 - Q: Can I restore the .tar file created by the backup client into a database that is different from my original one (i.e. Oracle -> MySQL, etc.)?
 - Q: I forgot the user/password for my old database schema. How will I perform the restore? How does it work?
 - Q: What is the difference between the parameter pairs "bitbucket.user & bitbucket.password" and "jdbc.user & jdbc.password"?
 - Q: I backed up Bitbucket Server of a particular version. Can I restore Bitbucket Server to a newer release version?
- Troubleshooting
 - Q: I'm getting a "broken pipe" error when pushing my commits.

Related pages:

- [Bitbucket Data Center FAQ](#)
- [Bitbucket Server Knowledge Base Home](#)
- [Support policies](#)

Child pages:

- [Bitbucket rebrand FAQ](#)
- [How do I change the external database password](#)
- [Bitbucket Server home directory](#)
- [Raising a request with Atlassian Support](#)
- [Support policies](#)
- [Building Bitbucket Server from source](#)
- [Contributing to the Bitbucket Server documentation](#)
- [Collecting analytics for Bitbucket Server](#)
- [Bitbucket Server EAP - How to update your add-on](#)

The Bitbucket Server product

Q: Why did you create a new product for Git repository management? Couldn't you build this into FishEye?

A: In FishEye 2.7 we added basic capabilities to host and manage Git repositories within FishEye. However, as we were planning future releases, we realized that the architecture of FishEye, built to index, browse and search across various SCMs, was not adequate for a DVCS repository management tool.

Therefore we have made the decision to build a new product, with a clear focus: hosting and managing Git repositories. Instead of a "Jack of all trades", we will have two products that are focused on 2 very different tasks:

1. Bitbucket Server – Host, manage and collaborate on Git repositories, and
2. FishEye – Track, search and browse Subversion, Perforce, Git, Mercurial and CVS repositories in one place.

Q: What about Git repository management in FishEye and Crucible?

A: *Internally managed* Git repositories were deprecated in FishEye and Crucible 2.8, and support for these was removed for the FishEye and Crucible 3.2 releases. We encourage those interested in Git repository management to check out Bitbucket Server.

FishEye and Crucible will continue to deliver new features and enhancements to help users browse, search, review and visualize across different Version Control Systems *including Git*, Subversion, Mercurial, Perforce and CVS.

Q: Does FishEye require Bitbucket Server? Does Bitbucket Server require FishEye? Can they be used together?

A: FishEye and Bitbucket Server are two separate standalone products that do not require each other.

If you are using multiple source code management systems (SCM) at your organization it makes sense to use both FishEye and Bitbucket Server. While you are managing your Git repositories with Bitbucket Server, you can use FishEye to browse, search and reference code from other SCMs including Subversion.

Also, if you are using Git, Bitbucket Server will provide your Git repository management, and FishEye will be a central place to keep track of changes and search for code across your repositories.

Q: Will Bitbucket Server be available for Atlassian Cloud?

A: Bitbucket Server will not be available in Atlassian Cloud. If you are looking for a distributed version control solution to use with Atlassian Cloud, we recommend using [Bitbucket](#), our cloud-based Git and Mercurial source code hosting solution. Bitbucket connects to Atlassian Cloud via the [JIRA DVCS connector](#).

Q: What is the difference between Bitbucket Server and Bitbucket Data Center?

Bitbucket Server is a single instance of Bitbucket Server running on a single machine. It can only handle as much load as a single machine is capable of handling before performance degrades, and if the machine goes down for any reason (for example, hardware failure, network fault, or planned maintenance), then Bitbucket Server is unavailable to users for the duration of the downtime.

Bitbucket Data Center, on the other hand, looks like a single instance of Bitbucket Server to users, but under the hood consists of a cluster of multiple machines ("cluster nodes") each running the Bitbucket Server web application, behind a load balancer. This provides important benefits over Bitbucket Server:

- **Performance at scale:** A cluster of many machines running Bitbucket Server can handle more load than a single machine.
- **High availability:** If one cluster node goes down, then the remaining cluster node(s) can continue servicing requests so users should see little or no loss of availability.
- **Instant scalability:** You can rapidly provision extra capacity without downtime.

For more information see [Bitbucket Data Center](#) and the [Bitbucket Data Center FAQ](#).

Repositories

Q: Does Bitbucket Server support Mercurial (Hg)? What about other version control systems?

A: Currently Bitbucket Server does not support Mercurial. We will be gauging demand for Mercurial support as

we move forward - [BSERV-2469](#) - Include Mercurial (Hg) support

[OPEN](#)

Integration

Q: Does Bitbucket Server work with JIRA Software? If so, what version of JIRA Software do I need to run Bitbucket Server?

A: Bitbucket Server works with JIRA 4.3+. However, you will require the latest version of the JIRA/FishEye plugin to view commits in JIRA Software. See our documentation on [JIRA Software integration](#).

Q: Will Bitbucket Server integrate with any other Atlassian Tools? Crowd? Bitbucket? SourceTree?

A: Bitbucket Server currently integrates with JIRA Software, SourceTree DVCS Mac client and Crowd user management solution. You can also connect to Bitbucket Server via Bamboo to run your builds and deployments and we are planning even tighter integrations in the future.

Licensing

Q: Does my Bitbucket Server license have to match the number of users in my external directory (LDAP, Active Directory, Crowd or JIRA Software)?

A: No. You can control which users in your external directory have access to Bitbucket Server, so that the license limit is not exceeded. A user is by [definition](#) any account that has permission to log into the Bitbucket Server application. If you synchronize Bitbucket Server with an external user directory, you can grant access to Bitbucket Server to a subset of users, so as to stay below your license limit. The [Global permissions](#) page explains in detail how to manage login rights for users and groups in Bitbucket Server.

Q: The number of users in my instance has exceeded my license count. Will Bitbucket Server still work properly?

A: As stated in the [Global permissions](#) document, any user assigned "Bitbucket Server User" permission or higher, granted to the individual or via a group, will count towards the license limit. Bitbucket Server will not allow you to grant the "Bitbucket Server User" permission if this will exceed the license limit while manually adding users using Bitbucket Server UI. However, if you happen to exceed the license limit by connecting your Bitbucket Server instance to a User Directory that contains more users than you license allows you to have, Bitbucket Server will display a banner with the content below:

You have more users than your license allows.

Users will not be able to push commits to repositories until you restrict the number of active users to match your license or you upgrade your current license.

To fix that bear in mind that users don't have to be removed from the database in order to reduce their license count. You merely have to make sure that they no longer have access to Bitbucket Server via the "Bitbucket Server User" permission through individual or group assignments.

Data recovery and backups

Q: Can I restore the .tar file created by the backup client into a database that is different from my original one (i.e. Oracle -> MySQL, etc.)?

A: Yes you can, as long as you specify all the `jdbc` parameters (`jdbc.override`, `jdbc.driver`, etc.) when running the restore. Please read [Restoring Bitbucket Server into a newly created DB](#) for more details.

Q: I forgot the user/password for my old database schema. How will I perform the restore? How does it work?

A: As described in [Restoring Bitbucket Server into a newly created DB](#), the restore client will only restore into an empty home directory and an **empty database**. The new database should be configured following the instructions in [Connecting Bitbucket Server to an external database](#) and its sub-page that corresponds to your database type. If you want to use a different type of database or a different user/password, you just need to specify all the jdbc parameters (`jdbc.override`, `jdbc.driver`, etc) when running the restore.

Q: What is the difference between the parameter pairs "`bitbucket.user` & `bitbucket.password`" and "`jdbc.user` & `jdbc.password`"?

A: `bitbucket.user` and `bitbucket.password` hold the credentials for a Bitbucket Server sys admin user. They are only used during the **backup** procedure so that the backup client can lock Bitbucket Server and instruct Bitbucket Server to perform the database-agnostic backup. The backup client does not need database credentials because the Bitbucket Server system performs the database backup.

`jdbc.user` and `jdbc.password` are only used during the **restore** procedure when `jdbc.override` is set to `true`. They are used to connect to the newly-installed database.

Q: I backed up Bitbucket Server of a particular version. Can I restore Bitbucket Server to a newer release version?

A: No. You need to use the same Bitbucket Server binary as the one originally used to back up your instance. Note that using an older Bitbucket Server binary will result in an error – downgrades are not possible. See [Using the Bitbucket Server Backup Client](#) for details on the backup/restore procedure.

Once you have restored Bitbucket Server, you can upgrade to a newer version of Bitbucket Server following the instructions in the [Bitbucket Server upgrade guide](#).

Troubleshooting

Q: I'm getting a "broken pipe" error when pushing my commits.

A: This error occurs when the amount of data you're trying to push in one go exceeds Git's http post buffer. Just run the following command to increase it to 500MB.

```
git config http.postBuffer 524288000
```

See [Git push fails with 'fatal: The remote end hung up unexpectedly'](#).

Bitbucket rebrand FAQ

We have unified our Git products under the Bitbucket name. With Bitbucket, now you have a range of options that can be adopted by teams of all sizes and requirements: [Bitbucket Cloud](#) (previously known as Bitbucket), [Bitbucket Server](#) (previously known as Stash) and [Bitbucket Data Center](#) (previously known as Stash Data Center). We hope the following answers some of the more common questions this might raise.

- [Why did we rebrand Stash to Bitbucket Server?](#)
- [What is the impact on my plug-ins? Do I need to reinstall them?](#)
- [How do I raise support issues?](#)
- [Are there any feature differences between deployment options?](#)
- [Why aren't the Server and Cloud features the same?](#)
- [Can I export my Bitbucket repositories between services?](#)
- [Will Bitbucket Server support Mercurial?](#)
- [Where do I go if I have any more questions?](#)
- [Can I develop Bitbucket Connect add-ons for Bitbucket Server?](#)
- [Can I develop P2 add-ons for Bitbucket Cloud?](#)
- [Why is there no free pricing tier for Bitbucket Server?](#)

- [Where can I find more information about Mirroring and Large File Support on Bitbucket Server?](#)
- [Where can I find more information about Projects and Build Status on Bitbucket Cloud?](#)

Why did we rebrand Stash to Bitbucket Server?

To make it easier for you to find a collaborative code management solution that best meets your needs, we have unified our Git products under the Bitbucket name.

- **Bitbucket Server** (previously known as Stash) fits into any enterprise environment because it can be deployed within existing infrastructure or on hosted infrastructure providers such as Amazon Web Services.
- **Bitbucket Cloud** (previously known as Bitbucket) is a fully managed, multi-tenant service used by organizations that do not want to deal with the overhead of managing their own infrastructure.
- **Bitbucket Data Center** (previously known as Stash Data Center) is the deployment option that provides high availability and massive scale for source code management systems. It is especially beneficial for large organizations that have thousands of developers, hundreds of continuous integration servers, and/or require sustained uptime regardless of load.

What is the impact on my plug-ins? Do I need to reinstall them?

Starting September 22nd, 2015, the new brand will be rolled out with the release of Bitbucket Server 4.0 including major API changes from Stash. The renaming of Stash to Bitbucket requires any custom add-ons to be changed, so it is important that you update your own custom add-ons and check for updated marketplace listed add-ons *before* upgrading. The new brand will only affect Bitbucket Server add-ons and will not have an impact on Bitbucket Cloud add-ons.

Immediate action to take:

- Read the [Bitbucket Server 4.0 release notes](#) to understand the add-ons impact
- Assess what you need to do to upgrade
- [Upgrade](#) to Bitbucket Server 4.0

How do I raise support issues?

Go to <https://support.atlassian.com>, select **Dev Tools**, and submit a support request. We'll get you on track.

Are there any feature differences between deployment options?

The high level benefits of Bitbucket Server and Bitbucket Cloud are the same. They both offer fine-grained permissions, pull request workflow, Git repository sharing and management, and a robust set of extension and integration APIs. However, there are some features that are not the same due to the history of the two deployment options. In the future, Bitbucket's deployment options will grow closer together so you have a familiar code management platform with closer feature, function, and UI parity no matter how you access Bitbucket.

There are a few key exceptions:

- **Mercurial support** - We will continue to support and build for Mercurial repository hosting in Bitbucket Cloud. We do not expect to add support for Mercurial in Bitbucket Server at this time.
- **Issue tracking and wiki** - Bitbucket Cloud offers basic wiki support and issue tracking. With the availability of Confluence for documentation, and JIRA Software for integrated software development planning and tracking, it is not our intention to build the same into Bitbucket Server, however we will continue to improve upon the integration with these products.
- **Snippets** - There are no immediate plans to add support for snippets to Bitbucket Server. A [3rd-party addon](#) is available for those wishing to manage code snippets in Bitbucket Server.

Why aren't the Server and Cloud features the same?

Bitbucket Server (previously known as Stash) was first released in May 2012 as an enterprise-grade, high performance, self-managed Git repository hosting and collaboration tool. It was built from the ground-up for self-managed deployment. Bitbucket Cloud (previously known as Bitbucket) was acquired by Atlassian in 2010 and was purpose-built for the multi-tenant public cloud. As a result, Bitbucket Cloud and Bitbucket Server had

different technical architectures with different feature roadmaps. Over the past 2 years, much progress had been made in bringing conceptual and functional parity to the products, and work will continue to improve upon this.

Can I export my Bitbucket repositories between services?

Yes. Because Git repositories are distributed you can simply upload your repositories into either service. Currently there is no way to migrate other data such as comments and pull requests, but this is something we plan to support at some point in the future.

Will Bitbucket Server support Mercurial?

Mercurial support has not been scheduled and is unlikely to become available in the near future for Bitbucket Server but will continue to be supported in Bitbucket Cloud.

Where do I go if I have any more questions?

Go to [Atlassian Answers](#) if you have more questions.

Can I develop Bitbucket Connect add-ons for Bitbucket Server?

At this time we're unlikely to incorporate support for Atlassian Connect integration framework in Bitbucket Server. We will continue to support the mature and complete Plugins 2 framework - it continues to ship with all of Atlassian's Server products. For more information, check out [Atlassian Connect for JIRA Software Server and Confluence Server](#).

Can I develop P2 add-ons for Bitbucket Cloud?

There are no plans to support P2 plugins for Bitbucket Cloud. We've built the Atlassian Connect platform for developing add-ons for our Cloud products.

Why is there no free pricing tier for Bitbucket Server?

In absence of a free plan, Atlassian provides a starter license for Bitbucket Server which is \$10 for 10 users. This is a perpetual license and comes with 12 months of maintenance and support. All proceeds go to the charity, [Room to Read](#), where our Starter Licenses have raised millions of dollars.

Where can I find more information about Mirroring and Large File Support on Bitbucket Server?

All current details are shared in this [blog post](#). Follow our Bitbucket blog for more details in the near future.

Where can I find more information about Projects and Build Status on Bitbucket Cloud?

We don't have much to share right now other than the information we have already shared in the [blog post](#). Follow our Bitbucket blog for more details in the near future.

How do I change the external database password

You can change the password the Bitbucket Server uses to connect to an external database, however you don't do this from the Bitbucket Server Administration area – you must follow the procedure described below.

Related pages:

- [Connecting Bitbucket Server to an external database](#)

To change the password that Bitbucket Server uses when connecting to an external database:

1. Stop Bitbucket Server. See [Starting and stopping Bitbucket Server](#).
2. Get your database administrator to change the password on your database.
3. Go to your [Bitbucket Server home directory](#).

Edit the `bitbucket.properties` file to change the line that looks like:

```
jdbc.password=MY_PASSWORD
```

replacing MY_PASSWORD with your new database password.

4. Restart Bitbucket Server. See [Starting and stopping Bitbucket Server](#).

Bitbucket Server home directory

What is the Bitbucket home directory?

The Bitbucket home directory is created automatically by the Bitbucket Server installer – see [Getting started](#) to install and start using Bitbucket Server.

The information on this page only applies if you are [manually installing](#) or upgrading Bitbucket Server.

The Bitbucket home directory is where your Bitbucket Server data is stored. The home directory location is defined either by the BITBUCKET_HOME environment variable, or in the BITBUCKET_HOME line of:

- Windows: <Bitbucket Server installation directory>\bin\setenv.bat
- Linux and Mac: <Bitbucket Server installation directory>/bin/setenv.sh

 Bitbucket Server 4.0 and later versions don't allow the Bitbucket Server home directory to be the same directory as, or a subdirectory of, the <Bitbucket Server installation directory>. The Bitbucket home directory, as defined by the BITBUCKET_HOME variable, must be in a separate location – Bitbucket Server will fail on startup otherwise. And by the way, you'll need separate Bitbucket Server home directories if you want to run multiple instances of Bitbucket Server (when these are not nodes for a Bitbucket Data Center).

 Where possible, you should choose a location for your Bitbucket home directory that will *never* need to be moved. Some home contents are location-sensitive, so moving the home directory may corrupt them. Bitbucket Server attempts to update contents when it detects that the home directory has moved, but the safest approach is to avoid the issue altogether by leaving the home directory in the same location.

 Where possible, do not configure Bitbucket Server instances to use a UNC path. This can cause issues forking and merging pull requests. Instead, [map a network drive](#) or use the mklink command to create a symbolic link to the networked location and update BITBUCKET_HOME to reference the new drive letter/path.

What does the Bitbucket home directory contain?

Your Bitbucket home directory contains the following directories and files:

Path	Description
caches	Contains cache and index files. It should be safe for these files to be deleted between application restarts; however, these files must not be modified or deleted while Bitbucket Server is running.
shared/config	Contains application configuration.
shared/data	Contains the Git repositories, project avatars, and the embedded HSQL database if an external database is not configured.
export	Contains dump files produced during database migrations.
lib	Can contain third-party jars such as the MySQL JDBC driver.
lib/native	Can contain <i>native libraries</i> , such as Tomcat's APR-based native library.

log	Contains log files for Bitbucket Server.
shared/plugins	Contains plugin related data (such as externally uploaded plugins) for Bitbucket Server.
tmp	Contains temporary files created by the system. Its contents can safely be deleted when Bitbucket Server is <i>not running</i> .
shared/bitbucket.properties	Allows configuring various aspects of how Bitbucket Server behaves, such as its database connection pool size and the location of the Git binary to use. This file will be created automatically during a database migration. It can be created manually otherwise. See Bitbucket Server config properties for further information.

Setting the Bitbucket home directory

Note that the Bitbucket home directory is created automatically by the Bitbucket Server installer – see [Getting started](#).

Only when you are [installing Bitbucket Server from an archive file](#) will you need to set the value of BITBUCKET_HOME yourself, as described in this section.

▼ [Click here if setting BITBUCKET_HOME on Linux or Mac...](#)

The Bitbucket Server [home directory](#) is where your Bitbucket Server data is stored.

Create your Bitbucket home directory (without spaces in the name), and then tell Bitbucket Server where you created it by editing the <Bitbucket Server installation directory>/bin/setenv.sh file – uncomment the BITBUCKET_HOME line and add the absolute path to your home directory. Here's an example of what that could look like when you're done:

```
#
if [ "${STASH_HOME}" = "x" ]; then
    export STASH_HOME="/home/username/stash-home"
fi
```

▼ [Click here if setting BITBUCKET_HOME on Windows...](#)

The Bitbucket Server [home directory](#) is where your Bitbucket Server data is stored.

Create your Bitbucket Server home directory, and then tell Bitbucket Server where you created it by setting a BITBUCKET_HOME environment variable, as follows.

For Windows 7:

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "BITBUCKET_HOME" as the **Variable name**, and the absolute path to your Bitbucket home directory as the **Variable value**. Don't use a trailing backslash.

There are a few things to know about setting up the Bitbucket home directory on Windows that will make life easier:

- You *should not* locate your Bitbucket home directory inside the <Bitbucket Server installation directory> — they should be entirely separate locations. If you do put the home directory in the <Bitbucket Server installation directory> it will be overwritten, and lost, when Bitbucket Server gets upgraded. And, by the way, you can't use the same Bitbucket home directory for multiple instances of Bitbucket Server.
- Keep the path length to the Bitbucket home directory as short as possible. See [Bitbucket Server always shows incorrect Merge Conflict in PRs](#) for an explanation.
- Don't use spaces in the path to the Bitbucket home directory.

Securing the Bitbucket home directory

The internal database files, the migration dump files and `bitbucket.properties` all contain information that may be considered secret (server settings, salted and hashed user passwords, database passwords, etc).

For production use, we strongly recommend that you secure this directory against unauthorized access.

We recommend the following precautions:

- Assign a separate restricted user account on the machine for running Bitbucket Server (not a root/administrator user)
 - If you wish to run Bitbucket Server on port 80, use a separate http front end as described in [Integrating Bitbucket Server with Apache HTTP Server](#) (do not run as root/Administrator if security of the home directory is important to you)
- Ensure that only the user running Bitbucket Server can access the Bitbucket home directory, and that this user has read, write and execute permissions, by setting file system permissions appropriately for your operating system.

About the repositories

As noted above, `data` contains the Git repositories being managed by Bitbucket Server, where "managed by Bitbucket Server" are the operative words. The repositories are for *Bitbucket Server* to interact with, and they are configured and managed accordingly. They are *not* a mechanism for configuring Bitbucket Server behavior. We *strongly* recommend that customers never modify them, nor interact with them directly. They are *intentionally* structured in a way which does not lend itself well to direct interaction.

Being Git repositories, there are certainly standard aspects to how the repositories on disk are stored and how they function. However, the exact way they are configured *can and does* change between Bitbucket Server releases. Bitbucket Server makes *no effort* to preserve unexpected configuration changes which have been applied by customers, and such changes may cause failures at runtime or during upgrades. If there is an aspect of Bitbucket Server's behavior you wish to configure, please open a feature request on jira.atlassian.com rather than trying to modify the repositories directly.

Repositories are *location sensitive*. Moving your Bitbucket home directory will result in the system being locked (briefly) on startup while Bitbucket Server updates the repositories on disk. Assuming the updates are applied successfully, the system will then unlock itself for normal usage.

Where possible, please choose a Bitbucket Server home location which will not need to be changed later.

Raising a request with Atlassian Support

If you encounter any problems when setting up or using Bitbucket Server, please let us know — we're here to help!

You may want to search the following first:

- the [Atlassian Answers site](#) (the Bitbucket Server forum), where Atlassian staff and Bitbucket Server users can answer your questions.
- the [Bitbucket Server Knowledge Base](#).

If you've found a bug in Bitbucket Server, or want to request a feature or improvement, raise a ticket in the Bitbucket Server project of our [public issue tracker](#). Try searching for similar issues - voting for an existing issue is quicker, and avoids duplicates.

If you still need assistance, please raise a support request, either from within Bitbucket Server or on the Atlassian Support site, as described in the following sections.

Providing as much information as possible about your Bitbucket Server installation with your initial request will help our Support Engineers to give you a faster and more complete response.

On this page:

- [Raising a Support Request from within Bitbucket Server](#)
- [Raising a Support request yourself at Atlassian Support](#)
- [Information you should provide](#)

Raising a Support Request from within Bitbucket Server

This method depends on having a [mail server configured for Bitbucket Server](#) that supports large zip file attachments.

1. Log in to Bitbucket Server (as a [System Administrator](#)) and go to the [admin area](#).
2. Click **Atlassian Support Tools** (under 'Support') then **Support Request**.
3. Provide as much information as possible in the Description, including steps to replicate the problem, and any error messages that are appearing on the console or in the [logs](#). For performance issues, please include [profiling logs](#). See the section below about [information you should provide](#).
4. Click **Send**.

This will produce a zip file containing the information categories selected from the list and will email this to Atlassian Support. You will receive an email advising you of details of the Support Request that was automatically created, and you will receive emailed updates about progress on your issue. You can also see the status of your request directly by visiting the [Atlassian Support System](#).

Raising a Support request yourself at Atlassian Support

1. Log in to Bitbucket Server (as a [System Administrator](#)) and go to the [admin area](#).
2. Click **Atlassian Support Tools** (under 'Support') then **Support Zip**.
3. Select information categories to include in the zip file.
4. Click **Create**.

The zip file is created in the [home directory](#) of the Bitbucket Server instance, for example `<Bitbucket home directory>\export\Bitbucket Server_support_2013-11-17-20-49-18.zip`.

When you now go to [Atlassian Support](#) and create a Support Request, you can attach the Support Zip file to the request.

Please provide as much information as possible in the request, including steps to replicate the problem, and any error messages that are appearing on the console or in the [logs](#). For performance issues, please include [profilin g logs](#). See the section below about [information you should provide](#).

Information you should provide

In addition to the logs and configuration information that you can include in the Support Request zip file, the following information can help to give you a faster response:

Environment details

- Bitbucket Server version
- Java version (for example OpenJDK 1.7.0 JRE)
- Git and Perl versions
- Operating system (for example, Windows 7, Mac OS X 10.6.8)
- Database type (for example, MySQL) and version
- Browsers and versions
- Network topology - is Bitbucket Server running behind a reverse proxy? Is that secured using HTTPS (SSL)?

Configuration

- Java settings, including JVM_MINIMUM_MEMORY, JVM_MAXIMUM_MEMORY

Logs

You may need to adjust the logging level, or enable profiling in Bitbucket Server, in order to get more detailed logs. See [Bitbucket Server debug logging](#).

- Debug logs – Bitbucket Server debug logs can be found in `<Bitbucket home directory>/log`.
- Profiling logs – Bitbucket Server profiling logs can help with analyzing performance issues and can be found in `<Bitbucket home directory>/log`.

Performance factors

- Number of concurrent Git clones
- Number of users
- The size of the .git directory
- CPU spec, number of cores, whether hyperthreading is enabled
- RAM and cache sizes

Integrations

- Other Atlassian applications (and their versions)
- Which build servers are integrated with Bitbucket Server, if any?
- Are Application Links configured?

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.

- [Bug fixing policy](#)
- [New features policy](#)
- [Security Bugfix Policy](#)
- [Finding Your Bitbucket Server Support Entitlement Number \(SEN\)](#)

To request support from Atlassian, please raise a support issue in our online support system. To do this, visit support.atlassian.com, log in (creating an account if need be) and create an issue under Bitbucket Server. Our friendly support engineers will get right back to you with an answer.

Bug fixing policy

Summary

- Our Support team will help with workarounds and bug reporting
- We'll generally fix critical bugs in the next maintenance release
- We schedule non-critical bugs according to a variety of considerations

[Report a bug](#)

Are you developing an add-on for an Atlassian product or using one of our APIs? Report any related bugs [here](#).

Bug reports

Atlassian Support is eager and happy to help verify bugs—we take pride in it! Create an issue in our [support system](#), providing as much information as you can about how to replicate the problem you're experiencing. We'll replicate the bug to verify, then lodge the report for you. We'll also try to construct workarounds if possible.

Search existing bug reports

Use our [issue tracker](#) to search for existing bugs, and [watch](#) the ones that are important to you. When you watch an issue, we'll send you an e-mail notification when the issue's updated.

How we approach 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 (the 1 in 6.0.1, for example).

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) we'll fix it in the next maintenance release, provided that:

- The fix is technically feasible (it doesn't require a major architectural change)
- It doesn't impact the quality or integrity of a product

For non-critical bugs, the developer assigned to fixing bugs prioritises the 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

Developers responsible for fixing bugs also monitor comments on existing and new bugs, so you can comment to provide feedback if you need to. We give high priority to [security issues](#).

When considering the priority of a non-critical bug, we try to determine a *value* score for a bug. The score takes into account the severity of the bug from our customers' perspective, how prevalent the bug is, and whether new features on our roadmap may render the bug obsolete. Our developers combine the value score with a *complexity* score (how difficult the bug is) when selecting issues to work on.

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

New features policy

Summary

- We encourage and display customer comments and votes openly in our issue tracking system, <http://jira.atlassian.com>.
- We do not publish roadmaps.
- Product Managers review our most popular voted issues on a regular basis.
- We schedule features based on a variety of factors.
- Our [Atlassian Bug Fixing Policy](#) is distinct from this process.
- Atlassian provides consistent updates on the top 20 issues.

How to track what features are being implemented

When a new feature or improvement is scheduled, the 'fix-for' version will be indicated in the JIRA issue. This happens for the upcoming release only. We maintain roadmaps for more distant releases internally, but because these roadmaps are often pre-empted by changing customer demands, we do not publish them.

How Atlassian chooses what to implement

In every [major release](#) we *aim* to implement highly requested features, but it is not the only determining factor. Other factors include:

- **Customer contact:** We get the chance to meet customers and hear their successes and challenges at Atlassian Summit, Atlassian Unite, developer conferences, and road shows.
- **Customer interviews:** All product managers at Atlassian do customer interviews. Our interviews are not simply to capture a list of features, but to understand our customers' goals and plans.
- **Community forums:** There are large volumes of posts on [answers](#), of votes and comments on [jira.atlassian.com](#), and of conversations on community forums like groups on LinkedIn.
- **Customer Support:** Our support team provides clear insights into the issues that are challenging for customers, and which are generating the most calls to support
- **Atlassian Experts:** Our [Experts](#) provide insights into real-world customer deployments, especially for customers at scale.
- **Evaluator Feedback:** When someone new tries our products, we want to know what they liked and disliked and often reach out to them for more detail.
- **In product feedback:** The [JIRA Issue Collectors](#) that we embed our products for evaluators and our Early Access Program give us a constant pulse on how users are experiencing our product.
- **Usage data:** Are customers using the features we have developed?
- **Product strategy:** Our long-term strategic vision for the product.
- Please read our [post on Atlassian Answers](#) for a more detailed explanation.

How to contribute to feature development

Influencing Atlassian's release cycle

We encourage our customers to vote on issues that have been raised in our public JIRA instance, <http://jira.atlassian.com>. Please find out if your request already exists - if it does, vote for it. If you do not find it you may wish to create a new one.

Extending Atlassian products

Atlassian products have powerful and flexible extension APIs. If you would like to see a particular feature implemented, it may be possible to develop the feature as a plugin. Documentation regarding the [plugin APIs](#) is available. Advice on extending either product may be available on the user mailing-lists, or at [Atlassian Answers](#).

If you require significant customisations, you may wish to get in touch with our [partners](#). They specialise in extending Atlassian products and can do this work for you. If you are interested, please [contact us](#).

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Security Bugfix Policy

See [Security @ Atlassian](#) for more information on our security bugfix policy.

Finding Your Bitbucket Server Support Entitlement Number (SEN)

Your Support Entitlement Number (SEN) is required when raising a support request in our Support system: <http://support.atlassian.com>.

See [Finding Your Support Entitlement Number](#) in the Support space for more general information about how Atlassian Support uses this number.

The three ways of finding your SEN are described below.

On this page:

- [Method 1 — Check the Bitbucket Server Administration Interface](#)
- [Method 2 — Check my.atlassian.com](#)
- [Method 3 — Check your Atlassian Invoice](#)

Method 1 — Check the Bitbucket Server Administration Interface

To find your SEN in the Bitbucket Server administration interface:

1. Click the



icon in the Bitbucket Server header.

2. Click **Licensing** in the left navigation panel (under 'Settings'). The SEN is shown as in the screenshot below:

Server ID [REDACTED]

License

License **Bitbucket (Server): Evaluation**

Licensed to **Atlassian**

Support Entitlement Number (SEN) [REDACTED]

Expires [REDACTED]

User limit **Unlimited**

Clustering **Not available with this license**

Method 2 — Check my.atlassian.com

To find your SEN via my.atlassian.com:

1. Log into my.atlassian.com as the Account Holder or Technical Contact for your Bitbucket Server product.
2. The SEN will be shown, as per the screenshot below:

The screenshot shows the 'Bitbucket (Server): Evaluation' page in the Atlassian interface. It features a table with the following data:

Server ID	License Key	SEN
[Redacted]	[Redacted]	AAABJvTaxVhVd mCZj+e9gEVpFDX

At the bottom of the table, there are links for 'Actions', 'Buy', and 'De'.

Method 3 — Check your Atlassian Invoice

Your Support Entitlement Number (SEN) appears on the third page of your Atlassian Invoice.

Building Bitbucket Server from source

This page has moved!

To our Development Hub at <https://developer.atlassian.com/bitbucket/server/docs/latest/how-tos/building-bitbucket-server-from-source-code.html>.

But you really wanted to [build a plugin anyway](#), right?

Contributing to the Bitbucket Server documentation

Would you like to share your Bitbucket Server hints, tips and techniques with us and with other Bitbucket Server users? We welcome your contributions.

Blogging your technical tips and guides

Have you written a blog post describing a specific configuration of Bitbucket Server or a neat trick that you have discovered? Let us know, and we will link to your blog from our documentation.

Contributing documentation in other languages

Have you written a guide to Bitbucket Server 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.

On this page:

- [Blogging your technical tips and guides](#)
- [Contributing documentation in other languages](#)
- [Updating the documentation itself](#)
 - [Getting permission to update the documentation](#)
 - [Our style guide](#)
 - [How we manage community updates](#)

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.

Collecting analytics for Bitbucket Server

We are continuously working to make Bitbucket Server better. Data about how you use Bitbucket Server helps us do that. We have updated our Privacy Policy so that we may collect usage data automatically unless you disable collection. The data we collect includes information about the systems on which your installation of Bitbucket Server is operating, the features you use in Bitbucket Server, and your use of common IT terminology within the product. For more details, see our [Privacy Policy](#), in particular the 'Analytics Information from Downloadable Products' section.

See also our [End User Agreement](#).

How to change data collection settings?

You can opt in to, or out of, data collection at any time. A Bitbucket Server admin can change the data collection settings by going to **Analytics** (under 'Settings') in the Bitbucket Server admin area.

How is data collected?

We use the Atlassian Analytics plugin to collect event data in Bitbucket Server. Analytics logs are stored locally and then periodically uploaded to a secure location.

Bitbucket Server EAP - How to update your add-on

Redirection Notice

This page will redirect to [DM:How to update your add-on](#).

This is an early preview of an upcoming launch from Atlassian

We are sharing early, privileged information with you as a trusted partner of Atlassian. We ask that you keep this information private and not share it with any other third parties (except any trusted development partners) until the official launch.

This document and referenced resources are here to help reduce the time

and effort needed on your part to make your add-on compatible with Bitbucket 4.0.

You should "watch" this page (press 'w') as we will be making updates to it as we receive feedback from our add-on developers and release subsequent EAPs.

On this page

- [About the changes](#)
- [How to update your add-on](#)
- [Other helpful resources](#)
- [Outline of API changes](#)
- [Outline of changes made to the REST API](#)
- [Javascript Events](#)
- [Add-on update strategies](#)
- [Rename checklist](#)

About the changes

Bitbucket Server 4.0 is the biggest API release Bitbucket Server has ever seen. The decision to break so many add-ons with this release was not taken lightly. As developers, we understand the friction such changes can cause. However, we strongly believe this short term pain will be for long term gain with a clearer, cleaner, more consistent and more robust API in 4.0 and beyond. We are making quite a few large changes in this release in the interest of consistency, and with the strong hope that we will not need to make such a drastic change again anytime soon.

Backend changes

One of the exciting changes with this release is the new **JDK minimum requirement of 1.8**, that allows you to use a wide new array of language features in your add-on code. *While Bitbucket Server 3.x supports running on Java 8; Bitbucket Server supports compiling to Java 8.*

The largest change was we **renamed our package namespace** from `com.atlassian.bitbucket` to `m.atlassian.bitbucket`, but this should also be a simple change when you update your plugins. The Bitbucket Server team updated over 100 add-ons internally and the process was quite straightforward using refactoring support in modern IDEs.

In addition to the repackaging, we've also **removed all @deprecated APIs from the codebase**. Most of these had existing replacement methods in place, but some were removed without replacement. You can consult the [latest Bitbucket Server 3.x documentation](#) for details on what replacement method to use if the new method is not obvious.

Finally, several of our bundled plugins were exporting API (our `com.atlassian.bitbucket:stash-build-integration` plugin, for example), which meant plugin developers added dependencies on that jar. However, only a small portion of the code in that jar was exported. This was a frequent source of plugin issues because plugin developers attempted to use our internal classes. In 4.0, the **exported APIs from all of our plugins have been extracted into separate modules** (like with the `stash-build-integration` example, the build API is now in `com.atlassian.bitbucket.server:bitbucket-build-api`). These new API modules contain *all* of the code that is published for plugin developers to use.

Front-end changes

Our Javascript and Soy API modules have moved to the Bitbucket namespace. AMD Modules, previously found under `stash/api/*`, are now `bitbucket/*`. Non-API modules will be under `bitbucket/internal`. For example, `stash/api/util/navbuilder` is now `bitbucket/util/navbuilder`. For API Soy templates, these are now also under the bitbucket namespace - `BitbucketServer.template.branchSelector` is now `bitbucket.component.branchSelector`.

Another front-end change is that most keys – including Form Fragments, Web Panel & Section locations, and Web Resource add-ons – have been moved to Bitbucket namespaces. There is more detail on these

changes below.

Any methods or modules that were deprecated for removal in 4.0 have been removed.

How to update your add-on

There is no distribution or installer for the Bitbucket Server 4.0 EAP1 build as this is a developer pre-release. It is only available via Maven artifacts. By updating your add-on's pom.xml following the instructions below, Maven will download the Bitbucket Server artifacts.

1. Update your pom.xml file to reference the latest version of the Bitbucket Server 4.0. You will need to update version properties for both Bitbucket Server and AMPS, which currently requires a pre-release version to build Bitbucket Server plugins, as well as dependencies on any API artifacts.

```
<dependency>
  <groupId>com.atlassian.bitbucket.server</groupId>
  <artifactId>bitbucket-api</artifactId>
  <version>${bitbucket.version}</version>
  <scope>provided</scope>
</dependency>
...
<properties>
  <bitbucket.version>4.0.0-eap1</bitbucket.version>

  <bitbucket.data.version>${bitbucket.version}</bitbucket.data.version>
  <amps.version>6.1.0-6cf99b0</amps.version>
  ...
</properties>
```

2. In your pom.xml you will also need to change or add configuration for the `bitbucket-maven-plugin` (depending on whether you are supporting both Bitbucket Server and Bitbucket Server, or just Bitbucket Server):

```

<build>
  <plugins>
    <plugin>
      <groupId>com.atlassian.maven.plugins</groupId>
      <artifactId>bitbucket-maven-plugin</artifactId>
      <version>${amps.version}</version>
      <extensions>true</extensions>
      <configuration>
        <products>
          <product>
            <id>bitbucket</id>
            <instanceId>bitbucket</instanceId>
            <version>${bitbucket.version}</version>

          </product>
        </products>
      </configuration>
    </plugin>
  </plugins>
</build>

```

3. Update your add-on's name and description in pom.xml to reference "Bitbucket" instead of "Bitbucket Server". For example:

```

<name>Bitbucket Server - Realtime Editor</name>
<description>Provides support for real-time collaborative
editing.</description>

```

4. Optional: If your plugin will only support Bitbucket Server, remove any Bitbucket Server dependencies

```

<groupId>com.atlassian.bitbucket</groupId>
<artifactId>stash-api</artifactId>

```

5. For a class with compilation errors, first remove any `com.atlassian.bitbucket` import statements that are red.
6. Use the suggested imports your IDE provides, and/or consult the API Changelog and table below.
7. Open the `atlassian-plugin.xml` inside your IDE
 - a. Rename any `com.atlassian.bitbucket` imported components to `com.atlassian.bitbucket` (or equivalent as mentioned in the API changelog)
 - b. If you are using any web-resources with a dependency on `com.atlassian.bitbucket.stash-web-api`, change them to `com.atlassian.bitbucket.server.bitbucket-web-api`
 - c. Check for any other changes in your resources required due to renamed frontend API
8. If your add-on has JavaScript which uses the Bitbucket Server JavaScript API, change your AMD module imports from `stash/api/*` to `bitbucket/*`
9. Test the add-on starts in Bitbucket Server using:

```

mvn clean bitbucket:debug

```


Other helpful resources

- [Bitbucket Server developer documentation](#). This link is to be quiet until launch of Bitbucket Server 4.0.
- Live help via the [Bitbucket Server EAP](#) public HipChat room. Bitbucket developers and other add-on developers will be available here to help with issues or questions you may have.
- As a last resort, you can create a support ticket in the [SSP project](#), and mention you are using the Bitbucket Server 4.0 EAP.

Please do not create issues in the Bitbucket Server project on jira.atlassian.com or ask questions on [Atlassian Answers](#) in relation to the Bitbucket Server 4.0 EAP, as these are public forums.

Outline of API changes

Area	Bitbucket Server 3.x	Bitb
Java packages	com.atlassian.bitbucket	com
Maven	<code><artifactId>maven-stash-plugin</artifactId></code> e.g. <pre> <plugin> <groupId>com.atlassian.maven.plugins</groupId> <artifactId>maven-stash-plugin</artifactId> <version>\${amps.version}</version> <extensions>true</extensions> <configuration> <products> <product> <id>stash</id> <instanceId>stash</instanceId> <version>\${stash.version}</version> <dataVersion>\${stash.data.version}</dataVersion> </product> </products> </configuration> </plugin> </pre>	<code><ar</code> e.g.
Exceptions	com.atlassian.bitbucket.exception.ServiceException	com The rem they atl
Java User model	com.atlassian.bitbucket.user.Bitbucket ServerUser	com
Java Authentication Context	com.atlassian.bitbucket.user.Bitbucket ServerAuthenticationContext	com

Java Event model	<code>com.atlassian.bitbucket.event.Bitbucket ServerEvent</code>	<code>com</code> The dow exar eve
Java model	<code>Changeset, DetailedChangeset</code>	<code>Com</code> We'h • , • , • - i All c equi Whe been ang
Soy templates	<code>changeset</code>	<code>com</code>
Application constants	<code>com.atlassian.bitbucket.Product</code> • <code>#NAME="Bitbucket Server"</code> • <code>#DATA_CENTER_NAME="Bitbucket Data Center"</code> • <code>#FULL_NAME="Atlassian Bitbucket Server"</code>	<code>com</code> • , • , • ,
Javascript API modules	<code>stash/api/*</code> (eg. <code>stash/api/util/server</code>)	<code>bit</code>
Soy API namespaces	<code>Bitbucket Server.template.branchSelector</code>	<code>bit</code>
Web API plugin module	<code>com.atlassian.bitbucket.stash-web-api</code>	<code>com</code>
Core web plugin module	<code>com.atlassian.bitbucket.stash-web-plugin</code>	This othe
Web Panel & Section Locations	<code>stash.*</code> e.g. <code>stash.branch.list.actions.dropdown</code>	<code>bit</code> e.g.
Web Resource Contexts	<code>stash.*</code> e.g. <code>stash.layout.pullRequest</code>	<code>bit</code> <code>bit</code>
Web Resource Modules	<code><stash-resource/></code>	<code><cl</code> Note whe
Web I18n	<code>stash_i18n</code>	<code>get</code> sta doe: usa

Form Fragments	stash.*	bit
Javascript Events	stash.*	bit See

Outline of changes made to the REST API

The *payloads* for some of the REST resources have changed. All URLs are the same except that the default context path is now `/bitbucket` instead of `/stash`.

Self Links

Some REST model classes, like `RestBitbucket ServerUser`, had their "self" links defined two ways:

```
"link": {
  "url": "/users/admin",
  "rel": "self"
},
"links": {
  "self": [
    {
      "href": "http://localhost:7990/stash/users/admin"
    }
  ]
}
```

The "link" attribute is from 1.0 and was deprecated in 2.11. From 4.0, the "link" attribute has been removed. The "self" entry in the "links" map remains.

Changeset to Commit

Ref output, such as branches and tags, had a "latestChangeset" attribute. The following output is from the `/projects/KEY/repos/slug/branches` resource:

```
{
  "size": 1,
  "limit": 1,
  "isLastPage": false,
  "values": [
    {
      "id": "refs/heads/search/STASHDEV-8813-search",
      "displayId": "search/STASHDEV-8813-search",
      "latestChangeset": "de307ea7b6abfa1aad8de6771d79da0a9a7fd3cb",
      "latestCommit": "de307ea7b6abfa1aad8de6771d79da0a9a7fd3cb",
      "isDefault": false
    }
  ],
  "start": 0,
  "nextPageStart": 1
}
```

A "latestCommit" attribute was added in 3.7 and "latestChangeset" was deprecated. It has been

removed in 4.0. *This also applies to pull request refs.*

AttributeMap to PropertyMap

Some REST objects, most notably `RestPullRequest`, had an "attributes" attribute which allowed plugin developers to attach arbitrary data. However, the model for this was very restrictive, being defined in `AttributeMap` as `Map<String, Set<String>>`. In 3.2, `PropertyMap`, defined as `Map<String, Object>`, was added to replace the attribute support. However, for performance and API consistency reasons, most existing attributes were not converted over. In 4.0, the changeover is now complete.

The following JSON snippets show the old and new layouts for pull request properties.

▼ Bitbucket Server

```
"attributes": {
  "resolvedTaskCount": [
    "0"
  ],
  "openTaskCount": [
    "1"
  ],
  "commentCount": [
    "2"
  ]
}
```

▼ Bitbucket Server

```
"properties": {
  "commentCount": 2,
  "openTaskCount": 1,
  "resolvedTaskCount": 0
}
```

As you can see, the new "properties" map allows its numeric entries to be numeric, resulting in much more readable, useful output.

Javascript Events

Since Bitbucket Server 3.0 we have provided a Javascript API module for creating and consuming events (`stash/api/util/events`, now `bitbucket/util/events`), however we haven't documented what events Bitbucket (Bitbucket Server) emits as part of our Developer Docs. Which events should be used and which should be considered internal implementation details were subject to change.

For this initial EAP, JS events have been renamed from `stash.*` to `bitbucket.internal.*`, however we are actively looking at which events we should consider part of the API. We will document usage and guarantee the stability for events that are part of the API for a major version. These events will be renamed to `bitbucket.*` prior to the release of Bitbucket 4.0 to signify this support. [We would like to hear any feedback](#) on which events you make use of in your add-ons, and why, to aid in our consideration of what to consider for the JS Events API.

Add-on update strategies

There are two primary strategies we are suggesting to update your add-on, and here we explain how to implement each and how to provide support for your add-on going forward.

Hard break

The simplest way forward is to branch your add-on and only release Bitbucket Server 4.0 compatible

versions in the future. **Replace** the old `com.atlassian.bitbucket` dependency with the new `com.atlassian.bitbucket` one, fix the resulting compilation errors, and create a new listing on Marketplace.

Backwards/forwards compatibility

The second option is to **add** the new `com.atlassian.bitbucket` dependencies to the current branch of your plugin, alongside your existing `com.atlassian.bitbucket` dependencies, and have both Bitbucket Server 3.x and Bitbucket Server 4.x support going forward.

This can be achieved by using an "application" attribute on any modules you define in `atlassian-plugin.xml`. This way, only those modules compatible with "bitbucket" will start at runtime if running in Bitbucket Server 4.0, and only the "stash" modules will be started if running in Bitbucket Server.

For example, a component that has the same interface but different implementations for Bitbucket Server and Bitbucket.

```
<!-- Bitbucket component -->
<component key="bitbucketComponent"
  name="myComponent"
  class="com.mycompany.plugin.BitbucketComponent"
  application="bitbucket">
  <interface>com.mycompany.plugin.Component</interface>
</component>

<!-- Bitbucket Server component -->
<component key="stashComponent"
  name="myComponent"
  class="com.mycompany.plugin.Bitbucket ServerComponent"
  application="stash">
  <interface>com.mycompany.plugin.Component</interface>
</component>
```

The benefits of this approach include keeping your add-on backward compatible with Bitbucket Server, and it also allows you to keep a single Marketplace listing for your add-on, and would be marked compatible with both Bitbucket Server and Bitbucket Server.

Rename checklist

1. Beware of changing any Strings which are used as keys for accessing data your add-on may store. e.g. namespaces used with `PluginSettingsFactory.createSettingsForKey` or prefixes used with `ApplicationPropertyService.getPluginProperty`
2. Beware of subtle, unexpected changes that can arise from changing your plugin's key, or its Maven `groupId` or `artifactId`
 - a. The default `atlassian-plugin.xml` generated by AMPS uses `key="${project.groupId}.${project.artifactId}"`, so changing your Maven `groupId` or `artifactId` will change your plugin key
 - b. If you are using SAL's `PluginUpgradeTask` and `<param name="build">7</param>` to upgrade your plugin between versions, changing your plugin key will result in *all upgrades being run again*—the build number is recorded against your plugin key, so your new plugin key is considered an all-new plugin
 - c. If you haven't defined a namespace attribute on your `<ao/>` module, by default it uses your plugin key
3. If you are using `ActiveObjects`, you are *strongly* encouraged to set the namespace attribute to ensure the unique hash in your table names does not change. Otherwise anyone who has installed your plugin will "lose" all of their data when your plugin starts using new tables! For example, here's how we defined the `<ao/>` module in our ref sync plugin:

```
<ao key="ao"
namespace="com.atlassian.bitbucket.stash-repository-ref-sync">
```

Releases

The following pages can be found in the [latest documentation for Bitbucket Server](#):

- the [Bitbucket Server upgrade guide](#)
- the [Bitbucket Server security advisories](#)
- the [End of support announcements for Bitbucket Server](#)
- the full release notes for every Bitbucket Server (and Stash) release.

You can get automated notifications about major and minor Bitbucket Server releases by subscribing to the [Atlassian dev tools blog](#).

The list below is a summary of all the Bitbucket Server (and Stash) releases. The change logs included in the release notes (linked below) have details of the related bug-fix releases.

Bitbucket Server was formerly known as Stash.

Bitbucket Server 4.7

15 June 2016

- Add commit tags from within the UI

Read more in the [Bitbucket Server 4.7 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Bitbucket Server 4.6

10 May 2016

- Code search, powered by Elasticsearch.
- Configure time zones for individual users.
- View the latest commit message and timestamp within source view.

Read more in the [Bitbucket Server 4.6 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Bitbucket Server 4.5

5 April 2016

- Support for Git LFS with Smart Mirroring.
- Specify user/group exceptions for branch permissions.

Read more in the [Bitbucket Server 4.5 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Bitbucket Server 4.4

1 March 2016

- Pull Request header and new reviewer statuses

Read more in the [Bitbucket Server 4.4 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Bitbucket Server 4.3

12 January 2016

- Smart Mirroring
- Git LFS

Read more in the [Bitbucket Server 4.3 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Bitbucket Server 4.2**8 December 2015**

Small improvements and bug fixes:

- Comment forms that hit the nesting limit are no longer misaligned.
- Plain Delegated LDAP configuration no longer mistakenly uses hidden forms.
- Updated Apache Commons Collections to v3.2.2
- HipChat plugin connection no longer fails with SQL Server database
- Allow specifying the multicast IP for Hazelcast in Bitbucket Data Center

Read more in the [Bitbucket Server 4.2 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Bitbucket Server 4.1**4 November 2015**

- Supports CommonMark standardization of Markdown in comments and README files.
- 10+ public issues resolved.

Read more in the [Bitbucket Server 4.1 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Bitbucket Server 4.0**21 September 2015**

- Atlassian Stash is now called Bitbucket Server.
- Better pull request searches.

Read more in the [Bitbucket Server 4.0 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.11**14 July 2015**

- Disable HTTP(S) access to Git repositories
- 50+ public issues solved

Read more in the [Stash 3.11 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.10**9 June 2015**

- HipChat integration v2
- Improved branch permissions management
- 25+ public issues resolved

Read more in the [Stash 3.10 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.9

14 May 2015

- Installer improvements
- Improved handling of unlicensed/unauthorized users
- 15+ public issues resolved

Read more in the [Stash 3.9 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.8

1 April 2015

- Completely provision Stash automatically
- Monitor Stash performance with JMX counters
- Deep linking into source file lines
- Cleaner handling of unlicensed and unauthorized users
- Tomcat's server.xml file has been relocated
- Stash Data Center node IDs stability
- Better support for the `go get` command

Read more in the [Stash 3.8 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.7

24 February 2015

- Customize the commit message when merging
- Like and reply to comments, from email notifications
- Installer improvements

Read more in the [Stash 3.7 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.6

13 January 2015

Small improvements and bug fixes:

- Create a branch starting from a tag
- Support for MariaDB
- Support for SMTPS for email notifications
- Stash is more accessible
- Syntax highlighting for fenced code blocks in markdown
- Support for Internet Explorer 9 is deprecated
- 30+ public issues resolved

Read more in the [Stash 3.6 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.5

25 November 2014

- Syntax highlighting in side-by-side diffs and source view
- Comment likes - amplify review feedback
- Tags are now displayed in the Commits list

- Support for Java 7 deprecated
- Support for Git versions earlier than 1.8 on the server is deprecated

Read more in the [Stash 3.5 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.4

21 October 2014

- Batched email notifications
- Support for Microsoft Office and OpenOffice/LibreOffice MIME types
- Aggregated group membership option for multiple user directories
- Disabling pull request assistive URLs in the console
- JMX support

Read more in the [Stash 3.4 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.3

10 September 2014

- Tasks for pull requests
- Tomcat 8 is now bundled
- Pull request URLs are displayed in the console after pushing

Read more in the [Stash 3.3 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.2

30 July 2014

- Improved workflow for creating pull requests
- Improved comment navigation in diffs - jump to next/previous comment
- Landing page for new users
- Stash analytics disclosure

Read more in the [Stash 3.2 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.1

24 June 2014

- Code search in pull request diffs
- Attachments for pull request comments and descriptions
- Stash installer for Linux, Mac OS X and Windows
- Microsoft SQL Server 2014 is now supported
- Oracle 12c is now supported
- Transcoding is now supported for diff views

Read more in the [Stash 3.1 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 3.0

20 May 2014

- Branch compare
- Sidebar redesign
- Webcam capture of user avatars
- Stash internationalization
- Java 6 support removed. Stash now requires Java 7 at least.
- Java 8 is now supported.
- Internet Explorer 8 support removed.
- Stash JavaScript API published
- Stash APIs deprecated in Stash 2.x releases (before 2.12) have now been removed.

Read more in the [Stash 3.0 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 2.12

25 March 2014

- Custom avatar images
- User name linking to profile page
- Read/write access keys
- Access key bulk revocation
- Diff hunk map
- DIY Backup
- Comment display toggling
- Markdown rendering in the Source view
- Markdown table syntax in comments

Read more in the [Stash 2.12 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 2.11

25 February 2014

- Commit comments
- File comments for pull requests and commits
- Side-by-side diffs
- Lock out recovery process
- MySQL 5.6.16+ support

Read more in the [Stash 2.11 release notes and change log](#).

See the [Bitbucket Server upgrade guide](#).

Stash 2.10

17 December 2013

Small improvements:

- [Branch Utils REST API](#)
- The SSH clone URL is now shown for admins who don't have a user key, when a project or repository access key is available
- The permission screens for projects and repositories have been tidied
- Log files in Stash now use UTF-8 encoding when you start Stash from IDEA or Maven
- The URL now updates correctly when you switch revisions via the file history, when viewing the 'Diff to previous' tab for a repo
- Stash has been updated to use [AUI 5.3](#)
- Atlassian platform upgrade for 2.10

Read the [Stash 2.10 release notes and change log](#).

Stash 2.9

19 November 2013

- Branch listing improvements
- SSH access keys for projects and repositories
- Pull request inbox – my pull requests
- Build status during branch creation
- Support for PostgreSQL 9.3
- Extra keyboard shortcuts
- Extra merge strategy option
- Support for changing usernames

Read the [Stash 2.9 release notes and change log](#).

Stash 2.8

1 October 2013

- Stash branching model
- Branch creation from within Stash
- Branch creation from within JIRA
- Automated merges
- Branch listing page
- Move Git repositories between Stash projects
- Improved integration with Atlassian SourceTree
- Small improvements

Read the [Stash 2.8 release notes and change log](#).

Stash 2.7

20 August 2013

- JIRA issue transitions
- Support for multiple JIRA instances
- Autolink JIRA issues in markdown
- Backup and restore client (beta)
- Small improvements

Read the [Stash 2.7 release notes and change log](#).

Stash 2.6

22 July 2013

- Fork synchronization
- Audit logging
- Repository Quicksearch
- Application navigator
- Public repositories list
- Small improvements

Read the [Stash 2.6 release notes and change log](#).

Stash 2.5

12 June 2013

- Public access to projects and repositories
- Edit a pull request's destination branch
- Get more context in diffs
- OpenJDK is now supported
- Small improvements

Read the [Stash 2.5 release notes and change log](#).

Stash 2.4

6 May 2013

- Forks
- Repository permissions
- Personal repositories
- Small improvements
- Deprecation of Java 6

Read the [Stash 2.4 release notes and change log](#).

Stash 2.3

26 March 2013

- Crowd single sign-on (SSO)
- Branch deletion
- Git submodule recognition
- SCM Cache plugin for Stash

Read the [Stash 2.3 release notes and change log](#).

Stash 2.2

05 March 2013

- Git repository hooks
- API for hook integrations
- Merge checks for pull requests

Read the [Stash 2.2 release notes and change log](#).

Stash 2.1

05 February 2013

- Pull request integration with JIRA
- Build status API
- Project avatars
- Pull request inbox
- Improved pull request title and description generation

Read the [Stash 2.1 release notes](#).

See the [change log](#) for Stash 2.1.x minor releases.

Stash 2.0

04 December 2012

- Branch permissions
- Markdown support
- Mentions
- Enterprise licenses for 1000 and 2000 users
- Deprecation of Internet Explorer 8

Read the [Stash 2.0 release notes](#).

See the [change log](#) for Stash 2.0.x minor releases.

Stash 1.3

09 October 2012

- Pull requests
- Notifications
- Improved keyboard shortcuts
- README – simple project documentation

Read the [Stash 1.3 release notes](#).

See the [change log](#) for Stash 1.3.x minor releases.

Stash 1.2

07 August 2012

- MySQL, PostgreSQL, SQL Server and Oracle support
- Database migration
- File search
- Add-ons ecosystem
- Small improvements

Read the [Stash 1.2 release notes](#).

See the [change log](#) for Stash 1.2.x minor releases.

Stash 1.1

19 June 2012

- SSH support
- Fast browsing
- Simple permissions
- Image diffs

Read the [Stash 1.1 release notes](#).

See the [change log](#) for Stash 1.1.x minor releases.

Stash 1.0 is released!

1st May 2012

Atlassian Stash is a repository management solution that allows everyone in your organization to easily collaborate on all your Git repositories.

In Stash you can:

- Create Git repositories and organize them into projects
- Browse your repositories and your commits
- View the changesets, diffs, blame and history of your files
- Create new users and organize them into groups
- Manage permissions at a global and at a project level
- Integrate with JIRA

Read the [Stash 1.0 release notes](#).

See the [change log](#) for Stash 1.0.x minor releases.