# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting started</td>
<td>3</td>
</tr>
<tr>
<td>Supported platforms</td>
<td>3</td>
</tr>
<tr>
<td>Installing Stash on Windows</td>
<td>4</td>
</tr>
<tr>
<td>Running Stash as a Windows service</td>
<td>7</td>
</tr>
<tr>
<td>Installing Stash on Linux and Mac</td>
<td>10</td>
</tr>
<tr>
<td>Getting started with Git and Stash</td>
<td>12</td>
</tr>
<tr>
<td>Configuring JIRA integration in the Setup Wizard</td>
<td>18</td>
</tr>
<tr>
<td>Using Stash</td>
<td>24</td>
</tr>
<tr>
<td>Creating projects and managing permissions for a project</td>
<td>25</td>
</tr>
<tr>
<td>Creating repositories</td>
<td>28</td>
</tr>
<tr>
<td>Basic Git commands</td>
<td>29</td>
</tr>
<tr>
<td>Permanently authenticating with Git repositories</td>
<td>31</td>
</tr>
<tr>
<td>Administering Stash</td>
<td>33</td>
</tr>
<tr>
<td>Users and groups</td>
<td>33</td>
</tr>
<tr>
<td>Global permissions</td>
<td>38</td>
</tr>
<tr>
<td>JIRA integration</td>
<td>39</td>
</tr>
<tr>
<td>JIRA compatibility</td>
<td>40</td>
</tr>
<tr>
<td>External user directories</td>
<td>43</td>
</tr>
<tr>
<td>Configuring an LDAP directory</td>
<td>45</td>
</tr>
<tr>
<td>Configuring delegated LDAP authentication</td>
<td>52</td>
</tr>
<tr>
<td>Connecting to Crowd</td>
<td>58</td>
</tr>
<tr>
<td>Connecting to JIRA for user management</td>
<td>59</td>
</tr>
<tr>
<td>Setting up your mail server</td>
<td>62</td>
</tr>
<tr>
<td>Advanced actions</td>
<td>63</td>
</tr>
<tr>
<td>Data recovery and backups</td>
<td>64</td>
</tr>
<tr>
<td>Enabling Stash debug logging</td>
<td>64</td>
</tr>
<tr>
<td>Integrating Stash with Apache HTTP Server</td>
<td>66</td>
</tr>
<tr>
<td>Connecting Stash to an external database</td>
<td>75</td>
</tr>
<tr>
<td>Scaling Stash</td>
<td>76</td>
</tr>
<tr>
<td>System settings</td>
<td>80</td>
</tr>
<tr>
<td>Securing Stash with HTTPS</td>
<td>80</td>
</tr>
<tr>
<td>Releases</td>
<td>87</td>
</tr>
<tr>
<td>Stash 1.0 Release Notes</td>
<td>88</td>
</tr>
<tr>
<td>Git resources</td>
<td>91</td>
</tr>
<tr>
<td>Stash FAQ</td>
<td>92</td>
</tr>
<tr>
<td>Installing the WAR file into your own web server</td>
<td>94</td>
</tr>
<tr>
<td>Support policies</td>
<td>94</td>
</tr>
<tr>
<td>Bug fixing policy</td>
<td>94</td>
</tr>
<tr>
<td>How to report a security issue</td>
<td>95</td>
</tr>
<tr>
<td>New features policy</td>
<td>96</td>
</tr>
<tr>
<td>Patch policy</td>
<td>97</td>
</tr>
<tr>
<td>Security advisory publishing policy</td>
<td>97</td>
</tr>
<tr>
<td>Security patch policy</td>
<td>98</td>
</tr>
<tr>
<td>Severity levels for security issues</td>
<td>99</td>
</tr>
</tbody>
</table>
Getting started
This section describes how to install, set up and get started with Atlassian Stash.

In this section:
- Supported platforms
- Installing Stash on Windows
  - Running Stash as a Windows service
- Installing Stash on Linux and Mac
- Getting started with Git and Stash
- Configuring JIRA integration in the Setup Wizard

Related pages:
- Administering Stash
- Stash FAQ

Supported platforms
This page lists the supported platforms for Stash 1.0.x.

Key: ✔️ = Supported; ❌ = Not Supported

<table>
<thead>
<tr>
<th>Java Version</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle JDK (1)</td>
<td>✔️ 1.7</td>
</tr>
<tr>
<td></td>
<td>✔️ 1.6</td>
</tr>
<tr>
<td></td>
<td>❌ 1.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operating Systems</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows (2)</td>
<td>✔️</td>
</tr>
<tr>
<td>Linux (2)</td>
<td>✔️</td>
</tr>
<tr>
<td>Apple Mac OS X (2)</td>
<td>✔️ (Supported for evaluation use only)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application Servers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Tomcat (3)</td>
<td>✔️ 6.0.x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Databases</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL (4)</td>
<td>✔️ 8.2, 8.3, 8.4</td>
</tr>
<tr>
<td>HSQLDB</td>
<td>✔️ (bundled)</td>
</tr>
<tr>
<td>Web Browsers</td>
<td></td>
</tr>
</tbody>
</table>
Microsoft Internet Explorer

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.0</td>
<td>✔️</td>
</tr>
<tr>
<td>8.0</td>
<td>✔️</td>
</tr>
<tr>
<td>7.0</td>
<td>❌</td>
</tr>
</tbody>
</table>

Mozilla Firefox

<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ Latest stable version supported</td>
</tr>
<tr>
<td>✔️ Tested with 3.6</td>
</tr>
</tbody>
</table>

Safari

<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ Latest stable version supported</td>
</tr>
</tbody>
</table>

Chrome

<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ Latest stable version supported</td>
</tr>
</tbody>
</table>

DVCS Clients

<table>
<thead>
<tr>
<th>Client</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Git - server side</td>
<td>1.7.6</td>
</tr>
<tr>
<td>Git - client side</td>
<td>1.6.6</td>
</tr>
</tbody>
</table>

Notes:

1. Oracle JDK:
   - For the server, please ensure that you have the Java JDK. It is not enough to have just the JRE. JDKs other than the Oracle JDK are not supported.
   - You can download the Java SE Development Kit (JDK) from the Oracle website.
   - Once the JDK is installed, you will need to set the environment variable, pointing to the root JAVA_HOME directory of the JDK. Some JDK installers set this automatically (check by typing 'echo %JAVA_HOME%' in a DOS prompt, or 'echo $JAVA_HOME' in a shell). You should do this before installing Stash.

2. Please note:
   - Stash is a pure Java application and should run on any platform, provided all the JDK requirements are satisfied.
   - If you are using Linux/UNIX: A dedicated user should be created to run Stash, as Stash runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run Stash in Linux:
     ```bash
     $ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/Stash --shell /bin/bash Stash
     ```

3. Deploying multiple Atlassian applications in a single Tomcat container is not supported. We do not test this configuration and upgrading any of the applications (even for point releases) is likely to break it.

We also do not support deploying multiple Atlassian applications to a single Tomcat container for a number of practical reasons. Firstly, you must shut down Tomcat to upgrade any application and secondly, if one application crashes, the other applications running in that Tomcat container will be inaccessible.

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

4. Stash does not yet support data migrations between databases. If you would like to connect Stash to an external database, you must do so before starting Stash for the first time. Please see our documentation on connecting Stash to an external database.

Installing Stash on Windows

This page describes how to install the Stash distribution on Windows.

Before you start
Supported platforms — Please read the Supported platforms page before you install Stash; it lists the applications servers, databases, operating systems, web browsers and JDKs that we have tested Stash with and recommend.

Hardware requirements — Atlassian only officially supports Stash running on x86 hardware and 64-bit derivatives of x86 hardware.

⚠ Git requirements — Stash requires Git 1.7.6 or later.

On this page:

Before you start
  1. Get Java
  2. Get Git
  3. Get Stash
  4. Start Stash
  5. Set up your mail server
  6. Secure Stash (optional)

Related pages:

- Installing Stash on Linux and Mac
- Getting started with Git and Stash
- Supported platforms
- Securing Stash with HTTPS

1. Get Java

1. Check your installed version of Java, and compare with the version on the Supported platforms page.

   ```
   java -version
   ```

2. If necessary, download and install the Java Platform JDK (not the JRE) from Oracle’s website.
3. Check the JAVA_HOME environment variable. The variable value should be the root directory of the Java JDK.

   *Note: this location must not contain spaces. Therefore, the standard installation directory in C:\Program Files (x86) must be changed.*

   ```
   echo %JAVA_HOME%
   ```

4. Set JAVA_HOME, if necessary:

   Windows 7:
   a. Go to Start > search for "sys env" and choose "Edit the system environment variables"
   b. Click Environment Variables....
   c. Click New for either the user account, or if available, system-wide.
   d. Enter JAVA_HOME as the Variable name, and the directory where you installed Java as the Variable value.
2. Get Git

Download and install Git using the Full installer for official Git for Windows.

Please note the following:

- Stash requires Git 1.7.6 or later.
- When installing Git on Windows, ensure that git.exe is available in the path.
  - The Installer will give three options for how to modify your path
  - You must select the option which puts git.exe and the other UNIX tools on the path, as shown below
  - Selecting the option to only add git to the path will not work with Stash

3. Get Stash

1. Download Stash from the Atlassian download site.
2. Extract the downloaded file. The extracted directory is referred to as `<Stash installation directory>` in these instructions.
3. Set the STASH_HOME variable. This points to the home directory — where you want Stash to store your data.
   Do one of the following:
   - Edit `<Stash installation directory>\bin\setenv.bat` and set up STASH_HOME by uncommenting the STASH_HOME line and adding the absolute path to your home directory.
   - Set the environment variable STASH_HOME to the absolute path to your home directory, in a similar way to how you set JAVA_HOME above. Stash checks for this environment variable at launch.

Please note that you cannot use the same Stash home directory for multiple instances of Stash. Also, we don’t recommend locating your Stash home directory inside the `<Stash installation directory>` — they should be entirely separate directories.

4. Start Stash

1. In a terminal, run:
2. In your browser, go to:
   
   [http://localhost:7990](http://localhost:7990)

3. The Setup wizard allows you to:
   
   a. Enter your Stash license.
   
   b. Create your first admin user.
   
   c. Set up an application link for JIRA integration. You can set up JIRA integration later if you wish; see Configuring JIRA integration in the Setup Wizard.

5. Set up your mail server

In order for your users to receive a link allowing them to generate their password you will need to have your mail server configured. See the instructions on [Setting up your mail server](#).

6. Secure Stash (optional)

If your instance of Stash is exposed to the internet, or if your data is sensitive, you should consider configuring access using HTTPS (HTTP over SSL). See the instructions on [Securing Stash with HTTPS](#).

### Running Stash as a Windows service

For long-term use on a Windows server, Stash should be configured to run as a Windows service. This has the following advantages:

- Stash will be automatically restarted when the operating system restarts.
- Stash is less likely to be accidentally shut down, as can happen if the console window Stash was manually started in is closed.
- Stash logs are properly managed by the Windows service.

**Before you start:**

- If you are running a 64-bit version of Windows, note that Apache Tomcat cannot run as a Windows service with a 64-bit JDK. Please ensure that you are using a 32-bit JDK.
- 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’ and choosing Run as administrator).
- Ensure the JAVA_HOME variable is set to the root of your Java platform’s installation directory.

*Note: Your JAVA_HOME cannot contain spaces, so the default installation directory in C:\Program Files\Java won’t work.*

**On this page:**

- [Setting up Stash as a Windows service](#)
- [Troubleshooting](#)

**Related pages:**

- [Installing Stash on Windows](#)
Setting up Stash as a Windows service

To set up Stash to run as a Windows service:

1. Stop Stash.
2. Open a Command Prompt.
3. Change directory ("cd") to the Stash installation directory and then into the bin subdirectory. If a directory in the path has spaces (e.g. C:\Program Files\..), use its eight-character equivalent (e.g. C:\Prog ra-1\..).
4. Run the following commands:

   ```
   > service.bat install
   > ./tomcat6 //US//STASH --Startup auto
   ```

   This will create a service with the name "STASH" and a display name of "Atlassian Stash". If you would like to customize the name you can instead run:

   ```
   > service.bat install MyName
   > ./tomcat6 //US//MyName --Startup auto
   ```

   This will create the service as "MyName" with a display name of "Atlassian Stash MyName".

5. Run the following command to increase the amount of memory that Stash can use (the default is 256 Mb):

   ```
   > tomcat6 //US//service_name --JvmMx 1024
   ```

6. Verify that the Stash service comes back up after restarting the machine.

Here is an example:
C:\Program Files 
(x86)\atlassian-stash-1.0.0\bin>service.bat install
Installing the service 'STASH' ...
Using CATALINA_HOME:    "C:\Program Files 
(x86)\atlassian-stash-1.0.0"
Using CATALINA_BASE:    "C:\Program Files 
(x86)\atlassian-stash-1.0.0"
Using JAVA_HOME:    "C:\Java\jre6"
Using JVM:    "auto"
The service 'STASH' has been installed.

C:\Program Files 
(x86)\atlassian-stash-1.0.0\bin>tomcat6.exe //US//STASH --Startup auto

C:\Program Files 
(x86)\atlassian-stash-1.0.0\bin>tomcat6.exe //US//STASH --JvmMx 1024

C:\Program Files 
(x86)\atlassian-stash-1.0.0\bin>net start STASH
The Atlassian Stash service is starting.
The Atlassian Stash service was started successfully.

Troubleshooting

- Problems may occur when trying to setup Stash to run as a Windows service with JDK 1.6. The problem is due to failure to locate MSVCR71.DLL, which can be found in %JAVA_HOME%/bin. There are two options to resolve this problem:
• Add %JAVA_HOME/bin to PATH, then restart the Stash server.
• Copy MSVCR71.DLL to system path, C:\WINDOWS\SYSTEM32 or C:\WINNT\SYSTEM32.
• Take note of the username that the service is running as, and be sure to modify the /temp and /work directories in your install directory so that this user has read and write permissions.
• You cannot run Stash as a service on a 64-bit operating system if you require allocating more than 1.5GB of memory, due to 32-bit JDK memory limitations and 64-bit JDK/Tomcat service issues.

Installing Stash on Linux and Mac
This page describes how to install the Stash distribution on Linux and Mac.

Before you start

Supported platforms — Please read the Supported platforms page before you install Stash; it lists the applications servers, databases, operating systems, web browsers and JDKs that we have tested Stash with and recommend.

Hardware requirements — Atlassian only officially supports Stash running on x86 hardware and 64-bit derivatives of x86 hardware.

⚠️ Git requirements — Stash requires Git 1.7.6 or later.

On this page:

1. Get Java
2. Get Git
3. Get Stash
4. Start Stash
5. Set up your mail server
6. Secure Stash (optional)

Related pages:

• Installing Stash on Windows
• Getting started with Git and Stash
• Supported platforms
• Securing Stash with HTTPS

1. Get Java

1. Check your installed version of Java, and compare with the version on the Supported platforms page.

```bash
java -version
```

2. If necessary, download and install the Java Platform JDK (not the JRE) from Oracle's website.

3. Check the JAVA_HOME environment variable. The variable value should be the root directory of the Java JDK.
4. Set `JAVA_HOME`, if necessary:

   If you don’t know what your JAVA home directory is try running the command `which java` in your terminal.

   ```
   echo $JAVA_HOME
   ```

   **Linux**

   Do either of the following:

   If `JAVA_HOME` is not set, log in with 'root' level permissions:

   ```
   echo JAVA_HOME="path/to/JAVA_HOME"
   >> /etc/environment
   ```

   If, `JAVA_HOME` needs to be changed, open the `/etc/environment` file in a text editor and modify the value for `JAVA_HOME`:

   ```
   JAVA_HOME="path/to/JAVA_HOME"
   ```

   **Mac**

   Insert the following in your `~/.profile` file:

   ```
   JAVA_HOME="path/to/JAVA_HOME"
   export JAVA_HOME
   ```

   where, for example, the path may be like: `/System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/`

   In a terminal, confirm that `JAVA_HOME` is set:

   ```
   $JAVA_HOME/bin/java -version
   ```

2. Get Git

   Download and install the latest stable Git release from the [Git website](https://git-scm.com/downloads).

   Please note the following:

   - Stash requires Git 1.7.6 or later.
   - See the [Git resources](https://git-scm.com/book) page for links to more Git download sites.
   - At the time of writing, the default Git version on Ubuntu Linux is 1.7.5.x, which is too old for Stash: see [https://launchpad.net/~git-core/+archive/ppa](https://launchpad.net/~git-core/+archive/ppa).

3. Get Stash

   1. Download Stash from the Atlassian download site.
   2. Extract the downloaded file. The extracted directory is referred to as `<Stash installation directory>` in these instructions.
   3. Set the `STASH_HOME` variable. This points to the home directory — where you want Stash to store your data.

   Do one of the following:

   - Edit `<Stash installation directory>)/bin/setenv.sh` and set up `STASH_HOME` by uncommenting the `STASH_HOME` line and adding the absolute path to your home directory.
   - Set the environment variable `STASH_HOME` to the absolute path to your home directory, in a similar way to how you set `JAVA_HOME` above. Stash checks for this environment variable at launch.
Please note that you cannot use the same Stash home directory for multiple instances of Stash. Also, we don't recommend locating your Stash home directory inside the <Stash installation directory> — they should be entirely separate directories.

4. Start Stash
   1. In a terminal, run:

   ```bash
   <Stash installation directory>/bin/start-stash.sh
   ```

   2. In your browser, go to:

   ```
   http://localhost:7990
   ```

   3. The Setup wizard allows you to:
      a. Enter your Stash license.
      b. Create your first admin user.
      c. Set up an application link for JIRA integration. You can set up JIRA integration later if you wish; see Configuring JIRA integration in the Setup Wizard.

5. Set up your mail server

   In order for your users to receive a link allowing them to generate their password you will need to have your mail server configured. See the instructions on Setting up your mail server.

6. Secure Stash (optional)

   If your instance of Stash is exposed to the internet, or if your data is sensitive, you should consider configuring access using HTTPS (HTTP over SSL). See the instructions on Securing Stash with HTTPS.

Getting started with Git and Stash

Atlassian Stash is the repository management solution that allows everyone in your organisation to easily collaborate on all your Git repositories.

This page will guide you through the basics of Stash. 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 to interact with them.
- Manage users and groups, and assign global permissions.

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.
Install Stash

Get Stash running on your computer. See the details here:

- Installing Stash on Linux and Mac
- Installing Stash on Windows

Add users to Stash and grant permissions

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

Go to the Users listing in the Administration area.

Click Create User to go directly to the user creation form.
Once you’ve created a user, click **Change permissions** to set up their access permissions.

There are 4 levels of **user permissions**:

- **System Admin** — can access all the configuration settings of the Stash instance.
- **Admin** — same as System Admins, but they can’t modify file paths or the Stash server settings.
- **Project Creator** — can create, modify and delete projects.
- **Users** — active users who can access Stash.
Create your first project and share it with collaborators

Creating your project

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

Simply click **Create Project** in the Projects page.

<table>
<thead>
<tr>
<th>Project</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angry Nerds</td>
<td>ANERDS</td>
<td>No description</td>
</tr>
<tr>
<td>Bitbucket</td>
<td>BB</td>
<td>Bitbucket is all up in Caviar, eating you</td>
</tr>
<tr>
<td>Caviar UX Lab</td>
<td>CUXLAB</td>
<td>Caviar Lab for UX / UI - Don't remove</td>
</tr>
<tr>
<td>Clover History</td>
<td>CLOV_HIST</td>
<td>Clover Coverage History</td>
</tr>
<tr>
<td>Confluence</td>
<td>CONF</td>
<td>No description</td>
</tr>
</tbody>
</table>

Complete the form and submit it to create your new project.
Opening up project access to others

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

Click the Permissions tab for the project.

On that page you can add users and groups to a project you've already created.

There are 3 levels of project access:

- **Project Administrator** — can create, edit and delete repositories and projects, and configure permissions for projects.
- **Project Writer** — can push to and pull from all the repositories in the project.
- **Project Reader** — can only browse code and comments in, and pull from, the repositories in the project.

Create a repository and get your code into Stash

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.

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.

**A simple clone and push**

Here you will simply 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 of the screen.

In a terminal, run the following command (replace `<stashURL>` with the URL for your instance of Stash):

```bash
git clone <stashURL>/git/<projectname>/<reponame>.git
```

Use your Stash 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 Stash.

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:
If everything went fine, when you refresh the Stash 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.

Check out our Basic Git commands for more information regarding the basic Git commands that you will probably use often.

Configuring JIRA integration in the Setup Wizard

This page describes the ‘JIRA integration’ screen of the Stash setup wizard.

You can connect your application to a JIRA server, to manage your users via JIRA and share information with JIRA. When you are installing the application, the setup wizard gives you the opportunity to configure the JIRA connection automatically. This is a quick way of setting up your JIRA integration with the most common options.

You can also configure the JIRA connections via the application administration screens. In that case, you will need to set up connections individually. There are two parts to the integration process:

- A peer-to-peer link between JIRA and the application for sharing information and facilitating integration features. This link is set up via Application Links.
- A client-server link between the application and JIRA for delegating user and group management to your JIRA server.

Requirements: You need JIRA 4.3 or later.

On this page:
- Connecting to JIRA in the Setup Wizard
- Troubleshooting
- Notes

Related pages:
- Getting started
- JIRA integration
- Connecting to JIRA for user management

Connecting to JIRA in the Setup Wizard

To configure JIRA integration while running the Stash setup wizard:

1. Configure the following setting in JIRA: Allow remote API access.
2. Click Integrate with JIRA and enter the following information when you get to the ‘Connect to JIRA’ step of the setup wizard:

```
cd <reponame>
git add .
git commit -a -m 'My first commit'
git push origin master
```
| **JIRA base URL** | The web address of your JIRA server. Examples: http://www.example.com:8080/jira/ http://jira.example.com |
| **JIRA admin username** | The credentials for a user with the ‘JIRA System Administrators’ global permission in JIRA. |
| **JIRA password** | |
| **Stash base URL** | JIRA will use this URL to access your Stash server. The URL you give here will override the base URL specified in your Stash administration console, for the purposes of the JIRA connection. |

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

**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.

<table>
<thead>
<tr>
<th><strong>Symptom</strong></th>
<th><strong>Cause</strong></th>
<th><strong>Solution</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>The setup wizard displays one of the following error messages:</td>
<td>The setup wizard failed to complete registration of the peer-to-peer application link with JIRA. JIRA integration is only partially configured.</td>
<td>Remove the partial configuration if it exists, try the ‘Connect to JIRA’ step again, and then continue with the setup. Detailed instructions are below.</td>
</tr>
<tr>
<td>• Failed to create application link from JIRA server at &lt;URL&gt; to this &lt;application&gt; server at &lt;URL&gt;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Failed to create application link from this &lt;application&gt; server at &lt;URL&gt; to JIRA server at &lt;URL&gt;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Failed to authenticate application link from JIRA server at &lt;URL&gt; to this &lt;application&gt; server at &lt;URL&gt;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Failed to authenticate application link from &lt;application&gt; server at &lt;URL&gt; to this JIRA server at &lt;URL&gt;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The setup wizard displays one of the following error messages:</td>
<td>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.</td>
<td>Remove the partial configuration if it exists, try the ‘Connect to JIRA’ step again, and then continue with the setup. Detailed instructions are below.</td>
</tr>
<tr>
<td>• Failed to register &lt;application&gt; configuration in JIRA for shared user management. Received invalid response from JIRA: &lt;response&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Failed to register &lt;application&gt; configuration in JIRA for shared user management. Received: &lt;response&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The setup wizard displays the following error message:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Error setting Crowd authentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The setup wizard displays the following error message:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Error reloading Crowd authentication</td>
</tr>
<tr>
<td>The setup wizard has completed the integration of your application with JIRA, but is unable to start synchronizing the JIRA users with your application.</td>
</tr>
<tr>
<td>Restart your application. You should then be able to continue with the setup wizard. If this solution does not work, please contact Atlassian Support.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The setup wizard displays the following error message:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• An error occurred: java.lang.IllegalStateException: Could not create the application in JIRA/Crowd (code: 500). Please refer to the logs for details.</td>
</tr>
<tr>
<td>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 &lt;application&gt; URL.</td>
</tr>
<tr>
<td>Remove the partial configuration if it exists, try the ‘Connect to JIRA’ step again, and then continue with the setup. Detailed instructions are below.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No users can log in after you have set up the application with JIRA integration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possible causes:</td>
</tr>
<tr>
<td>• There are no users in the group that you specified on the ‘Connect to JIRA’ screen.</td>
</tr>
<tr>
<td>• For FishEye: There are no groups specified in the ‘groups to synchronize’ section of your administration console.</td>
</tr>
<tr>
<td>• For Stash: You may not have granted any JIRA groups or users permissions to log in to Stash.</td>
</tr>
<tr>
<td>Go to JIRA and add some usernames to the group.</td>
</tr>
<tr>
<td>• For FishEye: Go to the FishEye administration screens and specify at least one group to synchronize. The default is ‘jira-users’.</td>
</tr>
<tr>
<td>• For Stash: Grant the Stash User permission to the relevant JIRA groups on the Stash Global permissions page.</td>
</tr>
<tr>
<td>If this solution does not work, please contact Atlassian Support.</td>
</tr>
</tbody>
</table>

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:

1. Log in to JIRA as a user with the ‘JIRA System Administrators’ global permission.
2. Click the ‘Administration’ link on the JIRA top navigation bar.
3. 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 the 'Delete' link next to the application link that you want to delete.
d. A confirmation screen will appear. Click the 'Confirm' button to delete the application link.

4. 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.

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 the ‘Delete’ link next to the application link that you want to delete.
   d. A confirmation screen will appear. Click the ‘Confirm’ button 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 the ‘Next’ button.
   d. Enter the following information:
      - ‘Create a link back to this server’ – Tick this check box 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 the ‘Next’ button.
   f. Enter the information required to configure authentication for your application link:
      - ‘The servers have the same set of users’ – Tick this check box, because the users are the same in both applications.
      - ‘These servers fully trust each other’ – Tick this check 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 the ‘Create’ button to create the application link.
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.
   e. 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’ on the FishEye/Crucible ‘Admin Menu’.
  b. Click ‘Edit’ in the section titled ‘JIRA or Crowd Authentication’.
  c. Enter the following information:
     - **Application name** and **Application password** – Enter the values that you defined for your application in the settings on JIRA.
     - **JIRA/Crowd URL** – The web address of your JIRA server. Examples:
       
       ```
       http://www.example.com:8080/jira/
       http://jira.example.com
       ```
     - **Auto-add** – Select ‘Create a FishEye/Crucible user on successful login’ (default) to ensure that your JIRA users will be automatically enrolled into FishEye/Crucible when they first log in via JIRA.
     - **Synchronize users with JIRA/Crowd** – Select ‘Yes’ (default) to ensure that JIRA will synchronize all changes in the user information on a regular basis. The **synchronization interval** is set to 60 minutes (1 hour) by default.
     - **Single sign on (SSO)** – Select ‘Disabled’. This option is not available when using JIRA for user management and if enabled will make the integration fail.
     - **Groups of users to synchronize** – Select at least one group to synchronize. The default is ‘jira-users’.
  d. Click ‘Apply changes’.

- For Stash:
  a. Go to the Stash 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 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 to JIRA for user management](#).

### Notes

When you connect to JIRA in the setup wizard, the setup procedure will configure OAuth authentication between Stash and JIRA. See [Configuring OAuth Authentication for an Application Link](#) for more information.

### Using Stash

This section describes the basics of using Stash.
Creating projects and managing permissions for a project

Projects allow you to group repositories and to manage permissions for them in an aggregated way.

1. Create a project

Click on **Create Project**:
Fill out the form and click **Create Project** when you're done.
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.

2. Manage a project’s permissions

Simply click on the Permissions tab of the project.

You have 3 types of permissions that you can assign to a user or group on a project: Project Reader, Project Writer or Project Admin.

<table>
<thead>
<tr>
<th></th>
<th>Browse</th>
<th>Clone / Pull</th>
<th>Push</th>
<th>Create Repositories</th>
<th>Edit settings / permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Reader</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Project Writer</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Project Admin</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Click on the Add users button in the corresponding section to grant users or groups a specific permission.

Simply hover over a particular user and click on the Remove button to remove a specific permission from a user or group.
Creating repositories

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

⚠️ In order to create repositories you need to be a Project Admin of the project you want to add a repository to.

**Related pages:**

- Getting started with Git and Stash
- Global permissions
- Creating projects and managing permissions for a 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.

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

Let other people collaborate with you

In order to grant users access to this repository you have to set up the permission at the parent project level. More information is available on Creating projects and managing permissions for a project.

Basic Git commands

Here is a list of some basic Git commands to get you going with Git.

See Git resources for Git cheat sheets and other resources.

<table>
<thead>
<tr>
<th>Git task</th>
<th>Notes</th>
<th>Git commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new local repository</td>
<td></td>
<td>git init</td>
</tr>
<tr>
<td>Check out a repository</td>
<td>Create a working copy of a local repository:</td>
<td>git clone /path/to/repository</td>
</tr>
<tr>
<td></td>
<td>For a remote server, use:</td>
<td>git clone username@host:/path/to/repository</td>
</tr>
<tr>
<td>Add files</td>
<td>Add one or more files to staging (index):</td>
<td>git add &lt;filename&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>git add *</td>
</tr>
<tr>
<td><strong>Commit</strong></td>
<td>Commiting changes to head (but not yet to the remote repository): git commit -m &quot;Commit message&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>Push</strong></td>
<td>Send changes to the master branch of your remote repository: git push origin master</td>
<td></td>
</tr>
<tr>
<td><strong>Connect to a remote repository</strong></td>
<td>If you haven't connected your local repository to a remote server, add the server to be able to push to it git remote add origin &lt;server&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>Branch</strong></td>
<td>Create a new branch and switch to it: git checkout -b &lt;branchname&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Switch from one branch to another: git checkout &lt;branchname&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete the feature branch: git branch -d &lt;branchname&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Push the branch to your remote repository, so others can use it: git push origin &lt;branchname&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete a branch on your remote repository git push origin :&lt;branchname&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>Update from the remote repository</strong></td>
<td>Fetch and merge changes on the remote server to your working directory: git pull</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To merge a different branch into your active branch: git merge &lt;branchname&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>View all the merge conflicts: git diff</td>
<td></td>
</tr>
<tr>
<td></td>
<td>View the conflicts against the base file: git diff --base &lt;filename&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Preview changes, before merging: git diff &lt;sourcebranch&gt; &lt;targetbranch&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After you have manually resolved any conflicts, you mark the changed file: git add &lt;filename&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>Tag</strong></td>
<td>You can use tagging to mark a significant changeset, such as a release: git tag 1.0.0 &lt;commitID&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Commitid is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using: git log</td>
<td></td>
</tr>
</tbody>
</table>
**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.

```bash
git checkout -- <filename>
```

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:

```bash
git fetch origin
git reset --hard origin/master
```

**Search**

Search the working directory for `foo()`

```bash
git grep "foo()"
```

---

### Permanently authenticating with Git repositories

Currently, Atlassian Stash only supports HTTP or HTTPS for pushing and pulling from managed Git repositories. Git has no method of caching the user's credentials, 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 Stash.

**On this page:**

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

**Related pages:**

- [Getting started with Git and Stash](#)
- [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.**

⚠️ `git-credential-winstore` currently **fails** for URLs that include a `:PORT` component such as `http://example.com:80/` and throws two intrusive exceptions on every remote git command (pull, push, etc). If you work with Git servers on ports other than :80 or :443, consider using `_netrc` instead of `git-credential-winstore`. 

---

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

Follow these steps if you want to use Git with credential caching on Linux or OSX:

1. Download the binary `git-credential-osxkeychain`.
2. Run the command below to ensure the binary is executable:

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

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

   ```bash
   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 Stash server 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:
Linux or OSX

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 Stash 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.

Administering Stash

This section describes the administrative actions that can be performed from the Stash Administration user interface.

In this section:

- Users and groups
- Global permissions
- JIRA integration
  - JIRA compatibility
- External user directories
  - Configuring an LDAP directory
  - Configuring delegated LDAP authentication
  - Connecting to Crowd
  - Connecting to JIRA for user management
- Setting up your mail server

Related pages:

- Supported platforms
- Using Stash
- Stash FAQ

System administration actions that can be performed outside of the Stash user interface include:

- Data recovery and backups
- Enabling Stash debug logging
- Integrating Stash with Apache HTTP Server
- Connecting Stash to an external database
- Scaling Stash
- System settings
- Securing Stash with HTTPS

Users and groups
Stash 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.

Stash Admins and Sys Admins can manage users and groups in Stash as described on this page. You can also set up Stash to use external user directories.

Note that access permissions can also be applied to projects.

On this page:
- Creating a user
- Creating a group
- Adding users to groups
  - From the user profile
  - From the group page

Related pages:
- Getting started with Stash
- External user directories

Creating a user

Go to the Users listing in the Administration section.

Click Create User to go directly to the user creation form:
Once you've created a user, click **Change permissions** to set up their access permissions.

**Creating a group**

Go to the **Group Listing** in the **Administration** section.
Enter the name of the group in the creation form and click **Create Group**.

**Groups**

Now you can add users to your new group:

**Adding users to groups**

**From the user profile**

Go to the user profile page in the **Administration** section and click **Add groups**.
**Users / User details**

John Doe

![Profile image]

- **jdoe**
- **john@doe.com**

Change password

**Groups**

![Add group button]

Filter groups...

stash-users

Remove group

You can use the filter to find the group you want to add the user to. Click on Add group button to add the user to the group.

You can use the filter to find the group you want to add the user to. Click on Add group button to add the user to the group.

Click **Done** when you have finished.

**From the group page**

Go to the group page and click Add members.
Groups / Mobile Developers

In the user picker click Add user to make the user a member of the group:

Click Done when you have finished.

Global permissions
Stash uses four levels of permissions to control user and group access to Stash projects and to the Stash server configuration.

<table>
<thead>
<tr>
<th></th>
<th>Login / Browse</th>
<th>Create projects</th>
<th>Manage users / groups</th>
<th>Manage global permissions</th>
<th>Edit application settings</th>
<th>Edit server config</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stash User</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Project Creator</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Admin</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sys Admin</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Note that access permissions can also be applied to projects.

Related pages:
- Getting started with Stash
- Users and groups

To edit the permissions for a Stash user or group:

1. Click Administration in the top menu.
2. Choose Accounts > Permissions.
3. Click the plus icon to add a user or group to a particular access level.
4. Click on a user or group name to remove it from an access level.

---

**JIRA integration**

When Stash is integrated with Atlassian JIRA, you can:

- see the JIRA issues related to particular commits in Stash (on the **Commits** tab for a Stash project).
- use JIRA for delegated user management. See [External user directories](#).

Stash integrates with JIRA by means of an 'application link'. You can set up application links either during the Stash **install process**, or as described below.

⚠️ Stash integration with JIRA requires an upgraded version of the FishEye plugin in JIRA. See [JIRA compatibility](#) for details about upgrading the JIRA FishEye plugin, and for download links to the upgraded plugin versions.

---

**On this page:**

- [Linking Stash with JIRA](#)
- [Known issues with the JIRA integration](#)

**Related pages:**

- [Installing Stash on Windows](#)
- [Installing Stash on Linux and Mac](#)
- [Configuring Application Links](#)
- [Connecting to JIRA for user management](#)

---

**Linking Stash with JIRA**

**To link Stash to a JIRA server:**

Go to the Application Links screen in Stash:
Click **Add Application Link**: 

Complete the application link wizard to connect Stash to your JIRA server. It is recommended that you make use of the automatic link-back from JIRA to Stash.

More detailed information about application links can be found on [Configuring Application Links](#).

Once finished you will not have to do any extra configuration. Your JIRA issues will appear in the changesets and commit lists in Stash. On the JIRA side, the commits associated with a specific issue will appear in the Source tab.

**Known issues with the JIRA integration**

We have tried to make the integration of JIRA with Stash as straightforward as possible. However, we are aware of the following issues:

- Stash only supports one JIRA server; we only pick the primary one.
- There is no checking for project or issue-key validity; Stash may link to issues that do not actually exist.

We apologise for the inconvenience, and are working hard to fix these for the next version. You can watch the 2 issues below to keep track of our progress:

- Error rendering macro 'jira': JIRA project does not exist or you do not have permission to view it.
- Error rendering macro 'jira': JIRA project does not exist or you do not have permission to view it.

**JIRA compatibility**
Atlassian JIRA’s support for Stash is built into the FishEye plugin that is bundled with JIRA. This allows you to see all of your code changes in one place, even if you’re running multiple Atlassian FishEye and Stash servers.

Supported JIRA Versions

If you’re using a version of JIRA earlier than 5.0.2 you may need to upgrade the FishEye plugin in JIRA to get support for Atlassian Stash.

<table>
<thead>
<tr>
<th>JIRA Version</th>
<th>Compatibility</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0.4+</td>
<td>Works straight out of the box!</td>
<td>NA</td>
</tr>
<tr>
<td>5.0-5.0.3</td>
<td>Requires JIRA FishEye Plugin 5.0.4.1</td>
<td><a href="https://maven.atlassian.com/content/repositories/atlassian-contrib/com/atlassian/jira/plugins/jira-fisheye-plugin/5.0.4.1/jira-fisheye-plugin-5.0.4.1.jar">URL</a></td>
</tr>
<tr>
<td>4.4.x</td>
<td>Requires JIRA FishEye Plugin 3.4.12</td>
<td><a href="https://maven.atlassian.com/content/repositories/atlassian-contrib/com/atlassian/jira/plugins/jira-fisheye-plugin/3.4.12/jira-fisheye-plugin-3.4.12.jar">URL</a></td>
</tr>
<tr>
<td>4.3.x</td>
<td>Requires JIRA FishEye Plugin 3.1.8</td>
<td><a href="https://maven.atlassian.com/content/repositories/atlassian-contrib/com/atlassian/jira/plugins/jira-fisheye-plugin/3.1.8/jira-fisheye-plugin-3.1.8.jar">URL</a></td>
</tr>
<tr>
<td>Earlier versions</td>
<td>JIRA-to-Stash integration is unsupported</td>
<td>NA</td>
</tr>
</tbody>
</table>

**JIRA 4.3.x Upgrade Note**

When upgrading the plugin in JIRA 4.3.x you may see a "zip file closed" error message in the logs. This can be ignored. See "IllegalStateException: zip file closed" when upgrading JIRA FishEye Plugin in JIRA 4.3 for more details.

**Upgrade Guide**

To upgrade the plugin, copy the link from the table above that matches your JIRA version. Then navigate in JIRA to **Administration > Plugins**.
Next, select the **Install Plugins** tab and click **Upload Plugin**.

Now paste the URL copied from the table above into the **From this URL** field, and click **Upload**.
You should see that the plugin is installed. Now you can continue integrating Atlassian Stash with your JIRA server. See [JIRA integration](#) for details.

**External user directories**

You can connect Stash to external user directories. This allows you to make use of existing users and groups stored in an enterprise directory.

Note that Stash 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.

⚠️ Connecting Atlassian Stash to your external directory will not be sufficient to allow your users to log in to Stash. You will need to grant them explicitly the right to use Atlassian Stash in the [global permission screen](#).

We recommend you to use groups instead of individual accounts when granting permissions.

**LDAP**

You can connect Stash to an LDAP directory for two purposes:

- For full user and group management, and for authentication — see the [Configuring an LDAP directory](#) page for instructions.
- For delegated authentication only, while still using the internal directory for user and group management — see the [Configuring delegated LDAP authentication](#) page for instructions.

You should consider connecting to an LDAP directory server if your users and groups are stored in an enterprise directory.
Stash 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

Crowd

You can connect Stash to Atlassian Crowd for user and group management, and for authentication.

Crowd is an application security framework that handles authentication and authorisation for your web-based applications. With Crowd you can integrate multiple web applications and user directories, with support for single sign-on (SSO) and centralised 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 the Connecting to Crowd page for configuration instructions.

JIRA

You can delegate Stash user and group management, and authentication, to an Atlassian JIRA instance.

You should consider connecting to JIRA as an alternative to using Atlassian Crowd for simple configurations with a limited number of users. Stash can only connect to a JIRA server running JIRA 4.3 or later.

See the Connecting to JIRA for user management page for configuration instructions.
Configuring an LDAP directory

You can connect Stash to an external LDAP user directory. This allows you to make use of existing users and groups stored in an enterprise directory.

Stash 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

To connect Stash to an LDAP directory:

1. Log in as a user with 'Admin' permission.
2. Click Administration in the top menu.
3. Choose Accounts > User Directories.
4. Click Add Directory and select either Microsoft Active Directory or LDAP as the directory type.
5. Configure the directory settings, as described in the tables below.
6. Save the directory settings.
7. Define the directory order by clicking the blue up- and down-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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a meaningful name to help you identify the LDAP directory server. Examples: * Example Company Staff Directory * Example Company Corporate LDAP</td>
</tr>
<tr>
<td>Directory Type</td>
<td>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.</td>
</tr>
<tr>
<td>Hostname</td>
<td>The host name of your directory server. Examples: * ad.example.com * ldap.example.com * opens.example.com</td>
</tr>
<tr>
<td>Port</td>
<td>The port on which your directory server is listening. Examples: * 389 * 10389 * 636 (for example, for SSL)</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Tick this check 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.</td>
</tr>
<tr>
<td>Username</td>
<td>The distinguished name of the user that the application will use when connecting to the directory server. Examples: * cn=administrator,cn=users,dc=ad,dc=example,dc=com * cn=user,dc=domain,dc=name * <a href="mailto:user@domain.name">user@domain.name</a></td>
</tr>
<tr>
<td>Password</td>
<td>The password of the user specified above.</td>
</tr>
</tbody>
</table>

LDAP schema

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

Created in 2012 by Atlassian. Licensed under a Creative Commons Attribution 2.5 Australia License.
| 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 formats with the domain name and local format for your specific configuration. Microsoft Server provides a tool called ldp.exe which is useful for finding out and configuring 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 |

**LDAP permission**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Only</td>
<td>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.</td>
</tr>
<tr>
<td>Read Only, with Local Groups</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Enable Nested Groups</td>
<td>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.</td>
</tr>
<tr>
<td>Use Paged Results</td>
<td>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.</td>
</tr>
<tr>
<td>Follow Referrals</td>
<td>Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup <code>java.naming.referal</code>) configuration setting. It is generally needed for Active Directory servers configured without proper DNS, to prevent a <code>javax.naming.PartialResultException: Unprocessed Continuation Reference(s)</code> error.</td>
</tr>
<tr>
<td>Naive DN Matching</td>
<td>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 DNs to determine if they are equal.</td>
</tr>
<tr>
<td></td>
<td>• If this check box is ticked, 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 DNs.</td>
</tr>
<tr>
<td></td>
<td>• If this check box is not ticked, the application will parse the DN and then check the parsed version.</td>
</tr>
</tbody>
</table>
Enable Incremental Synchronisation | Enable incremental synchronisation if you only want changes since the last synchronisation to be queried when synchronising a directory.

⚠️ Please be aware that when using this option, the user account configured for synchronisation must have read access to:

- The `uSNC` attribute of all users and groups in the directory that need to be synchronised.
- 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 JIRA.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronisation Interval (minutes)</td>
<td>Synchronisation 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.</td>
</tr>
<tr>
<td>Read Timeout (seconds)</td>
<td>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.</td>
</tr>
<tr>
<td>Search Timeout (seconds)</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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. |
<p>| User schema settings | |</p>
<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Object Class</td>
<td>This is the name of the class used for the LDAP user object. Example: <code>user</code></td>
</tr>
</tbody>
</table>
### User Object Filter

The filter to use when searching user objects. Example:

\[
(&(\text{objectCategory}=\text{Person})(\text{sAMAccountName}=*))
\]

### User Name Attribute

The attribute field to use when loading the username. Examples:

- `cn`
- `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`

### User Password Attribute

The attribute field to use when loading a user's password. Example:

- `unicodePwd`

---

**Group schema settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Object Class</td>
<td>This is the name of the class used for the LDAP group object. Examples:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>groupOfUniqueNames</code></td>
</tr>
<tr>
<td></td>
<td><code>group</code></td>
</tr>
<tr>
<td><strong>Group Object Filter</strong></td>
<td>The filter to use when searching group objects. Example: <code>&lt;objectCategory=Group&gt;</code></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Group Name Attribute</strong></td>
<td>The attribute field to use when loading the group's name. Example: <code>&lt;cn&gt;</code></td>
</tr>
<tr>
<td><strong>Group Description Attribute</strong></td>
<td>The attribute field to use when loading the group's description. Example: <code>&lt;description&gt;</code></td>
</tr>
</tbody>
</table>

**Membership schema settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group Members Attribute</strong></td>
<td>The attribute field to use when loading the group's members. Example: <code>&lt;member&gt;</code></td>
</tr>
<tr>
<td><strong>User Membership Attribute</strong></td>
<td>The attribute field to use when loading the user's groups. Example: <code>&lt;memberOf&gt;</code></td>
</tr>
</tbody>
</table>

**Use the User Membership Attribute, when finding the user's group membership**

Put a tick in the checkbox if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.)

- If this checkbox is ticked, 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 checkbox is not ticked, your application will use the members attribute on the group ('member' by default) for the search.
- If the 'Enable Nested Groups' checkbox is ticked, your application will ignore the 'Use memberOf Attribute on the User' 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

Put a tick in the checkbox if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.)

- If this checkbox is ticked, 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 search.
- If this checkbox is not ticked, your application will use the members attribute on the group ('member' by default) for the search.

Configuring delegated LDAP authentication

You can configure Stash to use an LDAP directory for delegated authentication, while still using the internal directory for user and group management. There is an option to create users in the internal directory automatically when they attempt to log in, as described in the settings section below.

To connect Stash to an LDAP directory for delegated authentication:

1. Log in as a user with ‘Admin’ permission.
2. Click Administration in the top menu.
3. Choose Accounts > User Directories.
4. Click Add Directory and select Internal with LDAP Authentication as the directory type.
5. Configure the directory settings, as described in the tables below.
6. Save the directory settings.
7. Define the directory order by clicking the blue up- and down-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.

On this page:

- Server settings
- Copying users on login
- LDAP schema
- Advanced settings
- User schema settings
- Group schema settings
- Membership schema settings

Server settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A descriptive name that will help you to identify the directory. Examples:</td>
</tr>
<tr>
<td></td>
<td>• Internal directory with LDAP Authentication</td>
</tr>
<tr>
<td></td>
<td>• Corporate LDAP for Authentication Only</td>
</tr>
</tbody>
</table>
| 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 | Select this check 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=adminstrator,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. |

**Copying users on login**

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

### Copy User on Login

This option affects what will happen when a user attempts to log in. If this check box is selected, 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 synchronised on each subsequent log in. If this check box is not selected, the user’s login will fail.

If you select this check box the following additional fields will appear on the screen, which are described in more detail below:

- Default Group Memberships
- Synchronise Group Memberships
- User Schema Settings (described in a separate section below)

### Default Group Memberships

This field appears if you select the **Copy User on Login** check 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, authorisation failures will result – users will not be able to access the applications or functionality based on the intended group name.

**Examples:**

- `confluence-users`
- `confluence-users,jira-users,jira-developers`

### Synchronise Group Memberships

This field appears if you select the **Copy User on Login** check box. If this check box is selected, group memberships specified on your LDAP server will be synchronised with Confluence each time the user logs in.

If you select this check 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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base DN</td>
<td>The root distinguished name (DN) to use when running queries against the directory server. Examples:</td>
</tr>
<tr>
<td></td>
<td>• o=example,c=com</td>
</tr>
<tr>
<td></td>
<td>• cn=users,dc=ad,dc=example,dc=com</td>
</tr>
<tr>
<td></td>
<td>• 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 ldp.exe which is useful for finding out and configuring the the LDAP structure of your server.</td>
</tr>
<tr>
<td>User Name Attribute</td>
<td>The attribute field to use when loading the username. Examples:</td>
</tr>
<tr>
<td></td>
<td>• cn</td>
</tr>
<tr>
<td></td>
<td>• sAMAccountName</td>
</tr>
</tbody>
</table>

## Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Paged Results</td>
<td>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.</td>
</tr>
<tr>
<td>Follow Referrals</td>
<td>Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup java.naming.referral) 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.</td>
</tr>
</tbody>
</table>

## User schema settings

Note: this section is only visible when **Copy User on Login** is enabled.
| **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 **Synchronise Group Memberships** are enabled.

<table>
<thead>
<tr>
<th><strong>Setting</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>

Created in 2012 by Atlassian. Licensed under a Creative Commons Attribution 2.5 Australia License.
### Documentation for Stash 1.0

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

---

| 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:  
<table>
<thead>
<tr>
<th></th>
<th>• ou=Groups</th>
</tr>
</thead>
</table>
| Group Object Class  | This is the name of the class used for the LDAP group object. Examples:  
|                     | • groupOfUniqueNames  
<table>
<thead>
<tr>
<th></th>
<th>• group</th>
</tr>
</thead>
</table>
| Group Object Filter | The filter to use when searching group objects. Example:  
<table>
<thead>
<tr>
<th></th>
<th>• (objectCategory=Group)</th>
</tr>
</thead>
</table>
| Group Name Attribute| The attribute field to use when loading the group’s name. Example:  
<table>
<thead>
<tr>
<th></th>
<th>• cn</th>
</tr>
</thead>
</table>
| 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 **Synchronise Group Memberships** are enabled.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Members Attribute</td>
<td>The attribute field to use when loading the group’s members. Example:</td>
</tr>
<tr>
<td></td>
<td>• member</td>
</tr>
<tr>
<td>User Membership Attribute</td>
<td>The attribute field to use when loading the user’s groups. Example:</td>
</tr>
<tr>
<td></td>
<td>• memberOf</td>
</tr>
</tbody>
</table>
| Use the User Membership Attribute, when finding the user’s group membership | Select the check box if your directory server supports the group membership attribute on the user. (By default, this is the ‘memberOf’ attribute.)  
|                                                    | • If this check box 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 retrieval.  
|                                                    | • If this check box is not selected, your application will use the members attribute on the group ('member' by default) for the search. |
Connecting to Crowd
You can configure Stash to use Atlassian Crowd for user and group management, and for authentication.

Atlassian Crowd is an application security framework that handles authentication and authorisation for your web-based applications. With Crowd you can integrate multiple web applications and user directories, with support for single sign-on (SSO) and centralised 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.

### On this page:

- Server settings
- Crowd permissions
- Advanced settings

To connect Stash to Crowd:

1. Log in as a user with 'Admin' permission.
2. Click **Administration** in the top menu.
3. Choose **Accounts > User Directories**.
4. Click **Add Directory** and select **Atlassian Crowd**.
5. Enter settings, as described below.
6. Test and save the directory settings.
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.

### Server settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A meaningful name that will help you to identify this Crowd server amongst your list of directory servers. Examples:</td>
</tr>
<tr>
<td></td>
<td>- Crowd Server</td>
</tr>
<tr>
<td></td>
<td>- Example Company Crowd</td>
</tr>
<tr>
<td>Server URL</td>
<td>The web address of your Crowd console server. Examples:</td>
</tr>
<tr>
<td></td>
<td>- <a href="http://www.example.com:8095/crowd/">http://www.example.com:8095/crowd/</a></td>
</tr>
<tr>
<td></td>
<td>- <a href="http://crowd.example.com">http://crowd.example.com</a></td>
</tr>
<tr>
<td>Application Name</td>
<td>The name of your application, as recognised 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 <a href="#">adding an application</a>.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Only</td>
<td>The users, groups and memberships in this directory are retrieved from Crowd and can only be modified via Crowd. You cannot modify Crowd users, groups or memberships via the application administration screens.</td>
</tr>
<tr>
<td>Read/Write</td>
<td>The users, groups and memberships in this directory are retrieved from Crowd. When you modify a user, group or membership via the application administration screens, the changes will be applied directly to Crowd. Please ensure that the application has modification permissions for the relevant directories in Crowd. See the Crowd documentation: Specifying an Application's Directory Permissions.</td>
</tr>
</tbody>
</table>

Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Nested Groups</td>
<td>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.</td>
</tr>
<tr>
<td>Synchronisation Interval (minutes)</td>
<td>Synchronisation 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.</td>
</tr>
</tbody>
</table>

Connecting to JIRA for user management

You can connect Stash to an Atlassian JIRA instance to delegate Stash user and group management, and authentication.

Choose this option as an alternative to Atlassian Crowd, for simple configurations with a limited number of users. Note that Stash can only connect to a JIRA server running JIRA 4.3 or later.
Connecting Stash and JIRA is a 3-step process:

1. Set up JIRA to allow connections from Stash.
2. Set up Stash to connect to JIRA.
3. Set up Stash users and groups in JIRA.

You can connect to JIRA either when you first run Stash, using the Setup Wizard, or at any time after setup is complete.

If using the Stash Setup Wizard to configure JIRA integration, we recommend that you make use of the automatic back-linking from JIRA to Stash.

⚠️ You need to be an administrator in both JIRA and Stash to do this.

### Connecting to JIRA

1. Set up JIRA to allow connections from Stash

   1. Log in as a user with the 'JIRA Administrators' global permission.
   2. For JIRA 4.3.x, select **Other Application** from the 'Users, Groups & Roles' section of the 'Administration' menu.
      For JIRA 4.4 or later, choose **Administration > Users > JIRA User Server**.
   3. Click **Add Application**.
   4. Enter the **application name** and **password** that Stash will use when accessing JIRA.
   5. Enter the **IP address** of your Stash server. 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](https://en.wikipedia.org/wiki/CIDR_notation) and [RFC 4632](https://tools.ietf.org/html/rfc4632).
   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. Set up Stash to connect to JIRA

   1. Log in as a user with 'Admin' permission.
   2. Click **Administration** in the top menu and then **User Directories**.
   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 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.
3. Set up Stash users and groups in JIRA

In order to use Stash, users must be a member of the Stash-users group or have Stash 'can use' permission. Follow these steps to configure your Stash groups in JIRA:

1. Add the stash-users and stash-administrators groups in JIRA.
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 users access to Stash:
   - Option 1: In JIRA, find the groups that the relevant users belong to. Add the groups as members of one or both of the above Stash groups.
   - Option 2: Log in to Stash using your JIRA account, click Administration and then Permissions. Assign the appropriate permissions to the relevant JIRA groups.

Server settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A meaningful name that will help you to identify this JIRA server amongst your list of directory servers. Examples:</td>
</tr>
<tr>
<td></td>
<td>• JIRA Server</td>
</tr>
<tr>
<td></td>
<td>• My Company JIRA</td>
</tr>
<tr>
<td>Server URL</td>
<td>The web address of your JIRA server. Examples:</td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.example.com:8080">http://www.example.com:8080</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://jira.example.com">http://jira.example.com</a></td>
</tr>
<tr>
<td>Application Name</td>
<td>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 &amp; Roles' section of the 'Administration' menu.</td>
</tr>
<tr>
<td>Application Password</td>
<td>The password used by your application when accessing the JIRA server that acts as user manager.</td>
</tr>
</tbody>
</table>

JIRA server permissions

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Only</td>
<td>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.</td>
</tr>
</tbody>
</table>
Read/Write

The users, groups and memberships in this directory are retrieved from the JIRA server that is acting as user manager. When you modify a user, group or membership, the changes will be applied directly to your application and to the JIRA server that is acting as user manager.

Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Nested Groups</td>
<td>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.</td>
</tr>
<tr>
<td>Enable Incremental Synchronisation</td>
<td>Enable or disable incremental synchronisation. Only changes since the last synchronisation will be retrieved when synchronising a directory.</td>
</tr>
<tr>
<td>Synchronisation Interval (minutes)</td>
<td>Synchronisation 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.</td>
</tr>
</tbody>
</table>

Setting up your mail server

Setting up your mail server will allow you to:

- Send emails when creating a user so that they can generate their own password.
- Allow your users to reset their password if they forget it.

Go to the ‘Mail server configuration’ page in the Administration.
Administration

Accounts

Users
Edit individual user details, passwords and groups membership

Groups
Create groups and manage the users within each group

Permissions
Configure global user permissions

Avatars
Configure user avatar settings

User Directories
Connect Stash to user directory servers - Active Directory, Crowd, LDAP and JIRA

Security

Authentication
Configure access, privacy and spam prevention settings

Applications & Plugins

Plugins
Install and manage your plugins

Application Links
Configuring links to other applications, including Atlassian

Settings

General settings
Set this server's name, URL and email address

Mail server
Configure the SMTP server used to send notification emails

Licensing
License information and configuration

I18n
Language and timezone selection

Fill in the form and click Save.

Mail server configuration

Mail settings

Hostname
smtp.mycompany.com
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.

Use TLS

Tick if the SMTP server you are connecting to uses TLS

Email from
stash@mycompany.com
Specifies the From: header in notification emails (for example: noreply@yourcompany.com).

Save Delete Cancel

Advanced actions

This section describes the administrative actions that can be performed from outside of the Stash Administration user interface.
Data recovery and backups

Stash does not currently have any built in data backup or recovery solutions. **We highly recommend** that you put in place a data recovery plan which is in line with your company’s policies.

⚠️ **Important:** Stash should be shutdown before taking any backups, to ensure data consistency.

---

The simplest and most effective backup solution is to make a backup of the entire Stash home directory. Your Stash home directory contains the following directories and files:

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Contains all repository data and embedded HSQL database if an external database is not configured. This is the most important directory in the home directory.</td>
</tr>
<tr>
<td>log</td>
<td>Contains logging files for Stash</td>
</tr>
<tr>
<td>plugins</td>
<td>Contains plugin related data (such as externally uploaded plugins) for Stash.</td>
</tr>
<tr>
<td>tmp</td>
<td>Temporary directory for run-time related operations. Can be safely deleted when the Stash is <em>not running</em>.</td>
</tr>
<tr>
<td>stash-config.properties</td>
<td>File which contains configuration properties for Stash.</td>
</tr>
</tbody>
</table>

If you are using Stash with an external database system (such as PostgreSQL), then please remember to backup your database schema as well. Please speak to your DBA or refer to your database software backup guidelines.

---

Enabling Stash debug logging

This page describes how to enable debug level logging in Stash. Stash logs can be found in `<STASH_HOME>/logs`.
When using the standard Stash distribution, logs for the Tomcat webserver that hosts Stash can be found in `<stash installation directory>/log`.

### Enabling debug logging on startup

To enable debug logging whenever Stash is started, edit the `<STASH_HOME>/stash-config.properties` file and add the following line:

```
logging.logger.ROOT=DEBUG
```

### Enabling debug logging at runtime

To enable debug logging for the root logger once Stash has been started, run the following command in your terminal:

```bash
curl -u <ADMIN_USERNAME> -v -X PUT -d "" -H "Content-Type: application/json" 
<BASE_URL>/rest/api/latest/logs/rootLogger/debug
```

# e.g.
```
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json"
http://localhost:7990/rest/api/latest/logs/rootLogger/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/debug

Integrating Stash with Apache HTTP Server

This page explains how to establish a network topology in which Apache HTTP Server acts as a reverse proxy for Stash. Typically, such a configuration would be used when Stash is installed in a protected zone "behind the firewall", and Apache HTTP Server provides a gateway through which users outside the firewall can access Stash.

Be aware that Stash does not need to run behind a web server, since it is capable of serving web requests directly; to secure Stash when run in this way see Securing Stash with HTTPS. Otherwise, if you want to install Stash in an environment that incorporates Apache HTTP Server, this document is for you.

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 6

The Stash distribution includes an instance of Tomcat 6, the configuration of which is determined by the contents of the server.xml file, which can be found in the conf directory immediately under the Stash installation directory. Note that any changes that you make to the server.xml file will be effective upon starting or re-starting Stash.

You may find it helpful to refer to the Apache Tomcat 6.0 Proxy Support HowTo page.
On this page:

- About using Apache software
- Step 1: Configure the Tomcat Connector
- Step 2 (optional): Set a context path for Stash
- Step 3: Enable mod_proxy and mod_proxy_http in Apache HTTP Server
- Step 4: Configure mod_proxy to map requests to Stash
- Step 5: Configure mod_proxy to disable forward proxying
- Step 6: Allow proxying to Stash from everywhere
- Step 7 (optional): Configure Apache HTTP Server for SSL
- Troubleshooting

Related pages:

- Securing Stash with HTTPS

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.

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 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 server.xml file, and add the scheme, proxyName, and proxyPort attributes as shown below. Instead of mycompany.com, set the proxyName attribute to the domain name that Apache HTTP Server will be configured to serve. This informs Stash 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 Stash functions that construct URL's.
<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 recognised by Stash instead of the hostname at which Stash 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 6.0 HTTP Connector Reference](#).

**Step 2 (optional): Set a context path for Stash**

By default, Stash 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, Stash is accessed at:

```
http://localhost:7990/
```

It's perfectly fine to run Stash with the empty context path as above. Alternatively, you can set a context path by changing the `Context` directive in Tomcat's `server.xml` file:
<Context path="/stash"
docBase="${catalina.home}/atlassian-stash.war" reloadable="false"
useHttpOnly="true">
  ....
</Context>

If you do set a context path, it is important that the same path be used in Step 4, when setting up the ProxyPass and ProxyPassReverse directives.

**Step 3: 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 Stash server. 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 Stash, since it has proven itself to be less reliable than mod_proxy.

**Step 4: Configure mod_proxy to map requests to Stash**

To configure mod_proxy for use with Stash, 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 (Stash) 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 Stash.

The `timeout` attribute specifies the number of seconds Apache HTTP Server waits for data to be sent to Stash.

If you set up a context path for Stash in Step 2, you'll need to use that context path in your `ProxyPass` and `ProxyPassReverse` directives. Suppose your context path is set to "/stash", the directives would be as follows:

```
ProxyPass /stash
http://localhost:7990/stash
collectiontimeout=5 timeout=300
ProxyPassReverse /stash
http://localhost:7990/stash
```

If Stash 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 Stash will run on port 9900 on `private.mycompany.com` under the context path /stash, then you would use the following directives:

```
ProxyPass /stash
http://private.mycompany.com:9900/stash
collectiontimeout=5 timeout=300
ProxyPassReverse /stash
http://private.mycompany.com:9900/stash
```

Step 5: 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 6: Allow proxying to Stash from everywhere

Strictly speaking, this step is unnecessary because access to proxied resources is unrestricted by default. Nevertheless, we explicitly allow access to Stash 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:
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 Stash via Apache HTTP Server. Here, we specify that all hosts are allowed access to Stash.

**Step 7 (optional): Configure Apache HTTP Server for SSL**

If you want to set up SSL access to Stash, take steps 7(a) to 7(d) below. When you are finished, users will be able to make secure connections to Apache HTTP Server; connections between Apache HTTP Server and Stash 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 (Stash), but that configuration is very unusual, and not recommended in most circumstances.

**Step 7(a): Configure the Tomcat Connector for SSL**

Find the normal (non-SSL) `Connector` directive in Tomcat's `server.xml` file, and change the `redirectPort`, `scheme` 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"
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 Stash 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.

Step 7(b): 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"
  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 `ProxyPass` and `ProxyPassReverse` directives should be set up in manner described in Step 4.

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 directory>/conf/extra/httpd-ssl.conf file, which is included in the standard Apache distribution.

**Step 7(c): Create SSL certificate and key files**

In Step 7(b), 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:

```bash
openssl genrsa -des3 -out server.key 1024
```

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.

Generate a certificate request file (server.csr):

```bash
openssl req -new -key server.key -out server.csr
```

Generate a self-signed certificate (server.crt):

```bash
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 7(b).

```bash
cp server.key /usr/local/apache2/conf/
cp server.crt /usr/local/apache2/conf/
```

Step 7(d): Change Stash’s base URL

After re-starting Stash and Apache HTTP Server, open a browser window and log into Stash using an administrator account. Visit Administration > Settings > General Settings, and change the base URL to match the context path that you set in Step 2. Don't forget to use 'https'.

Using a self-signed certificate

There are two implications of using the self-signed certificate:

- When you access Stash 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.

Troubleshooting

- 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.
- In general, if you are having problems:
  1. Ensure that Stash works as expected when running directly from Tomcat on http://localhost:7990/stash.
  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 Stash Knowledge Base.

Connecting Stash to an external database

Currently, Stash supports connecting to the following external databases:

- PostgreSQL

⚠️ Stash does not currently support migration between database systems. If you would like to connect the application to an external database, you must do so before starting Stash for the first time. Doing otherwise will result in data loss.

```
Related pages:
- Installing Stash on Windows
- Installing Stash on Linux and Mac
- Supported platforms
```
In your Stash Home directory, create the `stash-config.properties` file if it does not already exist. The following lines should be added to connect to an external database, in this case PostgreSQL:

```properties
jdbc.url=jdbc:postgresql://localhost:5432/stash
jdbc.user=stash_user
jdbc.password=s3cr3t
jdbc.driver=org.postgresql.Driver
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jdbc.url</td>
<td>This is the JDBC url that Stash will use to connect to the database. This should include the driver subprotocol (e.g., <code>postgresql:</code>), the hostname, port and database that you will connect to. This string may vary depending on the database you are connecting to. Please seek specific examples for other databases with your database provider.</td>
</tr>
<tr>
<td>jdbc.user</td>
<td>This is the user that Stash will connect to the database with. The user will need to be able to create and drop tables and indexes, as well as read and write operations on the entire database schema defined in <code>jdbc.url</code>.</td>
</tr>
<tr>
<td>jdbc.password</td>
<td>The password that the user defined by <code>jdbc.user</code> will connect with.</td>
</tr>
<tr>
<td>jdbc.driver</td>
<td>The JDBC driver class that should be used by Stash to connect to the database.</td>
</tr>
</tbody>
</table>

If none of the above values are specified in `stash-config.properties`, then a provided HSQL database will be used.

## Scaling Stash

### Hardware requirements

The type of hardware you require to run Stash depends on a number of factors:

- The number of users.
- The size of your repositories. On large repositories, many operations require more memory and more CPUs.
- The 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, Stash will be asked to perform multiple clones concurrently, putting significant load on your system.

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.
- Estimate or calculate the average repository size and allocate 1.5 x number of concurrent clone operations.
operations x min(repository size, 700MB) of memory.

Understanding Stash’s resource usage

Most of the things you do in Stash involve both the Stash server and one or more Git processes created by Stash. For instance, when you view a file in the Stash web application, Stash processes the incoming HTTP request, performs permission checks, creates a Git process to retrieve the file contents and formats the resulting webpage. In serving most pages, both the Stash server and Git processes are involved. The same is true for the ‘hosting’ operations: pushing your commits to Stash, cloning a repository from Stash or fetching the latest changes from Stash.

As a result, when configuring Stash for performance, CPU and memory consumption for both Stash and Git should be taken into account.

Memory

When deciding on how much memory to allocate for Stash, 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 Stash and cloning large repositories from Stash. 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.

Stash 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 Stash 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 Stash. The maximum amount of memory available to Stash 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 Stash, much of the heavy lifting is delegated to Git. As a result, when deciding on the required hardware to run Stash, 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.

**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:

---

**CPU Usage - Clone operation**

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>CPU Usage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
</tr>
</tbody>
</table>

**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.

**Stash (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.

---

**Memory Usage - Clone**

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>Memory Usage (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>270</td>
</tr>
<tr>
<td>20</td>
<td>270</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
</tr>
</tbody>
</table>

**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.

**Stash (red line)**

- Memory usage hovers around 800 MB and is not affected by the clone operation.
Configuring Stash scaling options and system properties

Stash only allows a fixed number of Git commands to be executed concurrently, to prevent the performance for all clients dropping below acceptable levels. Stash 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).

The settings can be overridden by creating a stash-config.properties in STASH_HOME with the following content:

```
# The maximum number of concurrent requests using git commands using the UI or REST services (e.g. git diff via the UI). Default value is 25
throttle.resource.scm-command=20

# The maximum number of concurrent requests using 'hosting' commands, git clone, git push, git pull. Default value is 50
throttle.resource.scm-servlet=20
```

What happens when the limits are reached?
For the given resource, Stash will start rejecting requests of that type.

When the non-servlet limit is reached, users will start seeing messages in the UI indicating that the server is under load.

When the servlet limit is reached, it can be a little harder to determine for a user, so Stash does a few different things:

- Stash will return either a 503 or a 501 response code to the Git client as described in [Git commands return error code 503](#) and [Git commands return error code 501](#).
- A warning message will be logged for every time a request is rejected due to the resource limits.
- For five minutes after a request is rejected, Stash will display a red banner in the UI to warn that the server is under load.

### System settings

This page lists the Java VM system properties which may be set to control certain aspects of Stash's behaviour.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Values</th>
<th>Default</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>file.encoding</td>
<td>utf-8</td>
<td>utf-8</td>
<td>The Stash WAR needs to be run with the file.encoding property set to utf-8 in the JVM.</td>
</tr>
</tbody>
</table>

**To set JVM system properties:**

There are 2 approaches to setting JVM system properties:

- The recommended approach is to add the property to `<Stash installation directory>/system.properties`. This file can be created as a plain text file (it follows the Java properties format).
- You can also pass the -D parameter to the JVM.

### Securing Stash with HTTPS

This page is intended for administrators setting up Stash for a small team, and describes how to enable SSL access for Tomcat, the webserver distributed with Stash, using a self-signed certificate. You should consider running Stash with HTTPS (HTTP over SSL) and making secure access mandatory, if Stash will be internet-facing where usernames, passwords and other proprietary data may be at risk.

Those setting up a production instance should consider using a CA certificate, briefly described below.

Be aware that you can set up Stash to run behind a web server, such as Apache HTTP Server. To secure Stash when Apache HTTP Server acts as a reverse proxy for Stash see [Integrating Stash with Apache HTTP Server](#).

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.

---

**On this page:**

- [1. Generate a self-signed certificate](#)
- [2. Configure HTTPS in Tomcat](#)
- [Troubleshooting](#)
- [Exporting the self-signed certificate](#)
- [Requesting a CA certificate](#)
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).

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, for Java 1.6. Other tools for generating certificates are available.

To generate a self-signed certificate:
- Log in with the user account that Stash will run under, and run the following command:

<table>
<thead>
<tr>
<th>Windows</th>
<th>&quot;$JAVA_HOME\bin\keytool&quot; -genkey -alias tomcat -keyalg RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, MacOS and Unix</td>
<td>$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA</td>
</tr>
</tbody>
</table>

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.

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 Stash. This is the name you would type in your web browser after ‘http://’ (no port number) to access your Stash installation. The qualified host name should match the base URL you have set in Stash (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 conf/server.xml by adding the following attribute to the <Connector/> tag: keystorePass="<password value>"

2. Configure HTTPS in Tomcat

To configure HTTPS in Tomcat:

- Edit conf/server.xml and, at the bottom, before the </Service> tag, add this section (or uncomment it 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 instead of 443 to
avoid conflicts).

Troubleshooting

Here are some troubleshooting tips if you are using a self-signed key created by keytool, 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 IE. 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/Linux the home directory is likely to be /home/<username>. For Windows it is likely to be C:\User\<UserName>.

Make sure you are running Stash as the same user who created the keystore. If this is not the case, or if you are running Stash on Windows as a service, you will need to specify where the keystore file is in conf/server.xml. Add the following attribute to the connector tag you uncommented:

```
keystoreFile="<location of keystore file>"
```

Incorrect password

```
jav.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 `<conf/server.xml>`.

Add the `keyAlias` attribute to the Connector tag you uncommented, with the relevant alias, for example:

```xml
<Connector port="8443"
   maxHttpHeaderSize="8192"
   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:

```xml
<Connector port="8443"
    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.
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.

```xml
<Connector
    ...
    clientAuth="true"
    ... />
```

For more information about Connector element parameters, please refer to the ‘SSL Support’ section of the Tomcat 6.0 documentation.

Exporting the self-signed certificate

If Stash 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 Stash will not be run as the user executing keytool --genkey. You can do so with the following command:

<table>
<thead>
<tr>
<th>Windows</th>
<th>&quot;%JAVA_HOME%/bin/keytool&quot; -export -alias tomcat -file file.cer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, MacOS</td>
<td>$JAVA_HOME/bin/keytool -export -alias tomcat -file file.cer</td>
</tr>
</tbody>
</table>
If you generate the certificate as one user and run Stash 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 Stash in a production environment, you will need a certificate issued by a CA, such as VeriSign, Thawte or TrustCenter. The instructions below are adapted from the Tomcat documentation.

First, you will generate a local certificate and create a 'certificate signing request' (CSR) based on that certificate. You 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**
   
   `"%JAVA_HOME%\bin\keytool" -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <MY_KEYSTORE_FILENAME>

   **Linux, MacOS and Unix**
   
   `$JAVA_HOME/bin/keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <MY_KEYSTORE_FILENAME>

3. Submit the generated file called `certreq.csr` to your chosen certificate authority. Refer to the documentation on the CA's website to find out how to do this.
4. The CA will send you a certificate.
5. Import the new certificate into your local keystore. 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**
   
   `"%JAVA_HOME%\bin\keytool" -import -alias tomcat -file file.cer

   **Linux, MacOS and Unix**
   
   `$JAVA_HOME/bin/keytool -import -alias tomcat -file file.cer

**Releases**

**Stash 1.0 is released!**

Atlassian Stash is a repository management solution that allows everyone in your organisation 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.

Stash 1.0 Release Notes
1st May 2012

Atlassian is proud to present Stash 1.0, which provides a central place to create and manage Git repositories. It's the place where all that distributed code comes back together, where you can find the latest official version of your project, and where you can keep track of what's going on.

Highlights of this release:

- Git repository management
- Projects and permissions
- Built for Git, focused on Enterprise
- Browse your source and history
- JIRA integration

Providing feedback:

Please log your votes and issues. They help us decide what needs doing, and are much appreciated!

1

Git repository management

Stash provides a simple and powerful interface to create and manage Git repositories. Create repositories in a couple of clicks, and quickly choose those of your users and groups who will be contributors to the project, and those who will be just observers.

2

Projects and permissions
Since projects rarely consist of a single repository, Stash provides a convenient **Project structure**. This helps you to organise and manage repositories, and makes managing access to your repositories really simple.

With Stash you can empower end users to manage repositories themselves, while keeping control of the key administration functions. And because we want to make it easy for you to manage teams, Stash has a group management feature to help you grant permissions across your organisation.

Built for Git, focused on Enterprise

Stash has everything you need to create and manage Git repositories efficiently behind the safety of your own firewall.

Stash doesn't force administrators to use a pre-packaged appliance and so give up control. Whether on **Windows**, **Linux** or **MacOS X**, Stash will feel right at home on all platforms.

With **LDAP**, **Crowd** and **JIRA** support, you can manage your team easily, whether they are a small number of users in Stash's internal directory, or 500 developers managed in your corporate directory.

### User Directories

The table below shows the user directories currently configured for Stash.

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 in the file directory where Stash has permission to make changes. It is recommended that each user exist only in a single directory.

<table>
<thead>
<tr>
<th>Directory Name</th>
<th>Type</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stash Internal Directory</td>
<td>Internal</td>
<td>🌟</td>
</tr>
<tr>
<td>Extranet Crowd</td>
<td>Atlassian Crowd</td>
<td>🌟</td>
</tr>
<tr>
<td>extranet-internal-directory (inactive)</td>
<td>Internal</td>
<td>🌟</td>
</tr>
</tbody>
</table>

Keep track of commits to the repositories you’re involved with and dive into the changesets to see exactly what has changed in the source. Use Stash’s user interface to quickly navigate your repository and view annotated changes in an easily digestible way.
JIRA integration saves users time when tracking and checking their development. Stash keeps track of all issues that are associated with commits. This allows users to navigate straight to the JIRA issues that are linked to the commits, and to see in JIRA an aggregate of all code changes related to an issue.

The Stash 1.0 team

Development

Core team

Adam Ahmed  
Brendan Humphreys  
Bryan Turner  
Conor MacNeill  
David Pinn  
Federico Silva Armas  
Geoff Crain  
Jason Hinch  
Jonathan Poh  
Michael McGlynn  
Michael Studman  
Nick Pellow  
Pierre-Etienne Poirot  
Xu-Heng Tjhin

Team leads

Matt Watson  
Seb Ruiz  
Tim Pettersen

Architect

Michael Heemskerk

Project manager
Anton Mazkovo

Support

Ajay Sridhar
Armen Khachatryan
Daniel Rohan
Douglas Fabretti
Felipe Kraemer
Gurleen Anand
Renan Battaglin
Rene Verschoor
Zed Yap

Others

Product management

Jens Schumacher
Sten Pittet

Design and user experience

Jake Causby
Matt Bond

Product marketing

Giancarlo Lionetti
Jeff Park

Technical writing

Paul Watson

Operations

James Fleming

Git resources

Get Git

Mac: http://code.google.com/p/git-osx-installer/downloads/list?can=3

Linux: http://book.git-scm.com/2_installing_git.html

Ubuntu Linux: https://launchpad.net/~git-core/+archive/ppa

Windows: Full installer for official Git for Windows

Basic Git

Basic Git commands

Git cheat sheets and other resources
Stash FAQ

Installation

Q: I see a blank page with a message saying “Better go get the administrator; something’s gone haywire in the server room”.

A: Chances are that you do not have the right version of Git installed. Atlassian Stash require Git version 1.7.6 or later. See Supported platforms. You can find download links in the Git resources.

Repositories

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’.

Q: Does Stash support Mercurial (Hg)? What about other version control systems?

A: Currently Stash does not support Mercurial. We will be gauging demand for Mercurial support as we move
forward.

Q: What about Git repository management in FishEye and Crucible?

A: The current Git repository management feature in FishEye will be deprecated in the near future. We encourage those interested in Git repository management to check out Stash. There is a promotional discount of 25% going on until June 30, 2012.

Q: Why did you create a new product for Git repository management? Couldn’t you build this into FishEye?

A: We thought we could. 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. Stash – Host, manage and collaborate on Git repositories
2. FishEye – Track, search and browse Subversion, Perforce, Git, Mercurial and CVS repositories in one place.

Integration

Q: Does Stash work with JIRA? If so, what version of JIRA do I need to run Stash?

A: Stash works with JIRA 4.3+. However, you will require the latest version of the JIRA/FishEye plugin to view commits in JIRA. See JIRA integration.

Q: Will Stash integrate with any other Atlassian Tools? Crowd? BitBucket? SourceTree?

A: Stash currently integrates with the JIRA issues tracker, SourceTree DVCS Mac client and Crowd user management solution. You can also connect to Stash via Bamboo to run your builds and deployments and we are planning even tighter integrations in the future.

Q: Will Stash be available for Atlassian OnDemand?

A: Atlassian Stash will not be available in OnDemand. If you are looking for a distributed version control solution to use with Atlassian OnDemand, we recommend using Bitbucket, our cloud based Git and Mercurial source code hosting solution. Bitbucket connects to Atlassian OnDemand via the JIRA DVCS connector.

Licensing

Q: The user tiers only go to 500 users - what do I do if I need more?

A: We have plans to release larger licenses in the not too distant future. Please contact us if you are interested in more than 500 users.
Q: I'm evaluating/have a quote for Stash and want to buy, but the promotion just ended. Am I still eligible for the discount?

A: All quotes will be honored up to 90 days past the end date (June 30, 2012) of the promotion period.

Q: How long does the promotional discount last?

A: The promotional discounts end on June 30, 2012.

### Installing the WAR file into your own web server

Stash comes bundled with a pre-configured Apache Tomcat installation which we highly recommend you use.

If you do require using your own web server, you can find the `atlassian-stash.war` file in the root directory of the Stash download, referred to as `<Stash installation directory>`.

There are things you will need to ensure when configuring your web server so that Stash will work correctly.

#### File encoding

The Stash war needs to be run with the `file.encoding` property set to UTF-8 in the JVM.

In the bundled Tomcat, this is configured in `<Stash installation directory>/bin/setenv.sh` (or `set env.bat` for Windows), by adding `"-Dfile.encoding=UTF-8"` to `JAVA_OPTS`. See [System settings](#) for more details.

### 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](#)
- [How to report a security issue](#)
- [New features policy](#)
- [Patch policy](#)
- [Security advisory publishing policy](#)
- [Security patch policy](#)
- [Severity levels for security issues](#)

To request support from Atlassian, please raise a support issue in our online support system. To do this, visit [support.atlassian.com](http://support.atlassian.com), log in (creating an account if need be) and create an issue under Stash. Our friendly support engineers will get right back to you with an answer.

### Bug fixing policy

**Summary**

- Atlassian Support will help with workarounds and bug reporting.
- Critical bugs will generally be fixed in the next maintenance release.
- Non critical bugs will be scheduled according to a variety of considerations.
Raising a Bug Report

Atlassian Support is eager and happy to help verify bugs — we take pride in it! Please open a support request in our support system providing as much information as possible about how to replicate the problem you are experiencing. We will replicate the bug to verify, then lodge the report for you. We'll also try to construct workarounds if they're possible.

Customers and plugin developers are also welcome to open bug reports on our issue tracking systems directly. Use http://jira.atlassian.com for the stand-alone products and http://studio.atlassian.com for JIRA Studio and Atlassian OnDemand.

When raising a new bug, you should rate the priority of a bug according to our JIRA usage guidelines. Customers should watch a filed bug in order to receive e-mail notification when a "Fix Version" is scheduled for release.

How Atlassian Approaches Bug Fixing

Maintenance (bug fix) releases come out more frequently than major releases and attempt to target the most critical bugs affecting our customers. The notation for a maintenance release is the final number in the version (ie the 1 in 3.0.1).

If a bug is critical (production application down or major malfunction causing business revenue loss or high numbers of staff unable to perform their normal functions) then it will be fixed in the next maintenance release provided that:

- The fix is technically feasible (i.e. it doesn't require a major architectural change).
- It does not impact the quality or integrity of a product.

For non-critical bugs, the developer assigned to fixing bugs prioritises the non-critical bug according to these factors:

- How many of our supported configurations are affected by the problem.
- Whether there is an effective workaround or patch.
- How difficult the issue is to fix.
- Whether many bugs in one area can be fixed at one time.

The developers responsible for bug fixing also monitor comments on existing bugs and new bugs submitted in JIRA, so you can provide feedback in this way. We give high priority consideration to security issues.

When considering the priority of a non-critical bug we try to determine a 'value' score for a bug which takes into account the severity of the bug from the customer's perspective, how prevalent the bug is and whether roadmap features may render the bug obsolete. We combine this with a complexity score (i.e. how difficult the bug is). These two dimensions are used when developers self serve from the bug pile.

Further reading

See Atlassian Support Offerings for more support-related information.

How to report a security issue

Finding and Reporting a Security Vulnerability

If you find a security bug in the product, please open an issue on http://jira.atlassian.com in the relevant project.

- Set the priority of the bug to ‘Blocker’.
- Provide as much information on reproducing the bug as possible.
- Set the security level of the bug to ‘Developer and Reporters only’.

All communication about the vulnerability should be performed through JIRA, so that Atlassian can keep track of
If you discover a security vulnerability, please attempt to create a test case that proves this vulnerability locally before opening either a bug or a support issue. When creating an issue, please include information on how the vulnerability can be reproduced; see our Bug Fixing Policy for general bug reporting guidelines. We will prioritise fixing the reported vulnerability if your report has information on how the vulnerability can be exploited.

Further reading

See Atlassian Support Offerings for more support-related information.

New features policy

Summary

- We do not publish roadmaps.
- Product Managers review our most popular voted issues on a regular basis.
- We schedule features based on a variety of factors.
- Our Atlassian Bug Fixing Policy is distinct from our Feature Request process.
- Atlassian provides consistent updates on the top 20 feature/improvement requests (in our issue tracker systems).

How to Track what Features are Being Implemented

When a new feature or improvement is scheduled, the ‘fix-for’ version will be indicated in the JIRA issue. This happens for the upcoming release only. We maintain roadmaps for more distant releases internally, but because these roadmaps are often pre-empted by changing customer demands, we do not publish them.

How Atlassian Chooses What to Implement

In every major release we aim to implement highly requested features, but it is not the only determining factor. Other factors include:

- Direct feedback from face to face meetings with customers, and through our support and sales channels.
- Availability of staff to implement features.
- Impact of the proposed changes on the application and its underlying architecture.
- How well defined the requested feature is (some issues gain in popularity rapidly, allowing little time to plan their implementation).
- Our long-term strategic vision for the product.

How to Contribute to Feature Development

Influencing Atlassian’s release cycle

We encourage our customers to vote on feature requests in JIRA. The current tally of votes is available online in our issue tracking systems, http://jira.atlassian.com and http://studio.atlassian.com. Find out if your improvement request already exists. If it does, please vote for it. If you do not find it, create a new feature or improvement request online.

Extending Atlassian Products

Atlassian products have powerful and flexible extension APIs. If you would like to see a particular feature implemented, it may be possible to develop the feature as a plugin. Documentation regarding the plugin APIs is available. Advice on extending either product may be available on the user mailing-lists, or at Atlassian Answers.

If you require significant customisations, you may wish to get in touch with our partners. They specialise in extending Atlassian products and can do this work for you. If you are interested, please contact us.
Further reading

See Atlassian Support Offerings for more support-related information.

Patch policy

Patch Policy

Atlassian will only provide software patches in extremely unusual circumstances. If a problem has been fixed in a newer release of the product, Atlassian will request that you upgrade your instance to fix the issue. If it is deemed necessary to provide a patch, a patch will be provided for the current release and the last maintenance release of the last major version (e.g. JIRA 4.2.4) only.

Patches are issued under the following conditions:

- The bug is critical (production application down or major malfunction causing business revenue loss or high numbers of staff unable to perform their normal functions).
- A patch is technically feasible (i.e., it doesn't require a major architectural change)
  OR
- The issue is a security issue, and falls under our Security Patch Policy.

Atlassian does not provide patches for non-critical bugs.

Provided that a patch does not impact the quality or integrity of a product, Atlassian will ensure that patches supplied to customers are added to the next maintenance release. Customers should watch a filed bug in order to receive e-mail notification when a "Fix Version" is scheduled for release.

Patches are generally attached to the relevant http://jira.atlassian.com issue.

Further reading

See Atlassian Support Offerings for more support-related information.

Security advisory publishing policy

Publication of Security Advisories

When a security vulnerability in an Atlassian product is discovered and resolved, Atlassian will inform customers through the following mechanisms:

- We will post a security advisory in the latest documentation of the affected product at the same time as releasing a fix for the vulnerability. This applies to all security advisories, including severity levels of critical, high, medium and low.
- We will send a copy of all security advisories to the 'Technical Alerts' mailing list for the product concerned.
  Note: To manage your email subscriptions and ensure you are on this list, please go to my.atlassian.com and click 'Email Prefs' near the top right of the page.
- If the person who reported the vulnerability wants to publish an advisory through some other agency, such as CERT, we will assist in the production of that advisory and link to it from our own.

Early warning of critical security vulnerabilities:

- If the vulnerability is rated critical (see our criteria for setting severity levels) we may send an early warning to the 'Technical Alerts' mailing list approximately one week before releasing the fix. This early warning is in addition to the security advisory itself, described above.
- However, if the vulnerability is publicly known or being exploited, we will release the security advisory and patches as soon as possible, potentially without early warning.
Further reading

See Atlassian Support Offerings for more support-related information.

Security patch policy

Product Security Patch Policy

Atlassian makes it a priority to ensure that customers’ systems cannot be compromised by exploiting vulnerabilities in Atlassian products.

Scope

This page describes when and how we release security patches and security upgrades for our products. It does not describe the whole of disclosure process that we follow. It also excludes JIRA Studio, since JIRA Studio will always be patched by Atlassian without additional notifications.

Critical vulnerabilities

When a Critical security vulnerability is discovered by Atlassian or reported by a third party, Atlassian will do all of the following:

- Issue a new, fixed release for the current version of the affected product as soon as possible, usually in a few days.
- Issue a binary patch for the current release.
- Issue a binary patch for the latest maintenance release of the previous version of the product.
- Patches for older versions or releases normally will not be issued.

Patches will be attached to the relevant JIRA issue. You can use these patches as a "stop-gap" measure until you upgrade your installation in order to fully fix the vulnerability.

Non-critical vulnerabilities

When a security issue of a High, Medium or Low severity is discovered, Atlassian will do all of the following:

- Include the fix into the next scheduled release, both for the current and previous maintenance versions.
- Where practical, provide new versions of plugins or other components of the product that can be upgraded independently.

You should upgrade your installation in order to fix the vulnerability.

Other information

Severity level of vulnerabilities is calculated based on Severity Levels for Security Issues.

Visit our general Atlassian Patch Policy as well.

Examples

**Example 1:** A critical severity vulnerability is found in a (hypothetical current release) JIRA 5.3.2. The last bugfix release in 5.2.x branch was 5.2.3. In this case, a patch will be created for 5.3.2 and 5.2.3. In addition, new bugfix releases, 5.3.3 and 5.2.4, which are free from this vulnerability, will be created in a few days.

**Example 2:** A high or medium severity vulnerability is found in the same release as in the previous example. The fix will be included into the currently scheduled releases 5.3.3 and 5.2.4. Release schedule will not be brought forward and no patches will be issued. If the vulnerability is in a plugin module, then a plugin upgrade package may still be supplied.
Severity levels for security issues

Severity Levels

Atlassian security advisories include a severity level. This severity level is based on our self-calculated CVSS score for each specific vulnerability. CVSS is an industry standard vulnerability metric. You can learn more about CVSS at FIRST.org web site.

CVSS scores are mapped into the following severity ratings:

- Critical
- High
- Medium
- Low

An approximate mapping guideline is as follows:

<table>
<thead>
<tr>
<th>CVSS score range</th>
<th>Severity in advisory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 2.9</td>
<td>Low</td>
</tr>
<tr>
<td>3 – 5.9</td>
<td>Medium</td>
</tr>
<tr>
<td>6.0 – 7.9</td>
<td>High</td>
</tr>
<tr>
<td>8.0 – 10.0</td>
<td>Critical</td>
</tr>
</tbody>
</table>

Below is a summary of the factors which illustrate types of vulnerabilities usually resulting in a specific severity level. Please keep in mind that this rating does not take into account details of your installation.

Severity Level: Critical

Vulnerabilities that score in the critical range usually have the following characteristics:

- Exploitation of the vulnerability results in root-level compromise of servers or infrastructure devices.
- The information required in order to exploit the vulnerability, such as example code, is widely available to attackers.
- Exploitation is usually straightforward, in the sense that the attacker does not need any special authentication credentials or knowledge about individual victims, and does not need to persuade a target user, for example via social engineering, into performing any special functions.

For critical vulnerabilities, is advised that you patch or upgrade as soon as possible, unless you have other mitigating measures in place. For example, if your installation is not accessible from the Internet, this may be a mitigating factor.

Severity Level: High

Vulnerabilities that score in the high range usually have the following characteristics:

- The vulnerability is difficult to exploit.
- Exploitation does not result in elevated privileges.
- Exploitation does not result in a significant data loss.

Severity Level: Medium
Vulnerabilities that score in the medium range usually have the following characteristics:

- Denial of service vulnerabilities that are difficult to set up.
- Exploits that require an attacker to reside on the same local network as the victim.
- Vulnerabilities that affect only nonstandard configurations or obscure applications.
- Vulnerabilities that require the attacker to manipulate individual victims via social engineering tactics.
- Vulnerabilities where exploitation provides only very limited access.

**Severity Level: Low**

Vulnerabilities in the low range typically have very little impact on an organisation's business. Exploitation of such vulnerabilities usually requires local or physical system access.

**Further reading**

See [Atlassian Support Offerings](#) for more support-related information.