Documentation for Stash 3.9

Contents

Getting started . . . 5
Supported platforms . . . 7
Using Stash in the enterprise . . . 11
Installing and upgrading Git . . . 14
Configuring JIRA integration in the Setup Wizard . . . 17
Getting started with Git and Stash . . . 25
Importing code from an existing project . . . 30
STASHDEV-9597 - Supported platforms . . . 32
Using Stash . . . 36
Creating projects . . . 37
Creating repositories . . . 38
Creating personal repositories . . . 40
Using repository hooks . . . 41
Permanently authenticating with Git repositories . . . 43
Controlling access to code . . . 45
Using branch permissions . . . 47
Branch permission patterns . . . 49
Using repository permissions . . . 49
Using project permissions . . . 51
Allowing public access to code . . . 51
Using SSH keys to secure Git operations . . . 52
Creating SSH keys . . . 53
SSH user keys for personal use . . . 55
SSH access keys for system use . . . 58
Workflow strategies in Stash . . . 60
Using branches in Stash . . . 61
Automatic branch merging . . . 67
Using forks in Stash . . . 69
Keeping forks synchronized . . . 70
Using pull requests in Stash . . . 73
Checks for merging pull requests . . . 80
Notifications . . . 81
Markdown syntax guide . . . 82
Requesting add-ons . . . 87
Integrating Stash with Atlassian applications . . . 89
JIRA integration . . . 91
HipChat notifications . . . 94
Bamboo integration . . . 96
Administering Stash . . . 98
Users and groups . . . 99
External user directories . . . 106
Connecting Stash to an existing LDAP directory . . . 108
Connecting Stash to JIRA for user management . . . 115
Delegating Stash authentication to an LDAP directory . . . 117
Connecting Stash to Crowd . . . 122
Global permissions . . . 124
Setting up your mail server . . . 126
Linking Stash with JIRA . . . 129
JIRA FishEye-Stash Plugin compatibility . . . 131
Using custom JIRA issue keys with Stash . . . 133
Connecting Stash to an external database . . . 133
Connecting Stash to MySQL . . . 135
Connecting Stash to Oracle . . . 138
Connecting Stash to PostgreSQL . . . 141
Connecting Stash to SQL Server . . . 144
Transitioning from jTDS to Microsoft's JDBC driver . . . 148

Migrating Stash to another server . . . 149
Specifying the base URL for Stash . . . 151
Configuring the application navigator . . . 151
Managing add-ons . . . 152
POST service webhook for Stash . . . 153
Audit logging in Stash . . . 157
Audit events in Stash . . . 157
Advanced actions . . . 162
Running the Stash installer . . . 162
Automated setup for Stash . . . 165
Starting and stopping Stash . . . 167
Install Stash from an archive file . . . 168
Running Stash as a Linux service . . . 174
Running Stash as a Windows service . . . 180
Stash config properties . . . 182
Proxying and securing Stash . . . 208
Securing Stash with Tomcat using SSL . . . 211
Integrating Stash with Apache HTTP Server . . . 218
Securing Stash with Apache using SSL . . . 222
Securing Stash behind nginx using SSL . . . 226
Enabling SSH access to Git repositories in Stash . . . 228
Setting up SSH port forwarding . . . 231
Using diff transcoding in Stash . . . 234
Changing the port that Stash listens on . . . 235
Moving Stash to a different context path . . . 236
Running Stash with a dedicated user . . . 237
Stash debug logging . . . 238
Data recovery and backups . . . 240
Using the Stash Backup Client . . . 242
Using Stash DIY Backup . . . 248
Lockout recovery process . . . 266
Scaling Stash . . . 266
Overview of Stash . . . 270
Scaling Stash for Continuous Integration performance . . . 276
Stash production server data . . . 278
High availability for Stash . . . 284
Clustering with Stash Data Center . . . 292
Installing Stash Data Center . . . 294
Adding cluster nodes to Stash Data Center . . . 305
Enabling JMX counters for performance monitoring . . . 307
Getting started with Stash and AWS . . . 313
Quick Start with Stash and AWS . . . 315
Launching Stash in AWS manually . . . 316
Administering Stash in AWS . . . 320
Recommendations for running Stash Server in AWS . . . 325
Securing Stash in AWS . . . 332
Using Stash DIY Backup in AWS . . . 334
Releases . . . 340
Stash upgrade guide . . . 346
End of support announcements for Stash . . . 365
Stash 3.9 release notes . . . 366
Stash 3.8 release notes . . . 367
Stash 3.7 release notes . . . 370
Stash 3.6 release notes . . . 372
Stash 3.5 release notes . . . 374
Stash 3.4 release notes . . . 377
Stash 3.3 release notes . . . 381
Stash 3.2 release notes . . . 384
Stash 3.1 release notes . . . 388
Stash 3.0 release notes . . . 392
Stash 2.12 release notes . . . 396
Stash 2.11 release notes . . . 402

- Stash 2.10 release notes . . . 405
- Stash 2.9 release notes . . . 407
- Stash 2.8 release notes . . . 412
- Stash 2.7 release notes . . . 418
- Stash 2.6 release notes . . . 422
- Stash 2.5 release notes . . . 426
- Stash 2.4 release notes . . . 429
- Stash security advisories . . . 434
  - Stash security advisory 2012-09-04 . . . 434
  - Stash security advisory 2014-02-26 . . . 435

Git resources . . . 436
- Basic Git commands . . . 437

Stash FAQ . . . 439
- How do I change the external database password . . . 443
- Stash home directory . . . 444
- Raising a request with Atlassian Support . . . 446
- Support policies . . . 448
  - Bug fixing policy . . . 448
  - New features policy . . . 449
  - Security Bugfix Policy . . . 450
- Building Stash from source . . . 450
- Contributing to the Stash documentation . . . 450
- Collecting analytics for Stash . . . 451

Getting started

To install and start using Stash, read on!

If you're upgrading Stash

... read the Stash upgrade guide instead.

For production installs

... we highly recommend that you first read Using Stash in the enterprise.

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

1. Install Git and Perl

Stash requires Git on the machine that will run Stash. If you need to check, install, or upgrade Git on the Stash server machine, see Installing and upgrading Git.

Check that you have all the other system requirements, including Perl, to avoid any trouble.

2. Install Stash

Download latest version

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

See Running the Stash installer for details.

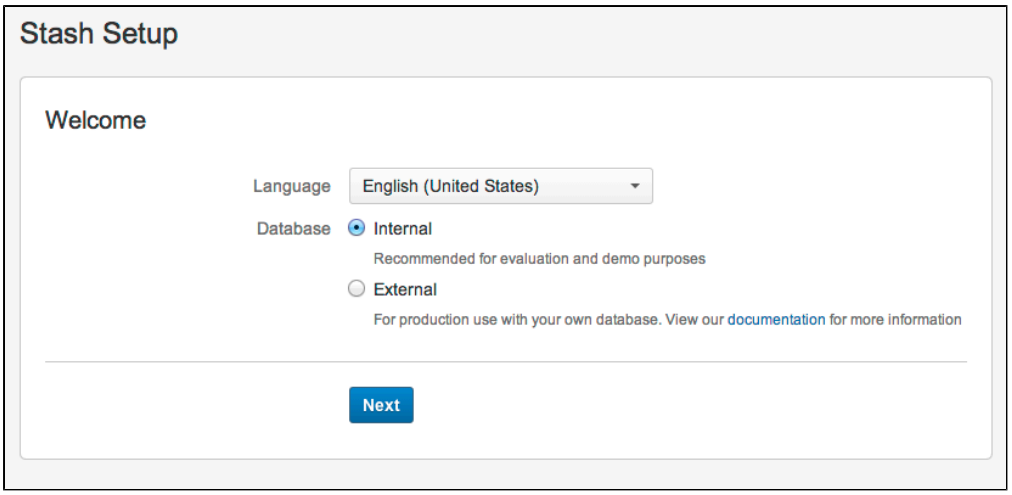
You may be interested in these alternative provisioning approaches:

- Installing Stash from an archive file
- Docker repository (evaluation only)
- Chef recipe for Stash
- Puppet/Vagrant How-To

3. Set up Stash

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

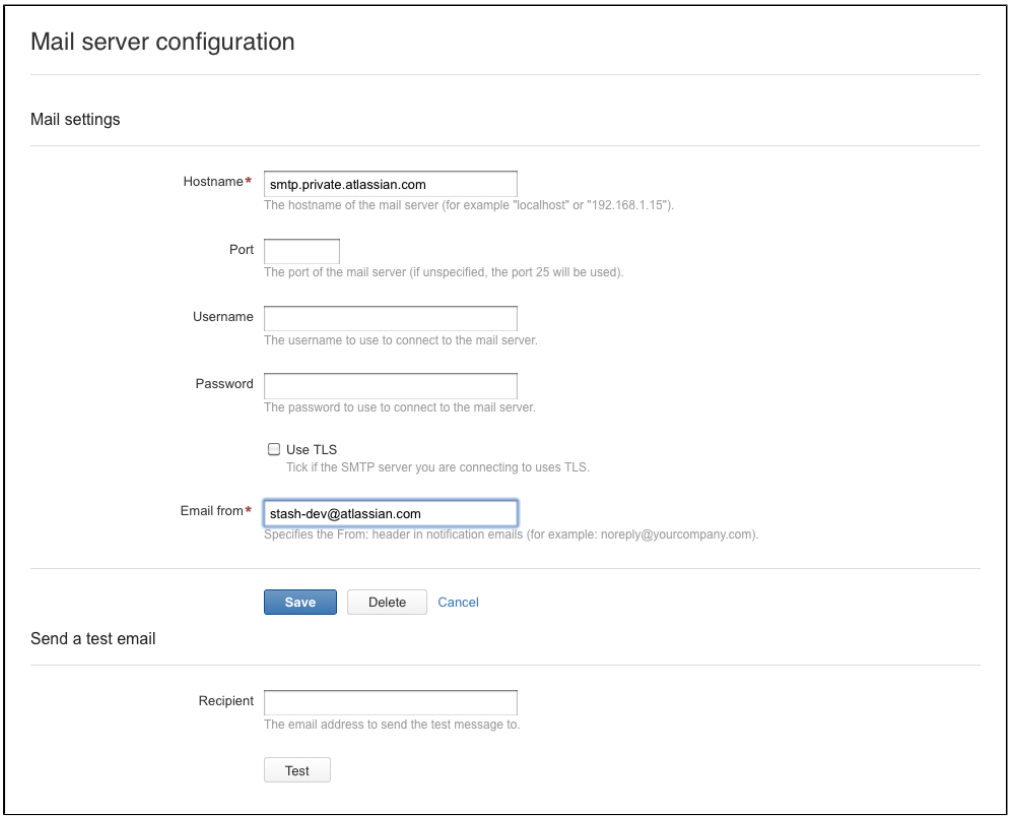
- Specify the default language for Stash.
- Connect Stash to an external database (the internal HSQL database is great for evaluating Stash, but is not recommended for production installations). You'll need to have created the external database before running the Setup Wizard. For MySQL, you'll need to have installed the JDBC driver too.
- Enter your Stash license key.
- Set the base URL for Stash.
- Set up an administrator account.
- Integrate Stash with JIRA.
- Log in to Stash.



If you are intending to use Stash for a production installation, see also Using Stash in the enterprise.

4. Set up the mail server

Configuring the Stash email server allows users to receive a link from Stash that lets them generate their own passwords. See Setting up your mail server.



5. Get working with Stash

Work with projects

Stash manages related Git repositories as projects. Find out how to set up projects and give your teams access to those.

If you have existing projects that you want to manage in Stash, then you'll want to read Importing code from

an existing project.

Integrate Stash with other Atlassian applications

See Integrating Stash with Atlassian applications for an overview of what is possible.

As a first step, see JIRA integration for information about using Stash with JIRA.

If you want to see results from your continuous integration or build server in Stash, see Bamboo integration.

Use Stash in your enterprise

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

- Using Stash in the enterprise
- High availability for Stash
- Scaling Stash
- Scaling Stash for Continuous Integration performance
- Stash Data Center
- Stash production server data

Use Git

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

- Git Tutorials and Training
- Basic Git commands
- Permanently authenticating with Git repositories
- Using SSH keys to secure Git operations
- Git resources

Supported platforms

This page lists the supported platforms for **Stash 3.9.x**.

See Integrating Stash with Atlassian applications for information about supported versions of JIRA.

See End of support announcements for Stash for upcoming changes to platforms supported by Stash.

Key: ✓ = Supported ⚠ = Deprecated ✗ = Not Supported

Hardware		
CPU	Evaluation: 1 core Production: 2+ cores	• If you are evaluating Stash, we recommend that you use a server with **at least 2GB of memory**. • As well as the memory *allocated* for Tomcat (768MB is the default configuration and suitable for most uses), additional memory and CPU capacity is required to support Git operations. • For Amazon Web Services (AWS) instance types and sizes, see Recommendations for running Stash Server in AWS. • ✗ Stash Data Center is not supported in AWS at this time. • The hardware requirements for a full production deployment depend on the number and frequency of Git operations and the number of active users. See Scaling Stash for further discussion and for details of how memory is allocated for Stash and Git.
Memory	2GB+	
Operating systems		
Apple Mac OS X	✓ Evaluation ✗ Production	• Stash is a pure Java application and should run on any platform, provided all the Java requirements are satisfied. • In production environments Stash should be run from a dedicated

Linux	✅	user account. • ❌ Apple Mac OS X is not supported for production deployment. • ❌ Microsoft Windows is not supported for 500+ Enterprise tiers.
Microsoft Windows	✅ < 500 users ❌ 500+ Enterprise	
Java		
Oracle Java	✅ 1.8 ⚠️ 1.7 ❌ 1.6	• ⚠️ Support for Java 7 is deprecated, and will be removed in Stash 4.0. See End of support announcements for Stash. • Java 8 is supported, as of Stash 3.0. • Stash only requires the Java JRE, not the JDK. • For Oracle Java, we recommend using Server JRE 7, which you can download from the Oracle website. • For OpenJDK, download and install instructions for Linux flavors are at http://openjdk.java.net/install/. • Note that the Stash installer will install a supported version of the Java JRE, which is only available to Stash, if necessary. See Running the Stash installer. • ❌ OpenJDK 1.8u25 and later versions are not supported until https://bugzilla.redhat.com/show_bug.cgi?id=1167153 is fixed. • ❌ Support for Java 6 was *removed* in Stash 3.0, as previously announced. Pre-installed Java on some AWS EC2 Linux instances might be installed with a subset of features. See SSH server fails to start on AWS EC2 instance for more information.
OpenJDK	✅ 1.8u0–1.8u20 ❌ 1.8u25+ ⚠️ 1.7 ❌ 1.6	
Databases		
HSQLDB	✅ Evaluation ❌ Stash Data Center	• Please see connecting Stash to an external database. • HSQLDB is bundled with Stash, and is only intended for evaluation use. • ❌ HSQLDB is not supported in Stash Data Center.
Microsoft SQL Server / Microsoft SQL Server Express	✅ 2014 ✅ 2012 ✅ 2008 R2 ✅ 2008 ✅ 2005	

MySQL	Stash Server: ✅ 5.6.16+ ✅ 5.5.x ✅ 5.1.x ❌ 5.7+ ❌ 5.6.0 – 5.6.15 ✅ MariaDB 5.5 ❌ Stash Data Center	MySQL, while supported by Stash Server, is currently ***not*** recommended, especially for larger instances, due to inherent performance and deadlock issues that occur in this database engine under heavy load. Affected systems may experience slow response times, deadlock errors and in extreme cases errors due to running out of database connections. These issues are intrinsic to MySQL (no other database engine supported by Stash shares this behavior) and are due to the way MySQL performs row-level locking in transactions. See http://dev.mysql.com/doc/refman/5.0/en/innodb-deadlocks.html for some general information on this. Stash does its best to work around the MySQL behavior - see issues STASH-4517, STASH-4701 and others, for example. But under very heavy load you will generally get better performance with any of the other database engines supported by Stash (such as PostgreSQL, which is also freely available) than you will with MySQL. • MariaDB 5.5, but no other version, is supported as of Stash 3.6. See Connecting Stash to MySQL for more information. • ❌ MySQL 5.6.15 and earlier: Note that Stash *does not support* versions of MySQL 5.6 earlier than 5.6.16 at all, because of bugs in its query optimizer (#68424, #69005). See Connecting Stash to MySQL for more information. • ❌ MySQL 5.7+ is not supported. • ❌ MySQL (any version) is not supported in Stash Data Center.
Oracle	✅ 12c ✅ 11g	
PostgreSQL	✅ 9.0, 9.1, 9.2, 9.3, 9.4 ✅ 8.2, 8.3, 8.4	
Web browsers		
Chrome	✅ Latest stable version supported	
Firefox	✅ Latest stable version supported	
Internet Explorer	✅ 11 ⚠️ 10 ⚠️ 9 ❌ 8	• ⚠️ Support for Internet Explorer 10 is deprecated and will be removed in Stash 4.0. See End of support announcements for Stash . • ⚠️ Support for Internet Explorer 9 is deprecated and will be removed in Stash 3.10. See End of support announcements for Stash . • ❌ Support for Internet Explorer 8 was *removed* in Stash 3.0, as previously announced.
Safari	✅ Latest stable version supported	
DVCS		
Git – server		The table lists the versions of Git that have been tested against the **Sta**

Supported	Tested	
	Linux	*Windows*
⚠ 2.3.0	2.3.4	
⚠ 2.2.0+	2.2.2	
2.1.0+	2.1.4	
2.0.0–2.0.1 2.0.4+	2.0.5	
1.9.0+	1.9.5	1.9.5.1
1.8.0–1.8.4.2 1.8.4.4+	1.8.0.3 1.8.1.5 1.8.2.3 1.8.3.4 1.8.4.5 1.8.5.6	1.8.0 1.8.1.2 1.8.3 1.8.4 1.8.5.2
⚠ 1.7.6+	1.7.6.6 1.7.7.7 1.7.8.6 1.7.9.7 1.7.10.5 1.7.11.7 1.7.12.4	1.7.6 1.7.7.1 1.7.8 1.7.9 1.7.10 1.7.11

sh 3.9.x releases.

- In general, we recommend using the most recent version of Git on both the Stash server and clients, where possible, and subject to the following notes and exceptions.
- The version of Git installed on machines that interact with Stash must be compatible with the version of Git installed for use by the Stash server.
- For Git 1.9.0 and later we only test the highest bugfix release – all earlier bugfix releases in that series are also supported, unless specifically indicated otherwise below.
- ⚠ Support for versions of Git earlier than v1.8 on the server is deprecated and will be removed in Stash 4.0. See End of support announcements for Stash.
- ⚠ Git 1.8.3.x has some performance regressions which may cause problems in Stash with large repositories.
- ⚠ Git 2.2.x and higher have some performance issues when interacting with NFS. Hence, these versions are currently not supported for Stash Data Center or for Stash Server installations that use NFS mounts for the home directory (Details) .
- ✖ **Cygwin** : Cygwin Git *is not supported* for use on Windows servers, regardless of version.
- ✖ Git 1.7.1 is not supported.
- ✖ Git 1.8.4.3 is not supported due to a critical bug in how symbolic refs are handled which breaks pushing and pulling for repositories with pull requests. (Details)

  STASH-4101 - Clone and fetch fail with "protocol error: impossibly long line" CLOSED
- ✖ Git 2.0.2 and 2.0.3 are not supported due to a critical bug in `git diff-tree` which breaks Stash's commit page. (Details)

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

[Security vulnerability CVE-2014-9390] affects multiple Git versions. Stash itself is not affected, however end-users should update their *clients* to a patched maintenance version: v1.8.5.6, v1.9.5, v2.0.5, v2.1.4 and v2.2.1 or newer. For instructions see Installing and upgrading Git.

Git – client	✓ 1.6.6+	[Security vulnerability CVE-2014-9390] affects multiple Git versions. Stash itself is not affected, however end-users should update their *clients* to a patched maintenance version: v1.8.5.6, v1.9.5, v2.0.5, v2.1.4 and v2.2.1 or newer. For instructions see Installing and upgrading Git. Windows git client users are recommended to use Git-1.9.5-preview20150319 or higher which fixes an openssl vulnerability.
Additional tools		
Perl	✓ 5.8.8+	
Mail clients		
Apple Mail	✓ Apple Mail 4	
Gmail	✓ Latest	
iOS Devices	✓ iPhone, iPad	
Microsoft Outlook	✓ Express, 2007, 2010	
Outlook.com Hotmail Windows Live Mail	✓ Latest	

Notes:

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

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

Using Stash in the enterprise

This page...

... describes best practice for using Stash in enterprise environments.

If you're evaluating Stash...

... we suggest that you begin with Getting started, instead of this page.

See also...

... Stash Enterprise Resources for a comparison of Stash and Stash Data Center, our clustered Stash solution.

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

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

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

Stash Server

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

Stash Data Center

For larger enterprises that require HA and greater performance at scale, Stash Data Center uses a cluster of Stash nodes to provide Active/Active failover, and is the deployment option of choice.

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

On this page:

- Platform requirements for hosting Stash
- Performance considerations with Stash
- High availability with Stash
- Scalability
- Provisioning Stash
- Setting up Stash Server in a production environment
- Administering Stash in a production environment

Platform requirements for hosting Stash

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

See the Supported platforms page for details of the supported versions of Java, external databases, web browsers and Git.

See Installing Stash Data Center for detailed information about Stash Data Center requirements.

Performance considerations with Stash

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

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

Stash Server – see Stash production server data for data from the Stash production instance we run internally at Atlassian.

Stash Data Center – see Stash Data Center Performance for the results of our performance testing for clusters of different sizes.

High availability with Stash

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

Stash Server – see High availability for Stash for the background information you need to set up Stash Server in a highly available configuration.

Stash Data Center – see Failover for Stash Data Center for information about how Stash Data Center provides HA and almost instant failover.

Scalability

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

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

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

Stash Server – see Scaling Stash for information about how you can tune your Stash server to grow with your organisation's needs. See also Scaling Stash for Continuous Integration performance for information specific to Stash performance when CI tools poll Stash for changes.

Stash Data Center – see Adding cluster nodes to Stash Data Center for information about how you can rapidly provision extra capacity without downtime.

Provisioning Stash

Some possible approaches to provisioning Stash include:

- Running the Stash installer in either console or unattended mode
- Docker container image for Stash (currently only supported for evaluations)
- Chef recipe for Stash
- Puppet/Vagrant How-To

Setting up Stash Server in a production environment

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

Run Stash as a dedicated user

- For production environments Stash should be run from a dedicated user account with restricted privileges. See Running Stash with a dedicated user.

Install Stash as a service

- See Running the Stash installer.

Use an external database

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

Connect to your existing user directory

- Connect Stash to your existing user directory (for example Active Directory). See External user directories.

Secure the Stash home directory

- For production environments the Stash home directory should be secured against unauthorised access. See Stash home directory.

Secure Stash with HTTPS

- Access to Stash should be secured using HTTP over SSL, especially if your data is sensitive and Stash is exposed to the internet. See Securing Stash with HTTPS.

Enable SSH access to Git repositories

- Enable SSH access for your Stash users to Git repositories in Stash so that they can add their own SSH keys to Stash, and then use those SSH keys to secure Git operations between their computer and the Stash server. See Enabling SSH access to Git repositories in Stash.

Change the context path for Stash

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

Administering Stash in a production environment

Upgrading Stash

- For production environments we recommend that you test the Stash upgrade on a QA server before deploying to production. See the Stash upgrade guide.

Backups and recovery

- **We highly recommend** that you establish a data recovery plan that is aligned with your company's policies. See Data recovery and backups for information about tools and backup strategies for Stash.

Logging

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

Installing and upgrading Git

> [Security vulnerability CVE-2014-9390] If you are running a Git client older than 1.8.5.6, 1.9.5, 2.0.5, 2.1.4 or 2.2.1 (all released 18 Dec 2014), you should upgrade Git as soon as possible.
>
> Stash *servers* are not affected by this vulnerability.
>
> See also this Atlassian blog post: Securing your Git server against CVE-2014-9390.

This page describes how to:

- Check your version of Git
- Install or upgrade Git on Linux
- Install or upgrade Git on Mac OS X
- Install or upgrade Git on Windows

The information on this page applies to installing or upgrading Git on either your local machine, or on the Stash server.

Check your version of Git

The versions of Git supported by Stash are listed on Supported platforms.

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

For example:

```
git --version
git version 2.2.1
```

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

Install or upgrade Git on Linux

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

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

Alternative download options are:

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

Now check the Git version – you should see the new version of Git.

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

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

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

Install or upgrade Git on Mac OS X

Download the latest stable Git release from the Git website.

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

Alternatively, you can:

- Use the native Git bundled with OS X.
- Use Homebrew to download and install Git.

Now check the Git version – you should see the new version of Git.

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

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

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

Install or upgrade Git on Windows

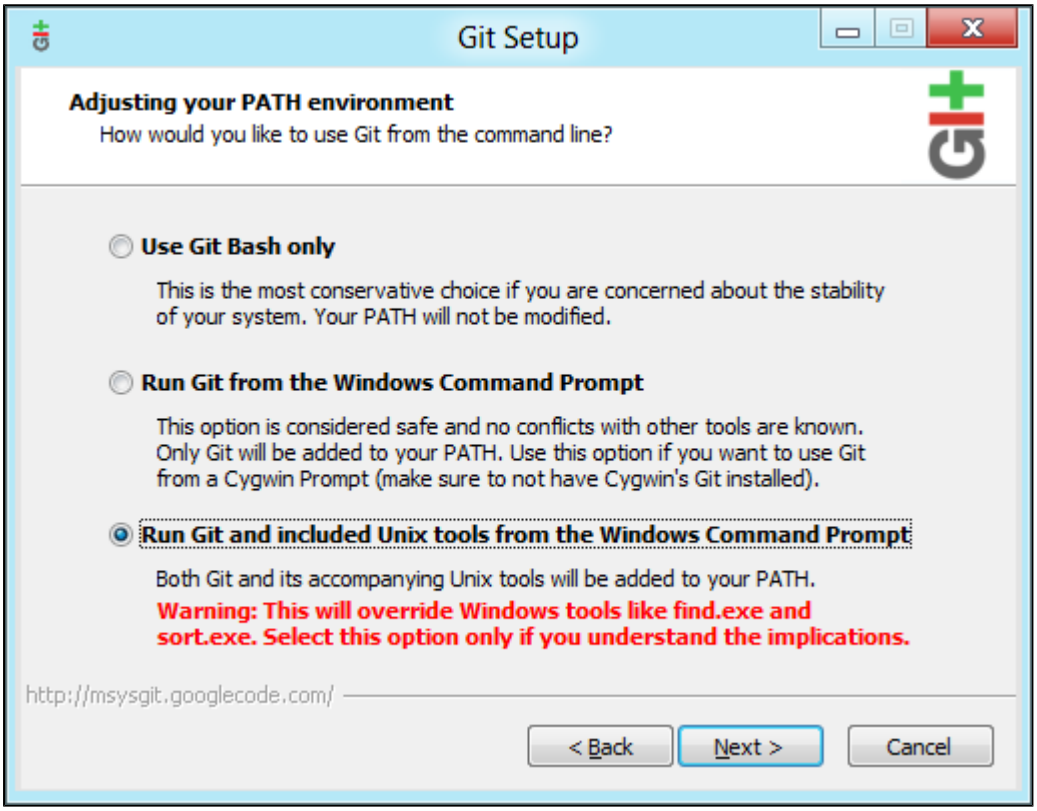
Download the latest stable Git release from the Git website.

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

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

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

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



Now, check the Git version – you should see the new version of Git.

⚠ msysGit is the *only supported distribution* when running Stash on Windows. Cygwin Git is *not supported* and has known issues.

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

Restart Stash if necessary

If you've been installing or upgrading Git for the Stash server, rather than for your local machine, you'll need to stop and restart Stash so that it will pick up the upgraded version of Git. See Starting and stopping Stash for details.

Configuring JIRA integration in the Setup Wizard

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

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

There are two aspects to integrating Stash with JIRA:

- Linking JIRA and Stash to enable the integration features. See JIRA integration.
- Delegating Stash user and group management to your JIRA server. See Connecting Stash to JIRA for user management.

On this page:

- Connecting to JIRA in the Setup Wizard
- Troubleshooting
- Notes

Connecting to JIRA in the Setup Wizard

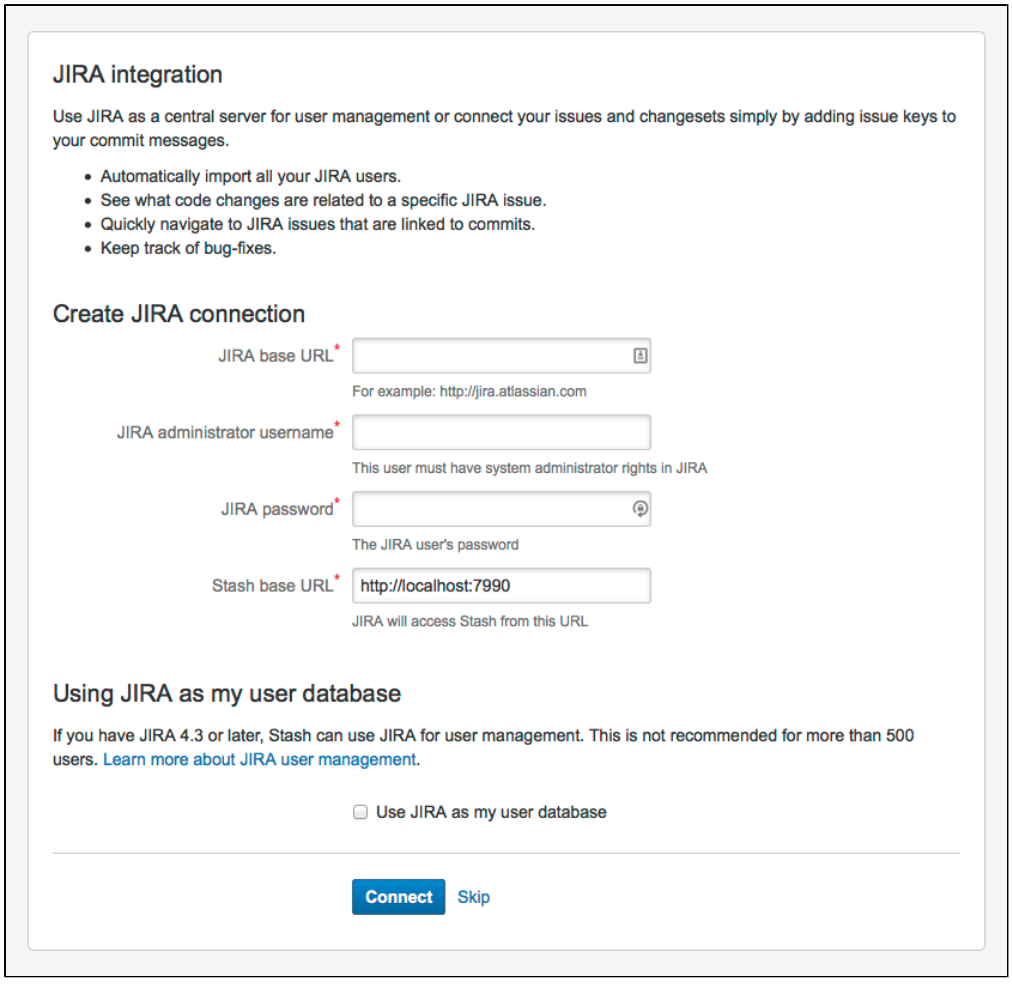
To configure JIRA integration while running the Stash Setup Wizard:

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:

JIRA base URL	The web address of your JIRA server. Examples are:
	`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.

Symptom	Cause	Solution

The setup wizard displays one of the following error messages: • Failed to create application link from JIRA server at <URL> to this <application> server at <URL>. • Failed to create application link from this <application> server at <URL> to JIRA server at <URL>. • Failed to authenticate application link from JIRA server at <URL> to this <application> server at <URL>. • Failed to authenticate application link from <application> server at <URL> to this JIRA server at <URL>.	The setup wizard failed to complete registration of the peer-to-peer application link with JIRA. JIRA integration is only partially configured.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
The setup wizard displays one of the following error messages: • Failed to register <application> configuration in JIRA for shared user management. Received invalid response from JIRA: <response> • Failed to register <application> configuration in JIRA for shared user management. Received: <response>	The setup wizard failed to complete registration of the client-server link with JIRA for user management. The peer-to-peer link was successfully created, but integration is only partially configured.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
The setup wizard displays the following error message: • Error setting Crowd authentication	The setup wizard successfully established the peer-to-peer link with JIRA, but could not persist the client-server link for user management in your `config.xml` file. This may be caused by a problem in your environment, such as a full disk.	Please investigate and fix the problem that prevented the application from saving the configuration file to disk. Then remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.
The setup wizard displays the following error message: • Error reloading Crowd authentication	The setup wizard has completed the integration of your application with JIRA, but is unable to start synchronizing the JIRA users with your application.	Restart your application. You should then be able to continue with the setup wizard. If this solution does not work, please contact Atlassian Support.
The setup wizard displays the following error message: • An error occurred: java.lang.IllegalStateException: Could not create the application in JIRA/Crowd (code: 500). Please refer to the logs for details.	The setup wizard has not completed the integration of your application with JIRA. The links are only partially configured. The problem occurred because there is already a user management configuration in JIRA for this <application> URL.	Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.

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

Solution 1: Removing a Partial Configuration – The Easiest Way

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

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

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 **Delete** next to the application link that you want to delete.
    d. A confirmation screen will appear. Click **Confirm** 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 configruration and then add the full integration manually. Please follow these steps:

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

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

      For example:

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

Or:

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

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

- Go to the following URL in your browser:

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

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

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

- The application links manifest will appear. Check the application ID in the `<id>` element.

c. In JIRA, click '**Delete**' next to the application that you want to remove.

6. Add the application link in JIRA again, so that you now have a two-way trusted link between JIRA and your application:
   a. Click **Add Application Link**. Step 1 of the link wizard will appear.
   b. Enter the **server URL** of the application that you want to link to (the 'remote application').
   c. Click **Next**.
   d. Enter the following information:
      - **Create a link back to this server** – Check to add a two-way link between the two applications.
      - **Username** and **Password** – Enter the credentials for a username that has administrator access to the remote application.
        Note: These credentials are only used to authenticate you to the remote application, so that Application Links can make the changes required for the new link. The credentials are not saved.
      - **Reciprocal Link URL** – The URL you give here will override the base URL specified in your remote application's administration console, for the purposes of the application links connection. Application Links will use this URL to access the remote application.
   e. Click **Next**.
   f. Enter the information required to configure authentication for your application link:
      - **The servers have the same set of users** – Check this box, because the users are the same in both applications.
      - **These servers fully trust each other** – Check this box, because you trust the code in both applications and are sure both applications will maintain the security of their private keys.
        For more information about configuring authentication, see Configuring Authentication for an Application Link.
   g. Click **Create**.
7. Configure a new connection for user management in JIRA:
   a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
      - In JIRA 4.3: Click '**Other Applications**' in the '**Users, Groups & Roles**' section of the JIRA administration screen.
      - In JIRA 4.4: Select **'Administration' > 'Users' > 'JIRA User Server'**.
   b. **Add** an application.
   c. Enter the **application name** and **password** that your application will use when accessing JIRA.

d. Enter the **IP address** or addresses of your application. Valid values are:
   - A full IP address, e.g. `192.168.10.12`.
   - A wildcard IP range, using CIDR notation, e.g. `192.168.10.1/16`. For more information, see the introduction to CIDR notation on Wikipedia and RFC 4632.
   - **Save** the new application.

8. Set up the JIRA user directory in the application.
   - For Confluence:
     a. Go to the **Confluence Administration Console**.
     b. Click '**User Directories**' in the left-hand panel.
     c. **Add** a directory and select type '**Atlassian JIRA**'.
     d. Enter the following information:
        - **Name** – Enter the name of your JIRA server.
        - **Server URL** – Enter web address of your JIRA server. Examples:

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

        - **Application name** and **Application password** – Enter the values that you defined for Confluence in the settings on JIRA.
     e. Save the directory settings.
     f. Define the **directory order** by clicking the blue up- and down-arrows next to each directory on the '**User Directories**' screen.
     For details see Connecting to Crowd or JIRA for User Management.
   - For FishEye/Crucible:
     a. Click **Authentication** (under 'Security Settings').
     b. Click **Setup JIRA/Crowd authentication**. Note, if LDAP authentication has already been set up, you will need to remove that before connecting to JIRA for user management.
     c. Make the following settings:

Authenticate against	Select **a JIRA instance**
Application name and **password**	Enter the values that you defined for your application in the settings on JIRA.

JIRA URL	The web address of your JIRA server. Examples:
	`http://www.example.com:8080/jira/`
	`http://jira.example.com`

Auto-add	Select **Create a FishEye user on successful login** so that your JIRA users will be automatically added as a FishEye user when they first log in.
Periodically synchronise users with JIRA	Select **Yes** to ensure that JIRA will synchronize all changes in the user information on a regular basis. Change the value for **Synchronise Period** if required.
When Synchronisation Happens	Select an option depending on whether you want to allow changes to user attributes from within FishEye.
Single Sign On	Select **Disabled**. SSO is not available when using JIRA for user management and if enabled will make the integration fail.

d. Click **Next** and select at least one user group to be synchronised from JIRA. If necessary, you could create a new group in JIRA, such as 'fisheye-users', and select this group here.
e. Click **Save**.

- For Stash:
  a. Go to the Stash administration area.
  b. Click **User Directories** in the left-hand panel.
  c. **Add** a directory and select type **Atlassian JIRA**.
  d. Enter the following information:
     - **Name** – Enter the name of your JIRA server.
     - **Server URL**– Enter web address of your JIRA server. Examples:

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

     - **Application name** and **Application password** – Enter the values that you defined for Stash in the settings on JIRA.
  e. Save the directory settings.
  f. Define the directory order by clicking the blue up- and down-arrows next to each directory on the 'User Directories' screen.
  For details see Connecting Stash to JIRA for user management.

Notes

When you connect to JIRA 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.

Getting started with Git and Stash

Atlassian Stash is the Git repository management solution for enterprise teams. It allows everyone in your organisation to easily collaborate on 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 for interacting with them.

Assumptions

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

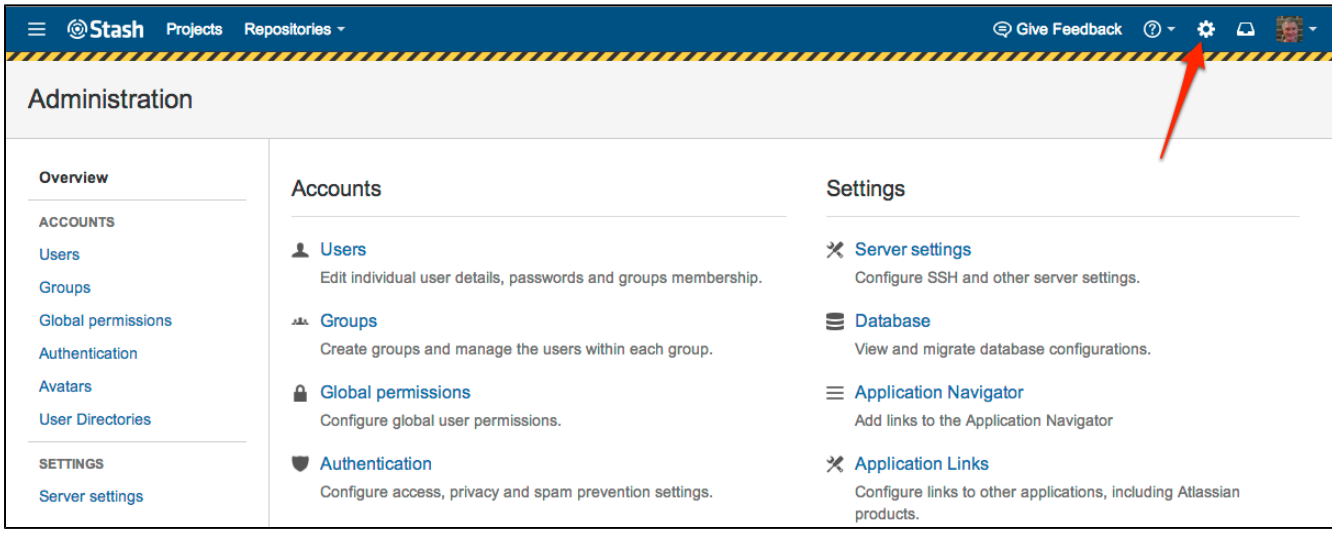
- You have Git version 1.7.6 or higher installed on your local computer.
- You are using a supported browser.
- You have Stash installed and running. If you haven't, see Getting started.

Please read Git resources or check out our Git tutorials for tips on getting started with Git.

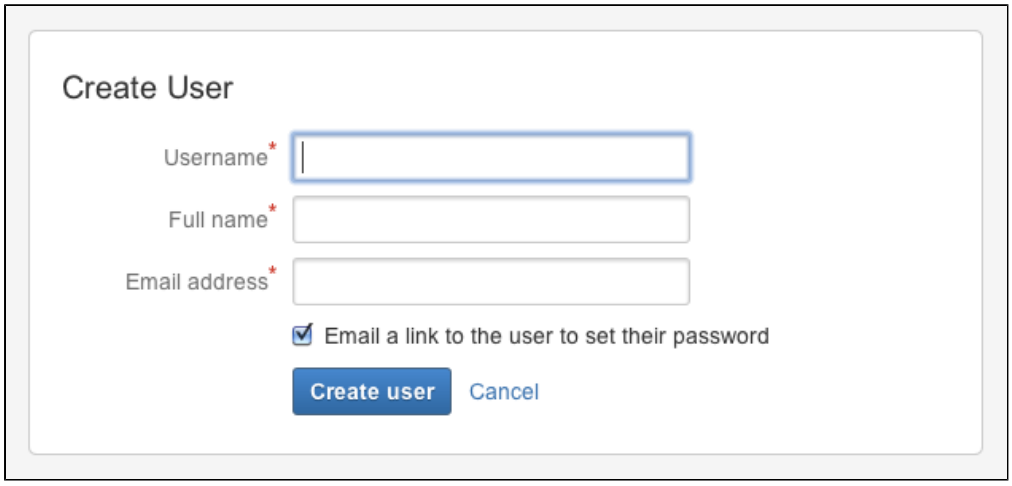
Add users to Stash and grant permissions

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

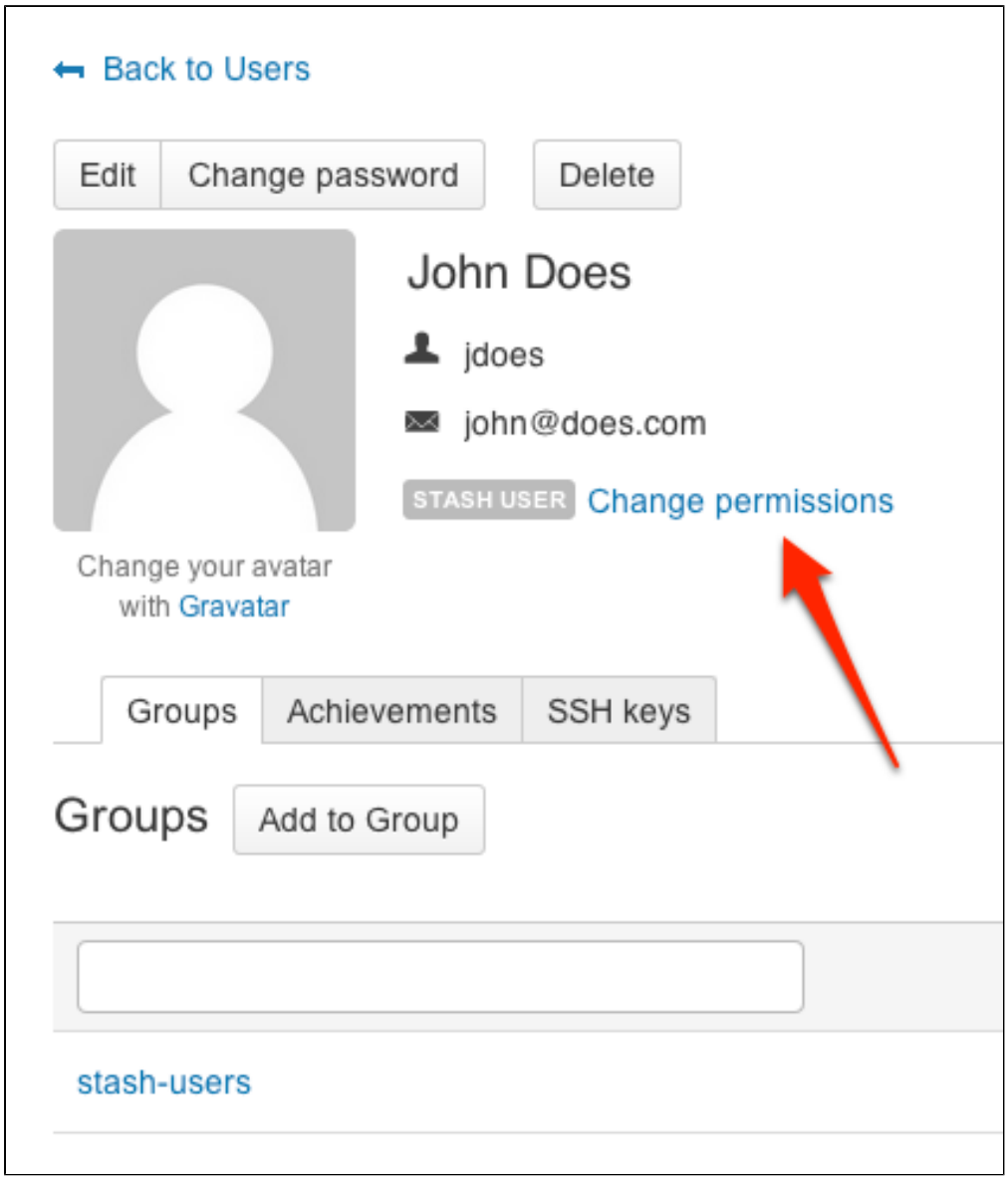
Go to the Stash administration area, by clicking the 'cog' menu in the header, and then click **Users** (under 'Accounts'):



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 authentication:

- **System Administrator** — can access all the configuration settings of the Stash instance.
- **Administrator** — same as System Admins, but they can't modify file paths or the Stash server settings.

- **Project Creator** — can create, modify and delete projects.
- **Stash User** — active users who can access Stash.

See Users and groups for more information about authentication.

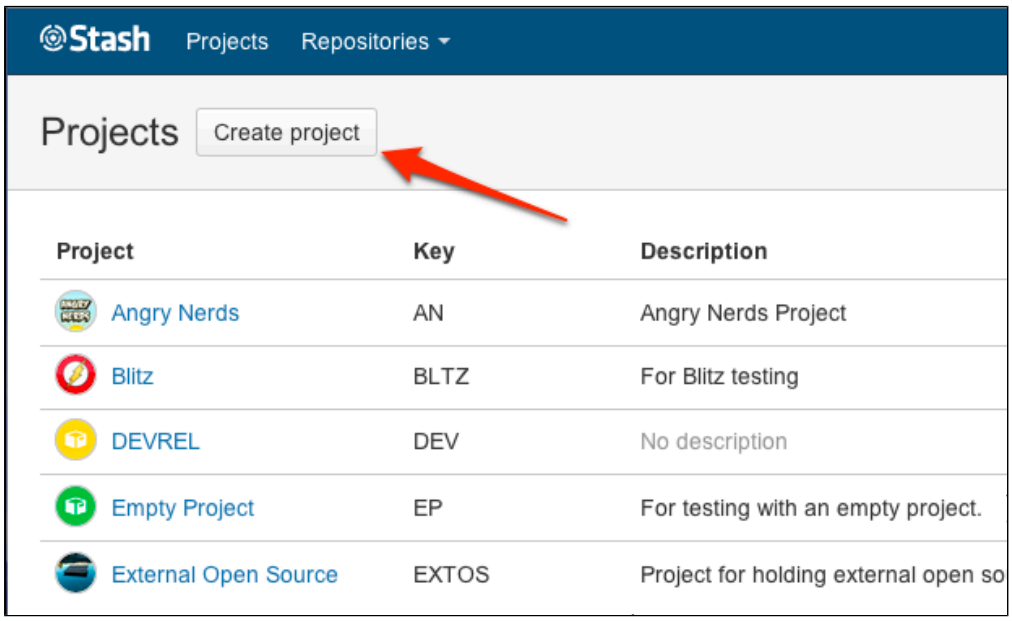
See External user directories if you have existing user identities you wish to use with Stash.

Create your first project and share it with collaborators

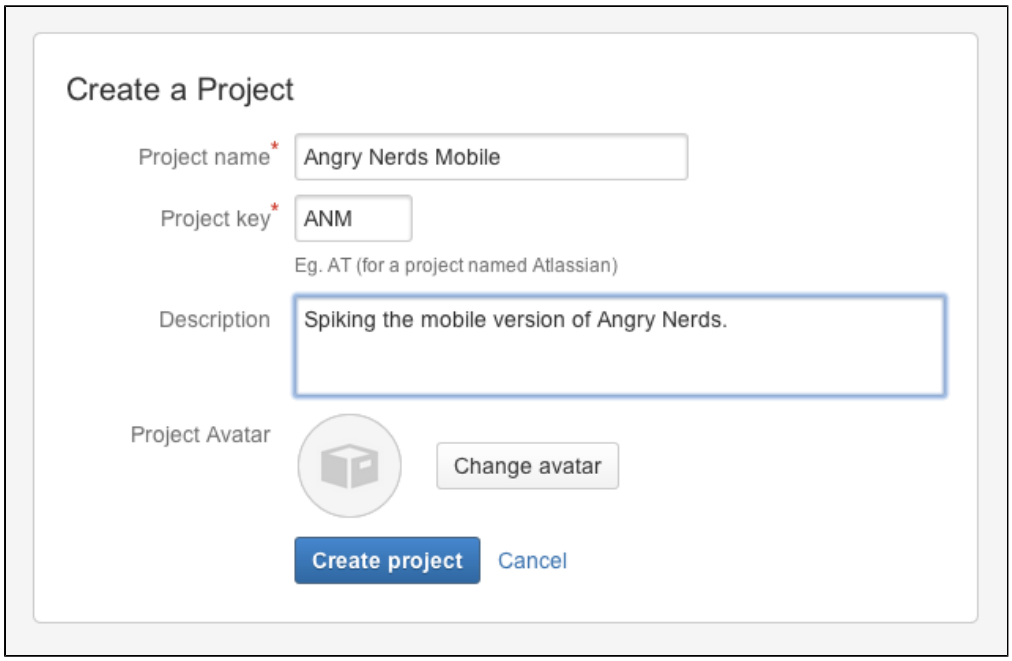
Creating your project

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

Simply go to 'Projects' and click **Create project**. (Initially, you won't see as many projects as shown in this screenshot.)



Complete the form and submit it to create your new project:

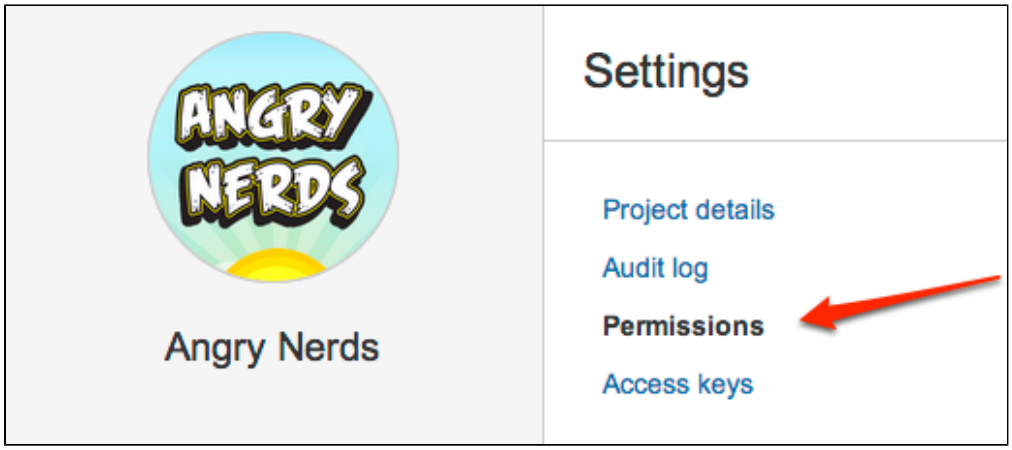


See Creating projects for more information.

Opening up project access to others

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

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



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

There are 3 levels of project access:

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

See Using project permissions for more information.

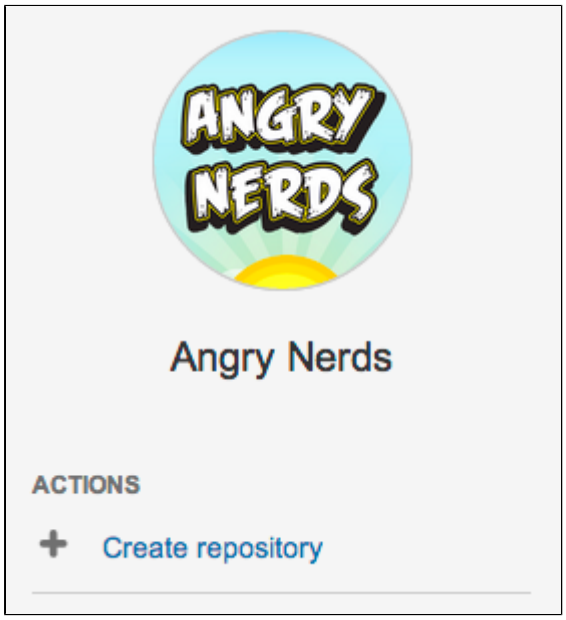
Create a repository and get your code into Stash

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. If you already have a Git project you'd like to use, see Importing code from an existing project.

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



Once submitted you will be taken directly to your repository homepage. As there is no content in your repository yet, you'll see some instructions to help you push code to your repository.

See Creating repositories for more information.

A simple clone and push

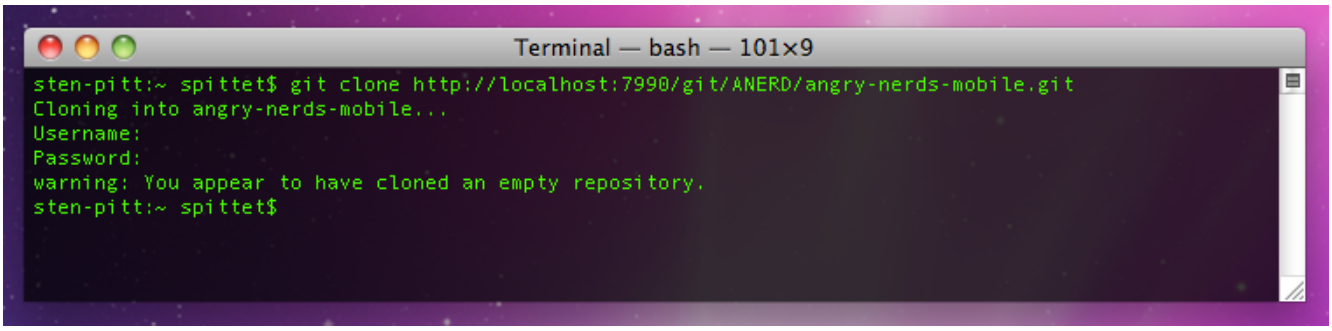
This section describes how to clone the repository you just created and then push a commit back to it. You can see the clone URL to use at the top right of the screen. SSH access may be available.

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

```
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

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

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.

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

Check out our Git tutorials and training for more information, and have a look at this list of basic Git commands t hat you will probably use often.

Importing code from an existing project

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

On this page:

- Import an existing, unversioned code project into Stash
- Import an existing Git project into Stash
- Mirror an existing Git repository

Import an existing, unversioned code project into Stash

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

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

1. Locally, change to the root directory of your existing source.
2. Initialise the project by running the following commands in the terminal:

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

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

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

6. Done! Your repository is now available in Stash.

Import an existing Git project into Stash

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

1. Check out the repository from your existing Git host. Use the *--bare* parameter:

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

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

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

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

```
git push --all stash
git push --tags stash
```

6. Remove your temporary local repository:

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

Mirror an existing Git repository

You can mirror an existing repository into a repository hosted in Stash.

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

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

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

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

5. Then push all branches and tags to Stash:

```
git push --all stash
git push --tags stash
```

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

STASHDEV-9597 - Supported platforms

This page lists the supported platforms for **Stash 3.9.x**.

See Integrating Stash with Atlassian applications for information about supported versions of JIRA.

See End of support announcements for Stash for upcoming changes to platforms supported by Stash.

Key: ✓ = Supported ⚠ = Deprecated ✗ = Not Supported

Hardware		
CPU	Evaluation: 1 core Production: 2+ cores	• If you are evaluating Stash, we recommend that you use a server with **at least 2GB of memory**. • As well as the memory *allocated* for Tomcat (768MB is the default configuration and suitable for most uses), additional memory and CPU capacity is required to support Git operations. • For Amazon Web Services (AWS) instance types and sizes, see Recommendations for running Stash Server in AWS. • ✗ Stash Data Center is not supported in AWS at this time. • The hardware requirements for a full production deployment depend on the number and frequency of Git operations and the number of active users. See Scaling Stash for further discussion and for details of how memory is allocated for Stash and Git.
Memory	2GB+	
Operating systems		
Apple Mac OS X	✓ Evaluation ✗ Production	• Stash is a pure Java application and should run on any platform, provided all the Java requirements are satisfied. • In production environments Stash should be run from a dedicated user account.
Linux	✓	

Microsoft Windows	✅ < 500 users ❌ 500+ Enterprise	• ❌ Apple Mac OS X is not supported for production deployment. • ❌ Microsoft Windows is not supported for 500+ Enterprise tiers.
Java		
Oracle Java	✅ 1.8 ⚠️ 1.7 ❌ 1.6	• ⚠️ Support for Java 7 is deprecated, and will be removed in Stash 4.0. See End of support announcements for Stash. • Java 8 is supported, as of Stash 3.0. • Stash only requires the Java JRE, not the JDK. • For Oracle Java, we recommend using Server JRE 7, which you can download from the Oracle website. • For OpenJDK, download and install instructions for Linux flavors are at http://openjdk.java.net/install/. • Note that the Stash installer will install a supported version of the Java JRE, which is only available to Stash, if necessary. See Running the Stash installer. • ❌ OpenJDK 1.8u25 and later versions are not supported until https://bugzilla.redhat.com/show_bug.cgi?id=1167153 is fixed. • ❌ Support for Java 6 was *removed* in Stash 3.0, as previously announced. Pre-installed Java on some AWS EC2 Linux instances might be installed with a subset of features. See SSH server fails to start on AWS EC2 instance for more information.
OpenJDK	✅ 1.8u0–1.8u20 ❌ 1.8u25+ ⚠️ 1.7 ❌ 1.6	
Databases		
HSQLDB	✅ Evaluation ❌ Stash Data Center	• Please see connecting Stash to an external database. • HSQLDB is bundled with Stash, and is only intended for evaluation use. • ❌ HSQLDB is not supported in Stash Data Center.
Microsoft SQL Server / Microsoft SQL Server Express	✅ 2014 ✅ 2012 ✅ 2008 R2 ✅ 2008 ✅ 2005	

MySQL	Stash Server: ✅ 5.6.16+ ✅ 5.5.x ✅ 5.1.x ❌ 5.7+ ❌ 5.6.0 – 5.6.15 ✅ MariaDB 5.5 ❌ Stash Data Center	MySQL, while supported by Stash Server, is currently ***not*** recommended, especially for larger instances, due to inherent performance and deadlock issues that occur in this database engine under heavy load. Affected systems may experience slow response times, deadlock errors and in extreme cases errors due to running out of database connections. These issues are intrinsic to MySQL (no other database engine supported by Stash shares this behavior) and are due to the way MySQL performs row-level locking in transactions. See http://dev.mysql.com/doc/refman/5.0/en/innodb-deadlocks.html for some general information on this. Stash does its best to work around the MySQL behavior - see issues STASH-4517, STASH-4701 and others, for example. But under very heavy load you will generally get better performance with any of the other database engines supported by Stash (such as PostgreSQL, which is also freely available) than you will with MySQL. • MariaDB 5.5, but no other version, is supported as of Stash 3.6. See Connecting Stash to MySQL for more information. • ❌ MySQL 5.6.15 and earlier: Note that Stash *does not support* versions of MySQL 5.6 earlier than 5.6.16 at all, because of bugs in its query optimizer (#68424, #69005). See Connecting Stash to MySQL for more information. • ❌ MySQL 5.7+ is not supported. • ❌ MySQL (any version) is not supported in Stash Data Center.
Oracle	✅ 12c ✅ 11g	
PostgreSQL	✅ 9.0, 9.1, 9.2, 9.3, 9.4 ✅ 8.2, 8.3, 8.4	
Web browsers		
Chrome	✅ Latest stable version supported	
Firefox	✅ Latest stable version supported	
Internet Explorer	✅ 11 ⚠️ 10 ⚠️ 9 ❌ 8	• ⚠️ Support for Internet Explorer 10 is deprecated and will be removed in Stash 4.0. See End of support announcements for Stash . • ⚠️ Support for Internet Explorer 9 is deprecated and will be removed in Stash 3.10. See End of support announcements for Stash . • ❌ Support for Internet Explorer 8 was *removed* in Stash 3.0, as previously announced.
Safari	✅ Latest stable version supported	
DVCS		
Git – server		The table lists the versions of Git that have been tested against the **Sta**

Supported	Tested	
	Linux	*Windows*
2.4.1+		
⚠ 2.3.0 – 2.4.0	2.3.4	
⚠ 2.2.0 – 2.4.0	2.2.2	
2.1.0+	2.1.4	
2.0.0–2.0.1 2.0.4+	2.0.5	
1.9.0+	1.9.5	1.9.5.1
1.8.0–1.8.4.2 1.8.4.4+	1.8.0.3 1.8.1.5 1.8.2.3 1.8.3.4 1.8.4.5 1.8.5.6	1.8.0 1.8.1.2 1.8.3 1.8.4 1.8.5.2
⚠ 1.7.6+	1.7.6.6 1.7.7.7 1.7.8.6 1.7.9.7 1.7.10.5 1.7.11.7 1.7.12.4	1.7.6 1.7.7.1 1.7.8 1.7.9 1.7.10 1.7.11

sh 3.9.x releases.

- In general, we recommend using the most recent version of Git on both the Stash server and clients, where possible, and subject to the following notes and exceptions.
- The version of Git installed on machines that interact with Stash must be compatible with the version of Git installed for use by the Stash server.
- For Git 1.9.0 and later we only test the highest bugfix release – all earlier bugfix releases in that series are also supported, unless specifically indicated otherwise below.
- ⚠ Support for versions of Git earlier than v1.8 on the server is deprecated and will be removed in Stash 4.0. See End of support announcements for Stash.
- ⚠ Git 1.8.3.x has some performance regressions which may cause problems in Stash with large repositories.
- ⚠ Git 2.2.x - 2.4.0 have some performance issues when interacting with NFS. Hence, these versions are currently not supported for Stash Data Center or for Stash Server installations that use NFS mounts for the home directory (Details) .
- ✖ **Cygwin** : Cygwin Git *is not supported* for use on Windows servers, regardless of version.
- ✖ Git 1.7.1 is not supported.
- ✖ Git 1.8.4.3 is not supported due to a critical bug in how symbolic refs are handled which breaks pushing and pulling for repositories with pull requests. (Details)

  STASH-4101 - Clone and fetch fail with "protocol error: impossibly long line" CLOSED
- ✖ Git 2.0.2 and 2.0.3 are not supported due to a critical bug in `git diff-tree` which breaks Stash's commit page. (Details)

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

[Security vulnerability CVE-2014-9390] affects multiple Git versions. Stash itself is not affected, however end-users should update their *clients* to a patched maintenance version: v1.8.5.6, v1.9.5, v2.0.5, v2.1.4 and v2.2.1 or newer. For instructions see Installing and upgrading Git.

Git – client	✓ 1.6.6+	[Security vulnerability CVE-2014-9390] affects multiple Git versions. Stash itself is not affected, however end-users should update their *clients* to a patched maintenance version: v1.8.5.6, v1.9.5, v2.0.5, v2.1.4 and v2.2.1 or newer. For instructions see Installing and upgrading Git. Windows git client users are recommended to use Git-1.9.5-preview20150319 or higher which fixes an openssl vulnerability.
Additional tools		
Perl	✓ 5.8.8+	
Mail clients		
Apple Mail	✓ Apple Mail 4	
Gmail	✓ Latest	
iOS Devices	✓ iPhone, iPad	
Microsoft Outlook	✓ Express, 2007, 2010	
Outlook.com Hotmail Windows Live Mail	✓ Latest	

Notes:

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

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

Using Stash

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

This section describes the essentials of using Stash.

If you are setting up Stash, see the Getting started section. If you want to configure Stash, see the Administering Stash section.

See Getting started with Git and Stash for an overview of how to work with Stash.

Related pages:

- Getting started
- Git Tutorials and Training
- Git resources
- Administering Stash
- Stash FAQ

Working with projects

Stash manages related repositories as projects. Find out how to set up projects and then give your teams access to those.

Working with repositories

If you have existing projects that you want to manage in Stash, then you'll want to read Importing code from an existing project.

See also:

- Creating repositories
- Controlling access to code
- Using pull requests in Stash

Git resources

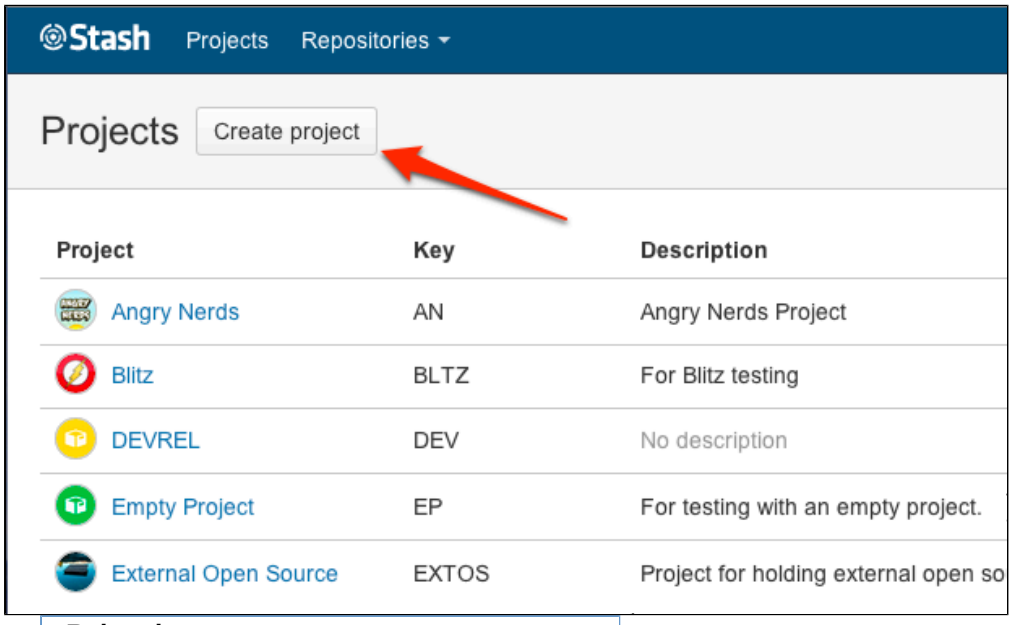
For those who are new to using Git:

- Using pull requests in Stash
- Basic Git commands
- Permanently authenticating with Git repositories

Creating projects

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

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



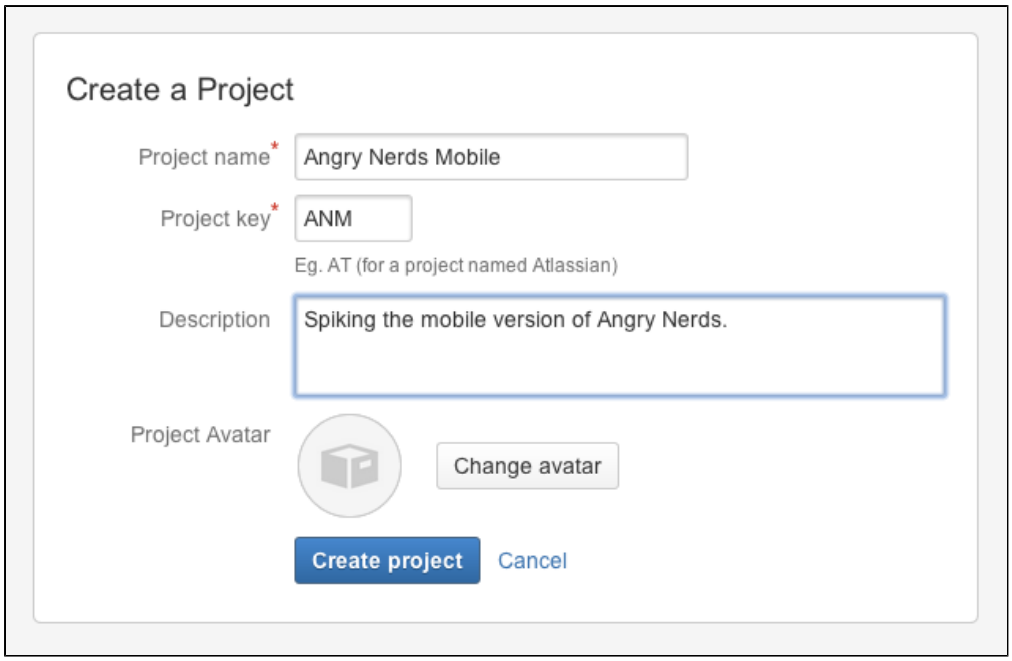
Related pages:

- Getting started with Git and Stash
- Using project permissions
- Creating repositories
- Global permissions

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

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

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



You'll want to add repositories to the project. See Creating repositories for details.

Creating repositories

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

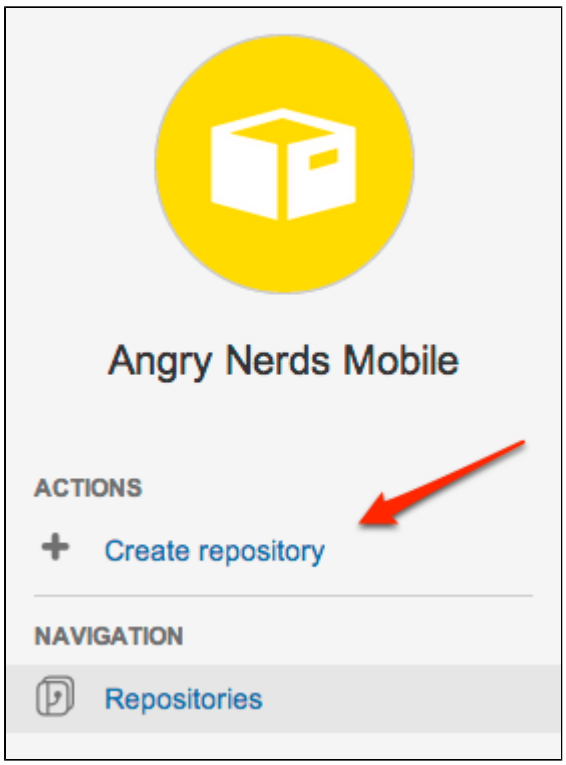
In order to create repositories you need to have Project Admin permission for the project to which you want to add a repository.

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

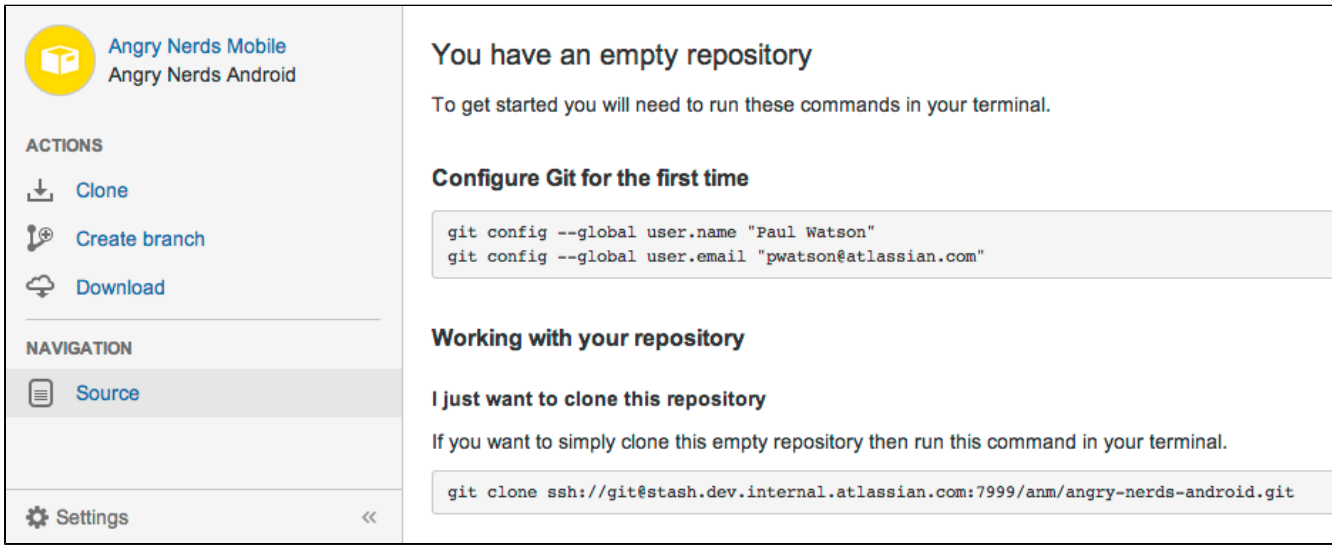
Related pages:

- Creating personal repositories
- Using repository permissions
- Creating projects
- Importing code from an existing project

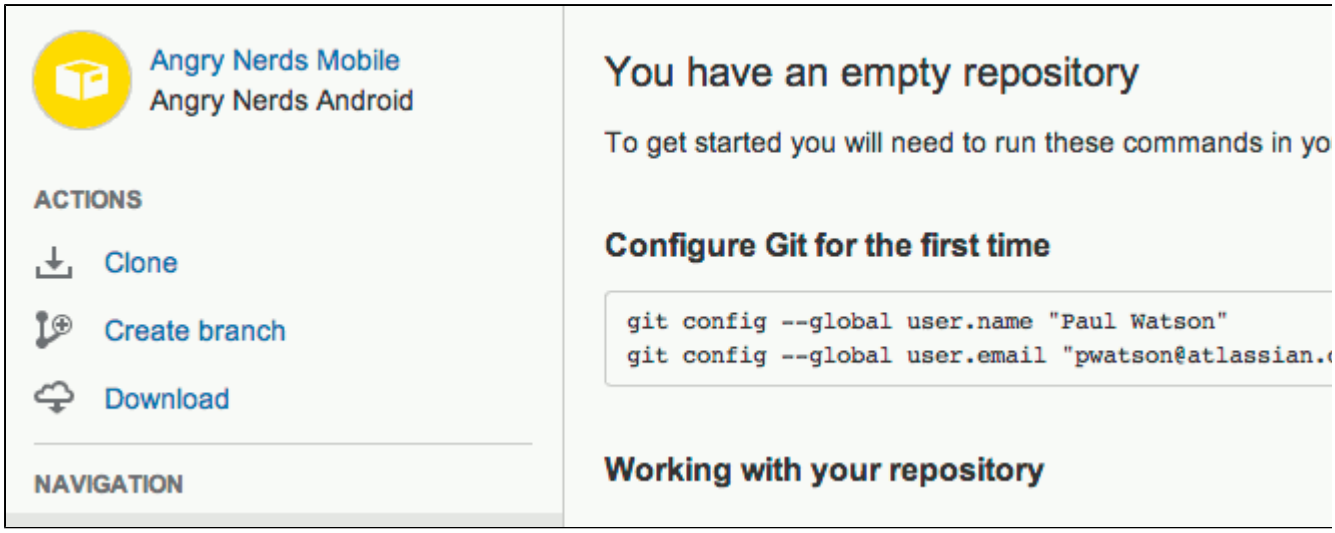
Go to the project and click **Create repository** to open the repository creation form:



Once submitted you will be taken directly to your repository homepage. 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 lefthand sidebar 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 permissions at the parent project level. More information is available on Creating projects.

Creating personal repositories

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

By default, personal repositories are not visible to other Stash users (unless they are a Stash system administrator). However, you can:

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

You can create personal repositories in 2 ways:

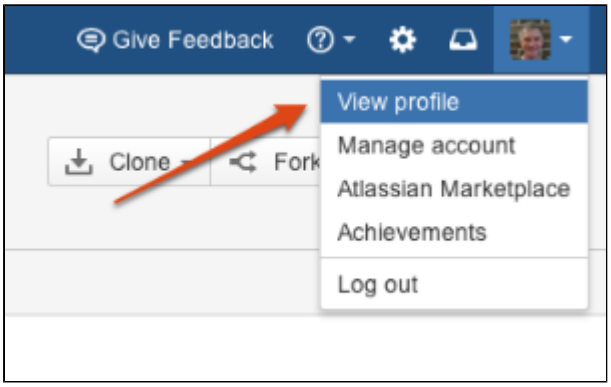
- Directly, from your profile.
- By forking another repository.

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

Directly creating a personal repository

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

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

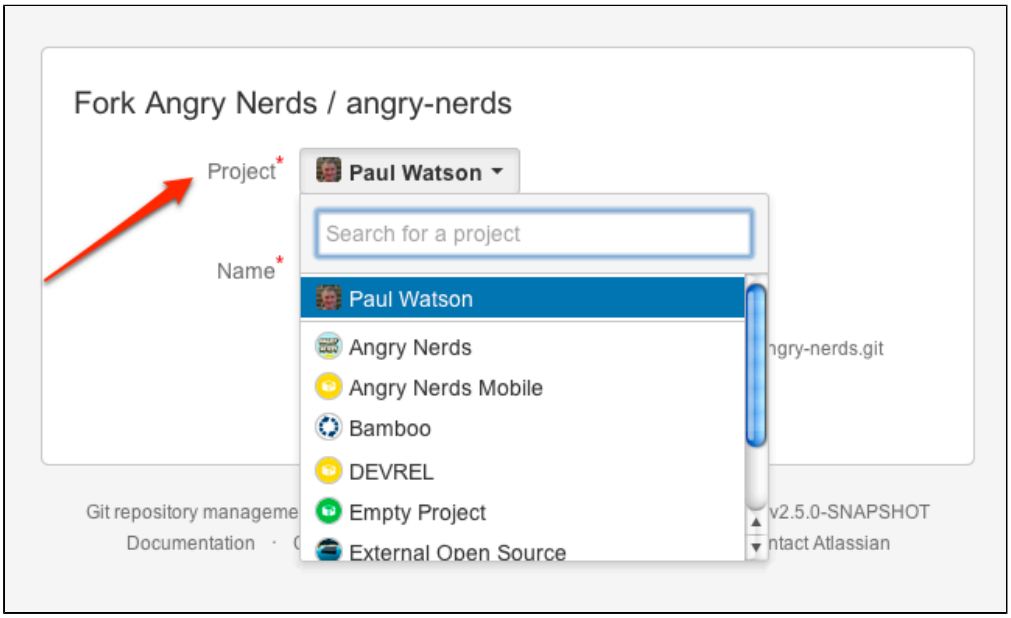


2. Click **Create repository**.
3. Set repository permissions on the new repository, if required.

Forking another repository

You can create a personal fork of any other repository in Stash for which you have permission:

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



4. Click **Fork repository**.
5. Set repository permissions on the new repository, if required.

Using repository hooks

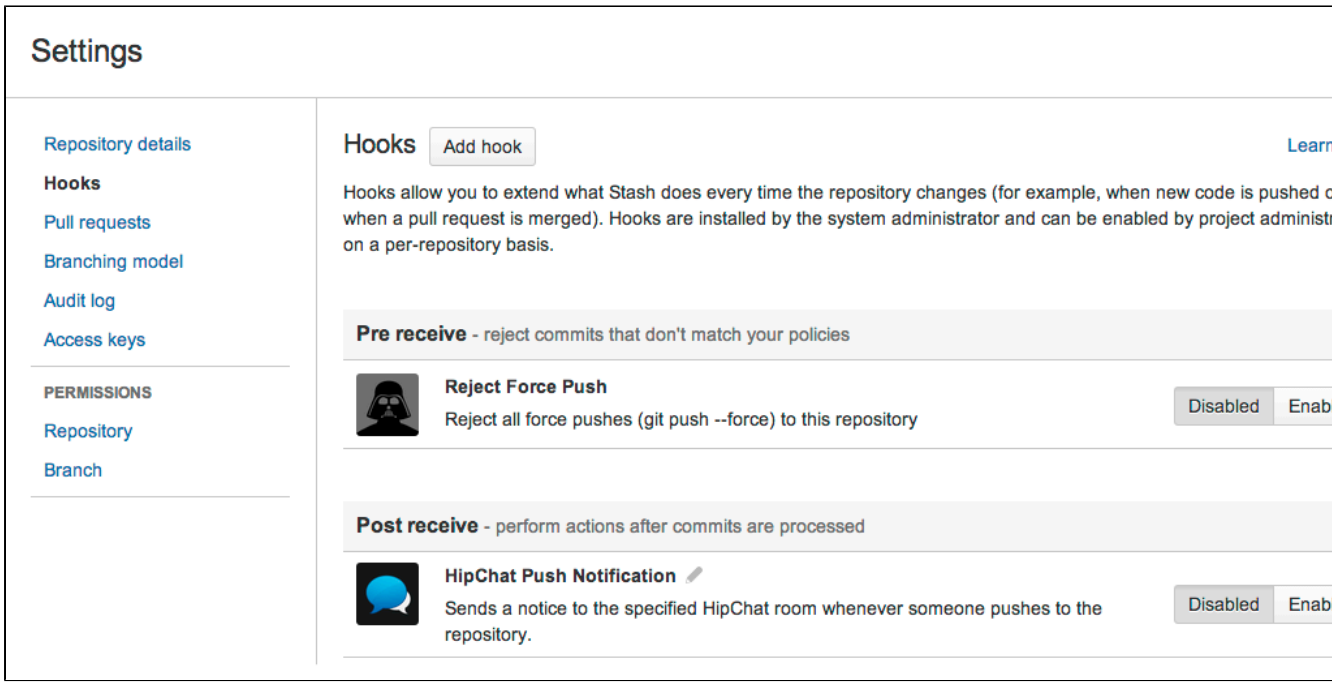
Hooks in Stash provide a way to customise a team's workflow and integrate with other systems. Stash currently supports two types of hooks, **pre** and **post-receive**.

On this page:

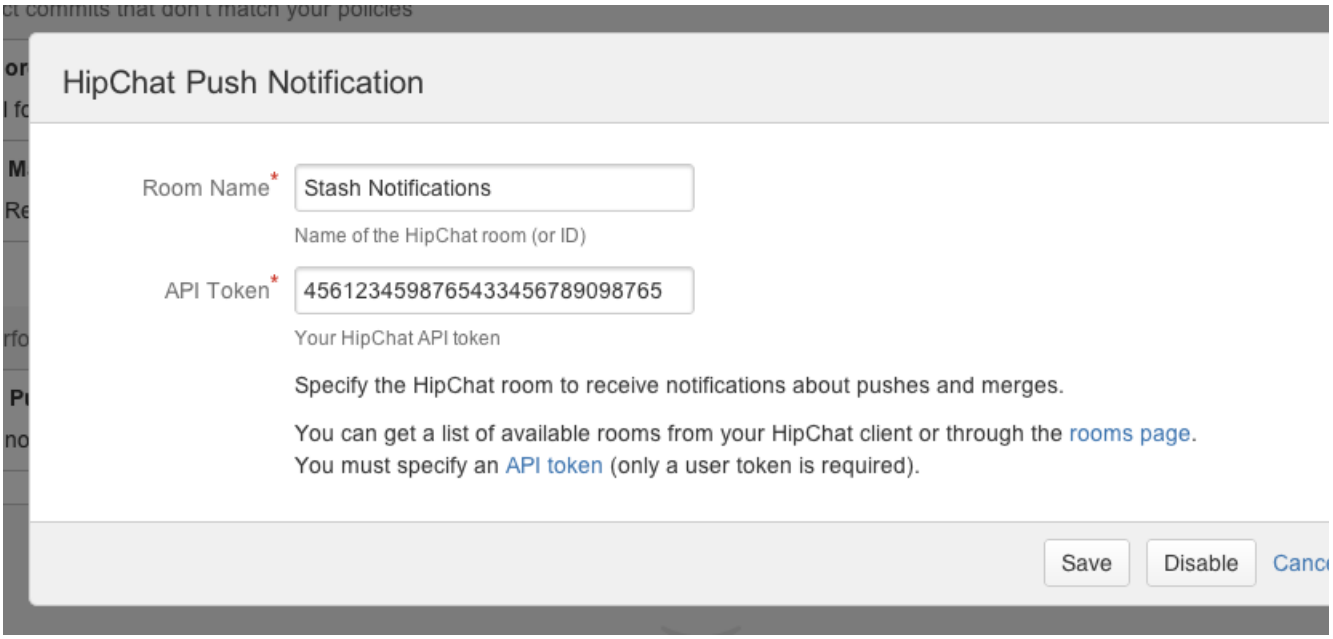
- Managing hooks
  - Pre-receive hooks
  - Post-receive hooks
- Getting hooks from the Atlassian Marketplace
- Creating your own hooks

Managing hooks

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



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



Stash currently ships with the following hooks:

- Reject Force Push – block all Git force pushes (`git push -- force`).
- HipChat Push Notifications – send a message to a HipChat room when someone pushes to the repository.

Pre-receive hooks

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

Post-receive hooks

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

Stash supports two types of post-receive hook:

- PostReceiveHooks map to Git's `post-receive` hooks. They run on the Stash server after a push.
- AsyncPostReceiveRepositoryHooks, executed by the Stash server.

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

Getting hooks from the Atlassian Marketplace

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

Creating your own hooks

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

For information about how to write your own hooks please see the Stash developer docs.

In particular, these pages will be helpful:

- Repository hooks
- Repository hook plugin module

See too this blog post about hooks for Stash: http://blogs.atlassian.com/2013/03/stash-git-hooks-api/

Permanently authenticating with Git repositories

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

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

On this page:

- Using credential caching
- 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.
3. You will be prompted for credentials the first time you access a repository, and Windows will store your credentials for use in the future.

Linux

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

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

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

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

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

```
git help credential-cache
```

OS X

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

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

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

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

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

Using the .netrc file

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

Warning!

- Git uses a utility called cURL under the covers, which respects the use of the .netrc file. Be aware that other applications that use cURL to make requests to servers defined in your `.netrc` file will also now be authenticated using these credentials. Also, this method of authentication is potentially unsuitable if you are accessing your 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:

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

Linux or OS X

1. Create a file called `.netrc` in your home directory (`~/.netrc`). Unfortunately, the syntax requires you to store your passwords in plain text - so make sure you modify the file permissions to make it readable only to you.
2. Add credentials to the file for the server or servers you want to store credentials for, using the format described in the 'Windows' section above. You may use either IP addresses or hostnames, and you do *not* need to specify a port number, even if you're running 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.

Controlling access to code

Stash provides the following types of permissions to allow fully customisable control of access to code.

Note that you can also:

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

Global permissions

- Control user and group access to Stash projects and to the Stash server configuration.
- For example, these can be used to control the number of user accounts that can can access Stash for

licensing purposes.
- See Global permissions.

Project permissions

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

Repository permissions

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

Branch permissions

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

Permissions matrix

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

Key

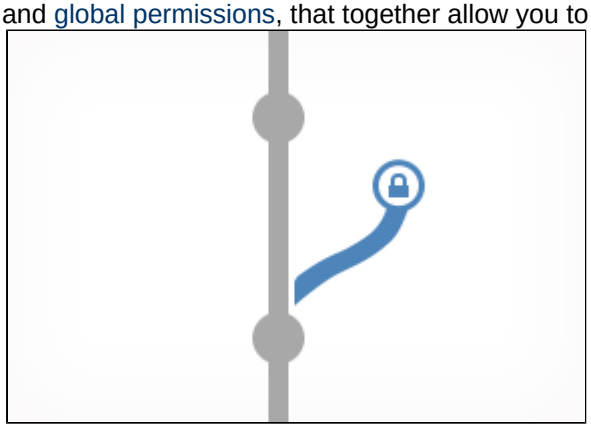
Permission	Effect
BROWSE	Can view repository files, clone, pull to local
READ	Can browse, clone, pull, create pull requests, fork to a personal project
WRITE	Can merge pull requests
ADMIN	Can edit settings and permissions

Global (logged in)	Project	Repository	Branch	Effective permission
✗	Personal	Personal	NA	No access
✗	Personal	Public access	NA	BROWSE just that repo
✗	No access	No access	NA	No access
✗	No access	Public access	NA	BROWSE just that repo
✗	Public access	Public access	NA	BROWSE all repos in project
✓	Personal	Personal	NA	No access
✓	Personal	Public access	NA	READ just that repo
✓	No access	No access	NA	No access
✓	No access	Public access	NA	READ just that repo

✓	Public access	No access	NA	READ all repos in project
✓	Public access	Public access	NA	READ
✓	Public access	Public access	For this user	READ that branch, no WRITE
✓	No access	READ	NA	READ just that repo
✓	Public access	READ	NA	READ just that repo
✓	READ	No access	NA	READ all repos in project
✓	READ	Public access	NA	READ all repos in project
✓	READ	READ	NA	READ all repos in project
✓	READ	No access	For this user	READ that branch, no WRITE
✓	No access	WRITE	NA	WRITE just that repo
✓	Public access	WRITE	NA	WRITE just that repo
✓	WRITE	No access	NA	WRITE all repos in project
✓	WRITE	WRITE	NA	WRITE all repos in project
✓	WRITE	WRITE	For other users	WRITE to other branches only
✓	ADMIN			Can edit settings and permissions

Using branch permissions

Branch permissions allow you to control who can commit to specific branches in a repository. Branch permissions provide another level of security within Stash, along with user authentication and project, repository and global permissions, that together allow you to control, or enforce, your own workflow or process.



Branch permissions:

- are based on users or groups.
- are actually restrictions, which are checked after project and repository level permissions.
- are used to limit branch access to specific people who must still have write access to the project or repository.
- prevent unauthorised users pushing to or deleting the branch.
- are based on explicit branch names, or you can use advanced branch permissions to match multiple branches (or tags) using pattern matching.

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

If a user does not have commit access to the branch, an error message will be shown on the Git command line when they try to push a change to the branch.

Note that if no branch permissions are defined then anyone with commit access to the repository can push to any branch.

On this page:

- Setting branch permissions
- Advanced branch permissions

Related pages:

- Using pull requests in Stash
- Controlling access to code
- Global permissions

Setting branch permissions

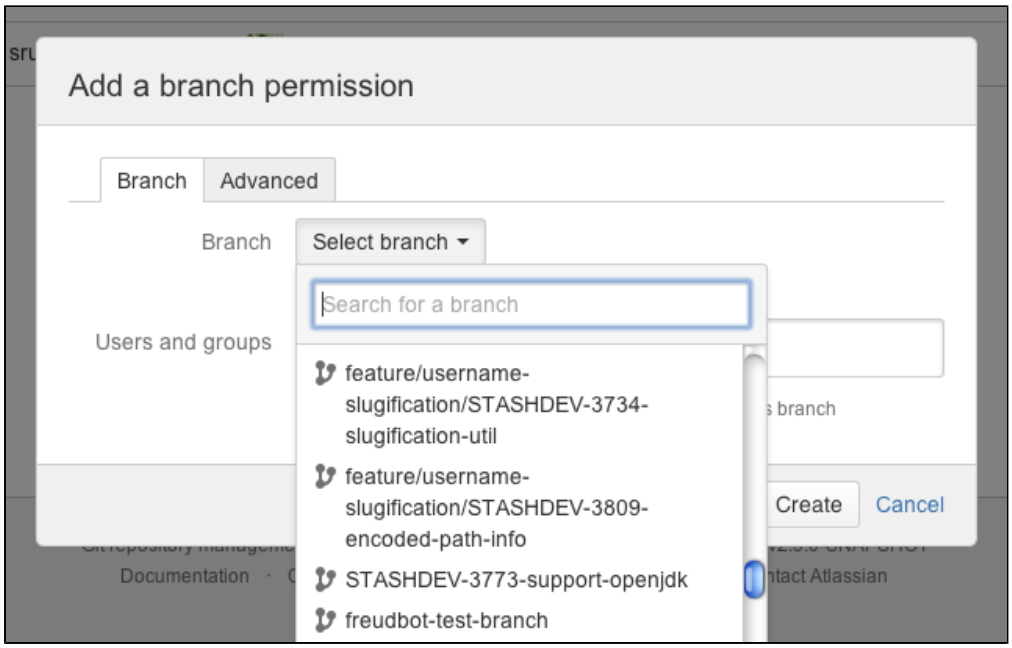
Branch permissions in Stash are set on a per-repository basis. Makes sense – branch permissions control access to *repository* branches, right?

You'll need either project admin, admin or sys-admin permissions to set branch permissions.

So, to set branch permissions:

1. Go to a repository in a project.
2. Choose **Settings** > **Branch** (under 'Permissions').
3. Click **Add permission**.
4. On the **Branch** tab, choose the branch for which you want to control access.
5. Add (or remove) users or groups that you want to have (or not have) commit access to the branch.
6. Click **Create** to finish.

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



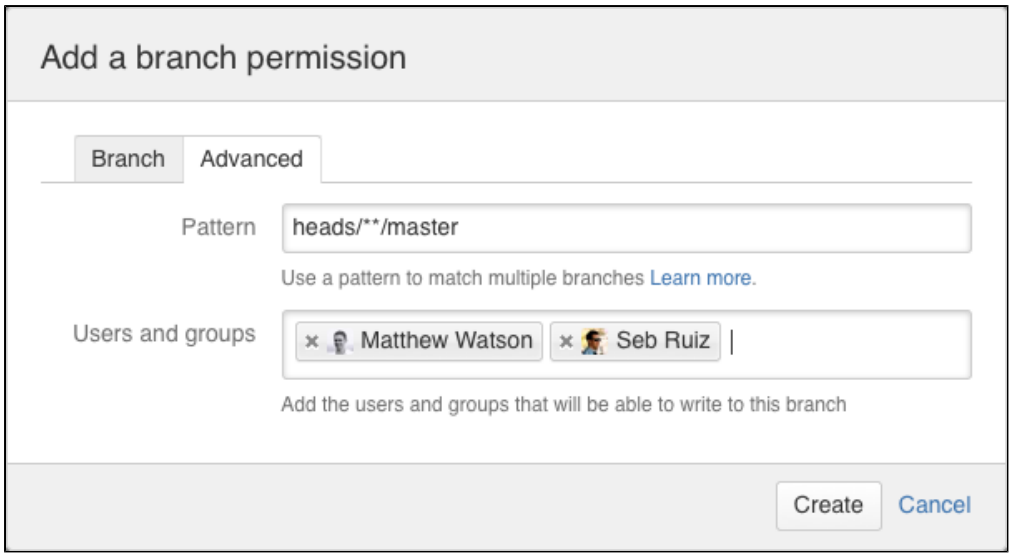
Advanced branch permissions

Advanced branch permissions specify a pattern that is matched against branches *and tags* being pushed to Stash; this allows you to restrict any pushes to branches that match the pattern.

Advanced branch permission also apply to attempts to create new branches; if a push to Stash attempts to create a new branch that matches a pattern, the user must be authorised for the operation to proceed.

To set advanced branch permissions, choose **Settings** > **Branch**, and click **Add permission**, as described above.

On the **Advanced** tab, enter a glob pattern to match the names of multiple branches for which you want to control access.



Branch permission patterns

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

These expressions use the following wild cards:

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

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

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

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

Also see the Ant documentation.

Examples

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

Using repository permissions

Stash allows you to manage the permissions for just a single repository, or for a group of repositories together from the project.

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

Stash supports 3 levels of permissions for repositories:

- Admin
- Write
- Read

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

Related pages:

- Using project permissions
- Using branch permissions
- Global permissions
- Allowing public access to code

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

Note that:

- Anyone with permission to browse a pull request can create a task on any comment, and can browse, resolve and reopen existing tasks in the pull request.
- Repository admins and pull request authors can edit and delete *any* task in the pull request. Reviewers and others can only edit or delete their *own* tasks.

Granting access to a repository

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

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

Granting access to all repositories within a project

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

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

Granting permission to create repositories

Only users with project administration permission can create *new repositories*.

Using project permissions

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

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

Related pages:

- Creating projects
- Global permissions
- Using branch permissions
- Using repository permissions
- Allowing public access to code

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

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

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

.

Allowing public access to code

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

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

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

In Stash, you can:

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

On this page:

- Making a repository publicly accessible
- Making a project publicly accessible
- Viewing public repositories
- Disabling public access globally

Related pages:

- Using project permissions
- Using repository permissions

Making a repository publicly accessible

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

You need admin permission for the repository.

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

Making a project publicly accessible

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

You need admin permission for the project.

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

Viewing public repositories

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

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

Disabling public access globally

Stash provides a system property that allows you to turn off public access for the whole instance.

To do this, set the `feature.public.access` property to false in the `stash.config.properties` file in your Stash home directory.

Using SSH keys to secure Git operations

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

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

Related pages:

- Creating SSH keys
- Enabling SSH access to Git repositories in Stash
- Permanently authenticating with Git repositories

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

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

Note that:

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

Creating SSH keys

This page describes how to create SSH keys.

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

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

The SSH key needs to be added to Stash, and your Stash administrator must have enabled SSH access to Git repositories, before you can make use of the key.

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

You can use an existing SSH key with Stash if you want, in which case you can go straight to either SSH user keys for personal use or SSH access keys for system use.

On this page:

Related pages:

- Using SSH keys to secure Git operations
- Enabling SSH access to Git repositories in Stash
- Permanently authenticating with Git repositories

Creating an SSH key on Windows

1. Check for existing SSH keys

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

Open a command prompt, and run:

```
cd %userprofile%/.ssh
```

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

  ```
  dir id_*
  ```

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

2. Back up old SSH keys

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

In a command prompt on your local computer, run:

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

3. Generate a new SSH key

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

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

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

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

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

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

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

Creating an SSH key on Linux & Mac OS X

1. Check for existing SSH keys

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

Open a terminal and run the following:

```
cd ~/.ssh
```

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

```
ls id_*
```

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

2. Back up old SSH keys

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

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

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

3. Generate a new key

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

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

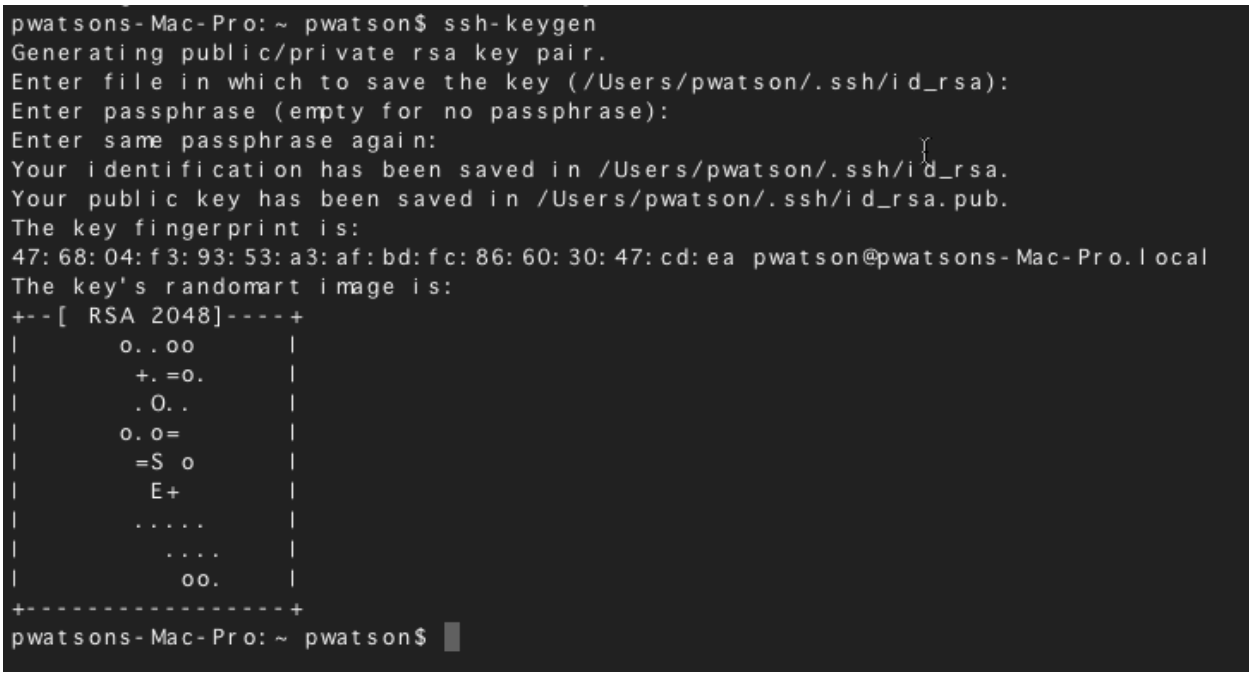
   ```
   ssh-keygen -t rsa -C "your_email@example.com"
   ```

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

   You'll see a response similar to this:

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

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

SSH user keys for personal use

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

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

- your Stash administrator must have already enabled SSH access in

Stash.

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

Once you have an SSH key associated with your Stash account, using it is easy! See Use SSH keys to connect to Stash repositories below.

Related pages:

- Creating SSH keys
- Enabling SSH access to Git repositories in Stash
- Permanently authenticating with Git repositories

Note that:

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

Add an SSH key to your Stash account

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

Windows

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

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

Mac OS X

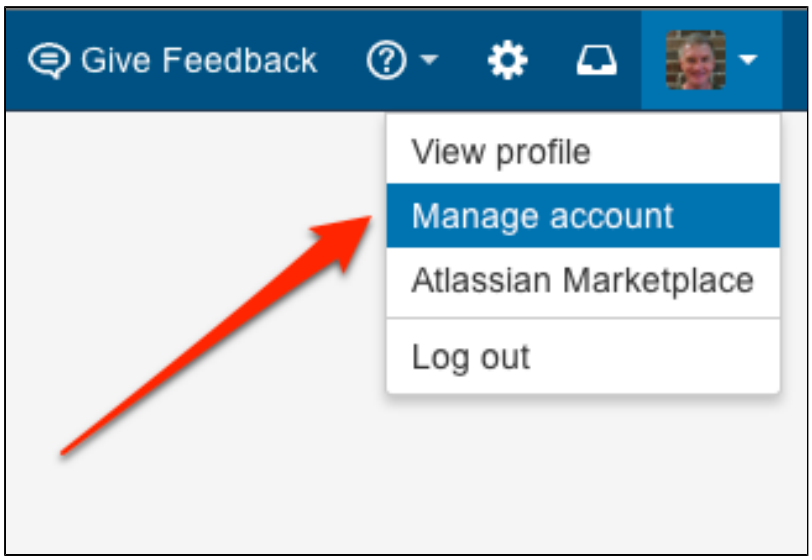
```
pbcopy < ~/.ssh/id_rsa.pub
```

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

Linux

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

2. In Stash, go to your account:



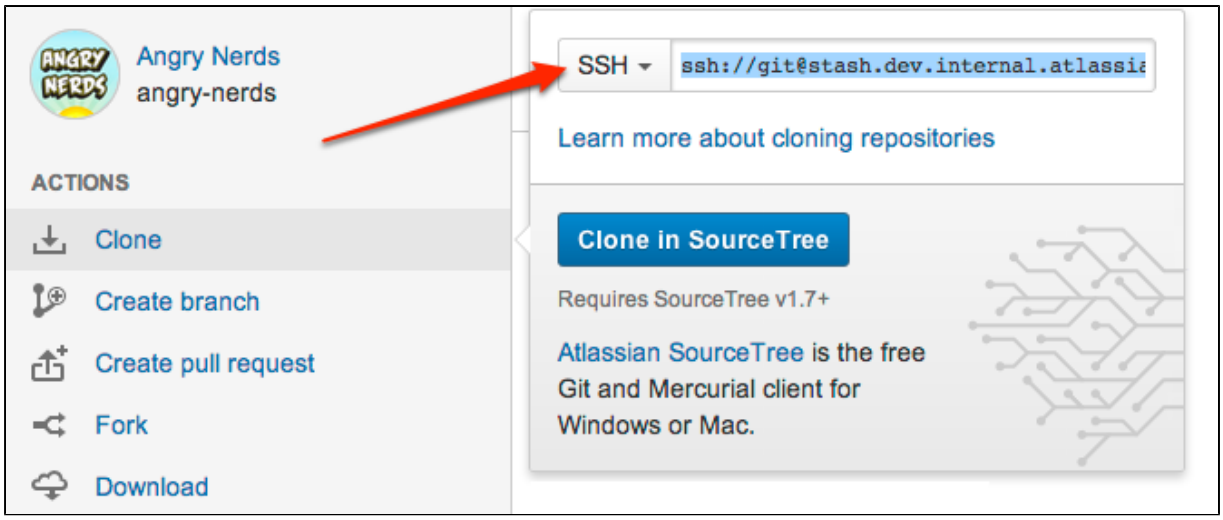
3. Click on **SSH keys** and then **Add key**.
4. Paste the key into the text box:

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

Use SSH keys to connect to Stash repositories

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

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



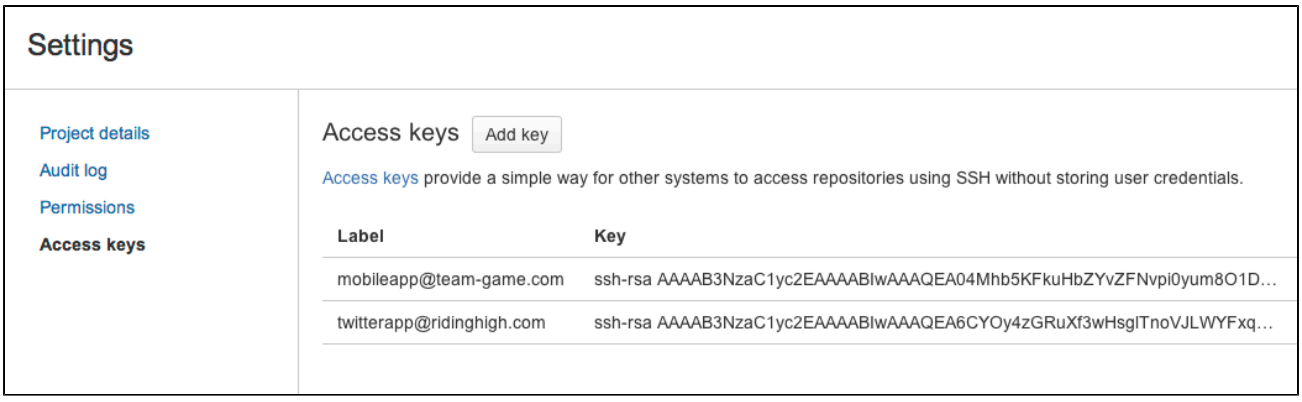
SSH access keys for system use

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

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

Related pages:

- Creating SSH keys
- Enabling SSH access to Git repositories in Stash
- Permanently authenticating with Git repositories



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

Before you can use SSH keys to secure a connection with Stash the following must have already been

done:

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

Using SSH keys to allow access to Stash repositories

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

- Add the private key to that system. For Bamboo, see this page: Sharing repository credentials.
- Add the public key to Stash as described here:

Add an SSH access key to either a Stash project or repository

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

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

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

2. Now, in Stash, go to the **Settings** tab for the project or repository.
3. Click **Access keys** and then **Add key**.
4. Choose the **Read** permission, for `git pull` or `git clone` operations for example, where you want to be sure that the system will *not* be able to write back to the Stash repository.
   Choose the **Read / Write** permission, for `git push` or `git merge` operations for example, where you may want your build system to merge successful feature branch builds to the default branch in the Stash repository, or so that deployments can be tagged.
   Note that if you attempt to add a key already present on a project or repository but with a different permission to what it currently has, the permission will simply be updated.
5. Paste the key into the text box and click **Add key**.

Stash license implications

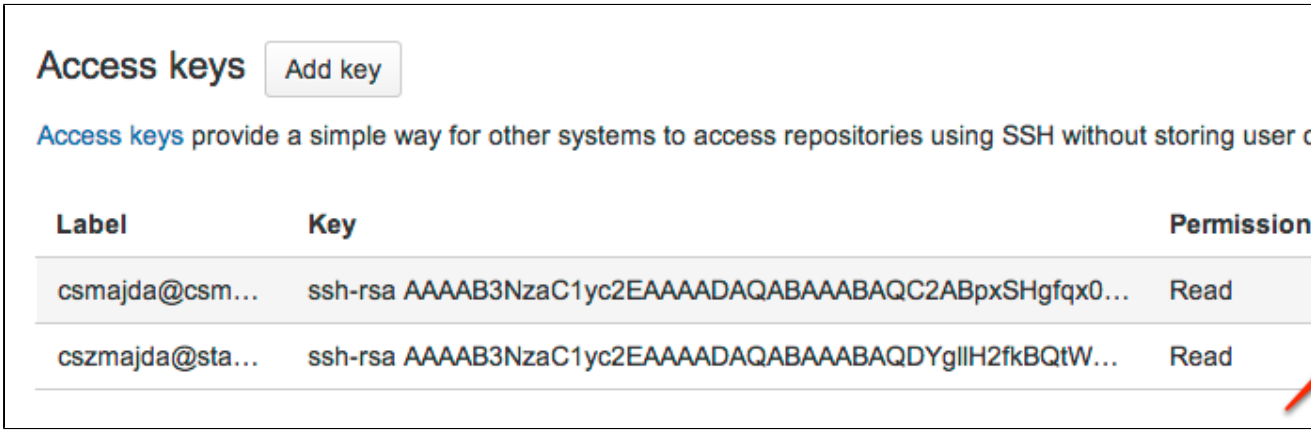
- System access keys do not require an additional Stash user license.

Reusing access keys

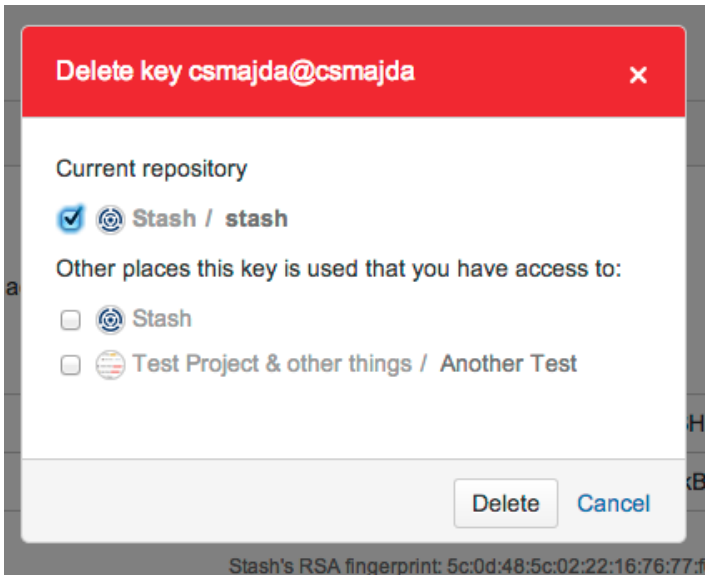
- You can use the same SSH access key for multiple repositories or projects.
- Keys used for personal user accounts can't be re-used as a project or repository system access key, and keys used as a project or repository access key can't be re-used for user accounts.

Deleting an access key

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



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



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

Workflow strategies in Stash

Various Git workflows are supported by Stash:

- Centralized Workflow
- Feature Branch Workflow
- Gitflow Workflow
- Forking Workflow

For information about setting up Git workflows in Stash see Using branches in Stash and Using forks in Stash.

Centralized Workflow

Like Subversion, the

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

Read more about the Centralized Workflow...

Feature Branch Workflow

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

Read more about the Feature Branch Workflow...

Gitflow Workflow

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

Read more about the Gitflow Workflow...

Forking Workflow

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

Read more about the Forking Workflow...

Using branches in Stash

Stash makes it easy for each member of your team to use a branching workflow for your Git development process. Your workflow can be mapped to branches in the Stash 'branching model', allowing Stash to:

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

On this page:

- Configuring the branching model
- Creating branches
- Automating the branch workflow
- Managing all your branches
- Read more

See also Using branch permissions for information about restricting access to branches in Stash.

Configuring the branching model

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

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

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

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

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

Development This is generally the integration branch for feature work and is often the default branch (e.g. `master`) or a named branch such as `develop`. In a workflow using pull requests, this is usually the branch where new feature branches are targeted. In other cases, developers might commit directly to this branch.	
Feature Feature branches are used for specific feature work or improvements. They generally branch from, and merge back into, the development branch, by means of pull requests. See Feature branch workflow.	`feature/`
Production The production branch is used while deploying a release. It branches from, and merges back into, the development branch. In a Gitflow-based workflow it is used to prepare for a new production release.	
Release Release branches are used for release task and long-term maintenance of software versions. Typically, they branch from, and fixes are merged back into, the development branch. Merging into an older release branch allows for automatic merging to newer release branches as well as the development branch.	`release/`
Bugfix Bugfix branches are typically used to fix release branches.	`bugfix/`
Hotfix Hotfix branches are used to quickly fix the production branch without interrupting changes in the development branch. In a Gitflow-based workflow, changes are usually merged into the production and development branches.	`hotfix/`

Note that:

- Prefixes can't be empty.
- Prefixes can't be longer that 30 characters.
- Prefixes can't overlap; for example PROD and PRODUCT would be overlapping prefixes.
- For Stash instances using Microsoft SQL Server, prefixes can't use non-ASCII characters. See
  STASH-3884 - Non-ASCII values used as branch model prefixes/branch names don't work in MSSQL **OPEN** .

Creating branches

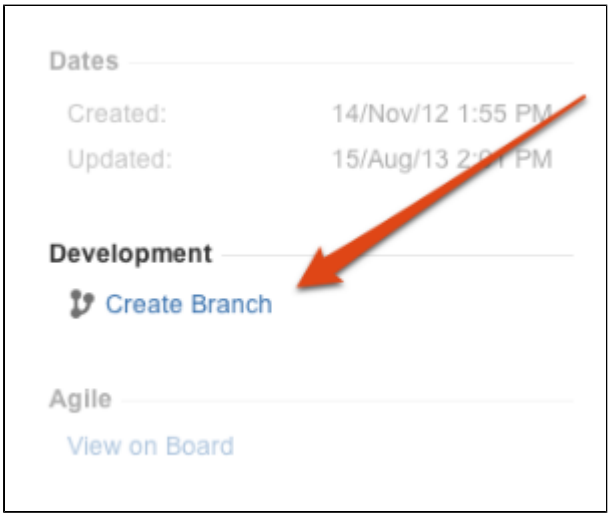
You can create a new branch when in JIRA (version 6.1 and above) or in Stash. Either way, you can override

the settings that Stash suggests for the repository, branch type, branching point and branch name.

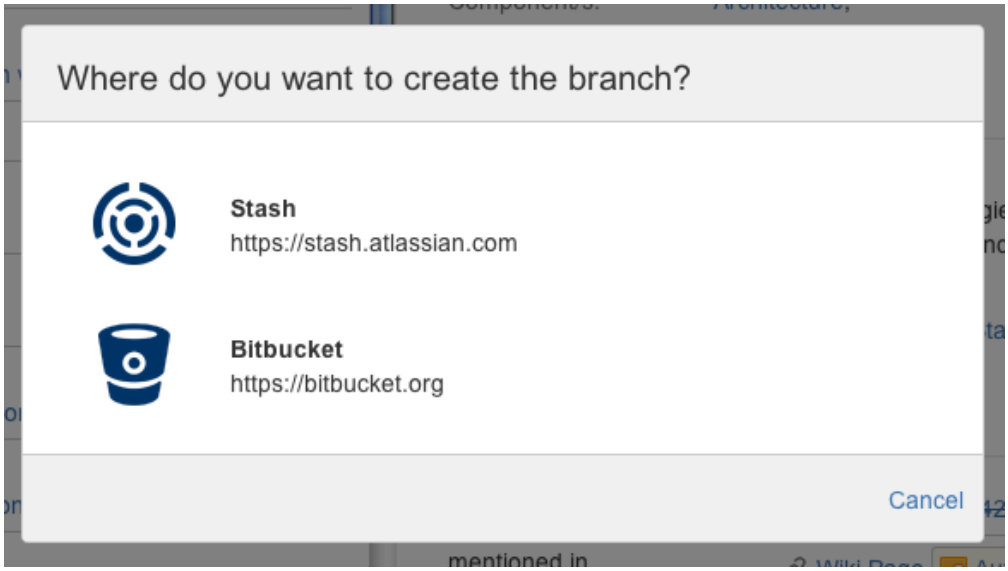
Create a branch from a JIRA issue

JIRA must be connected with Stash by an application link for this functionality to be available.

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



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

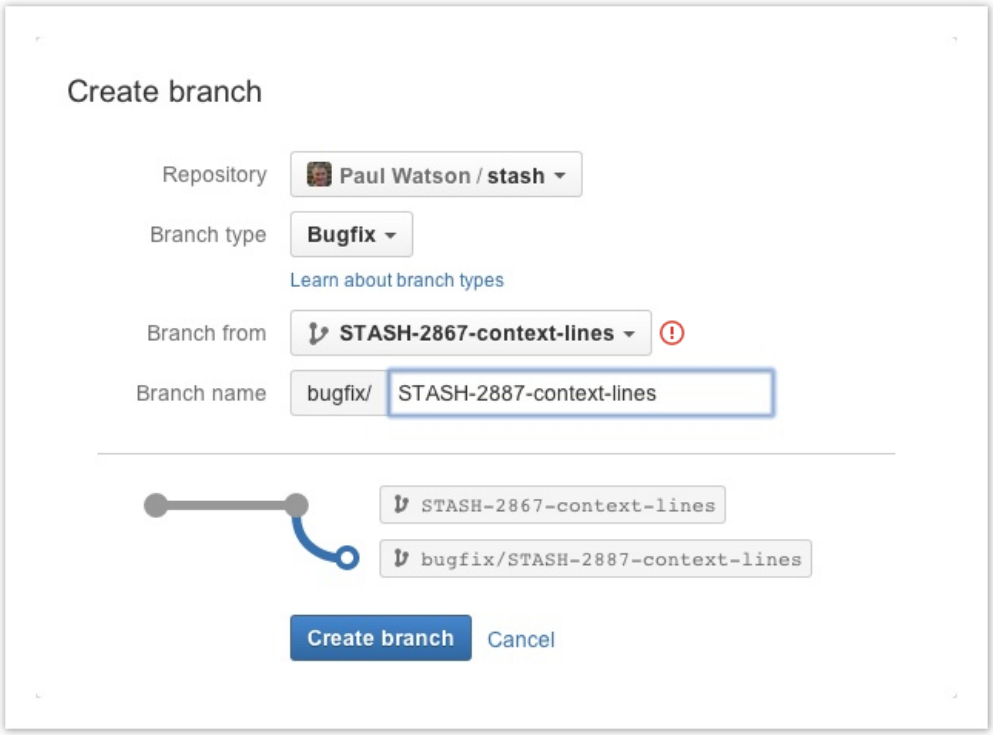


Stash suggests the **Branch type** and **Branch name** based on the JIRA issue type and summary. Change the settings suggested by Stash, if necessary:

Create a branch from within Stash

In Stash, choose **Create branch** from the sidebar.

Stash will suggest the **Branch type** and **Branch name** based on the JIRA issue type and summary. Notice that Stash displays the current build status beside the source branch picker. Change the settings suggested by Stash if necessary:



Creating the branch

You can specify:

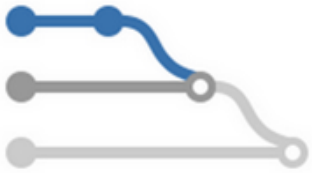
- the **Repository**
- the **Branch type**, if a branching model has been previously configured – choose **Custom** if you need an *ad hoc* branch type
- the **Branch from** point – you can choose either a branch or a tag.
- the **Branch name** – the prefix is based on the branch type you selected, and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

Note that Stash suggests a **Branch type** based on the JIRA issue type, when a branching model is configured. The mapping is:

JIRA issue type	Stash branch type
Bug	Bugfix
Story	Feature
New Feature	Feature

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

Automating the branch workflow



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

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

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

See Automatic branch merging for more information about the conditions for automatic merging, and how Stash determines the ordering of branches.

Managing all your branches

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

Searching for branches

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

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

Reading the table

Behind/Ahead

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

Pull requests

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

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

Builds

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

Actions

The Actions menus include tasks for working with branches:

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

Navigation

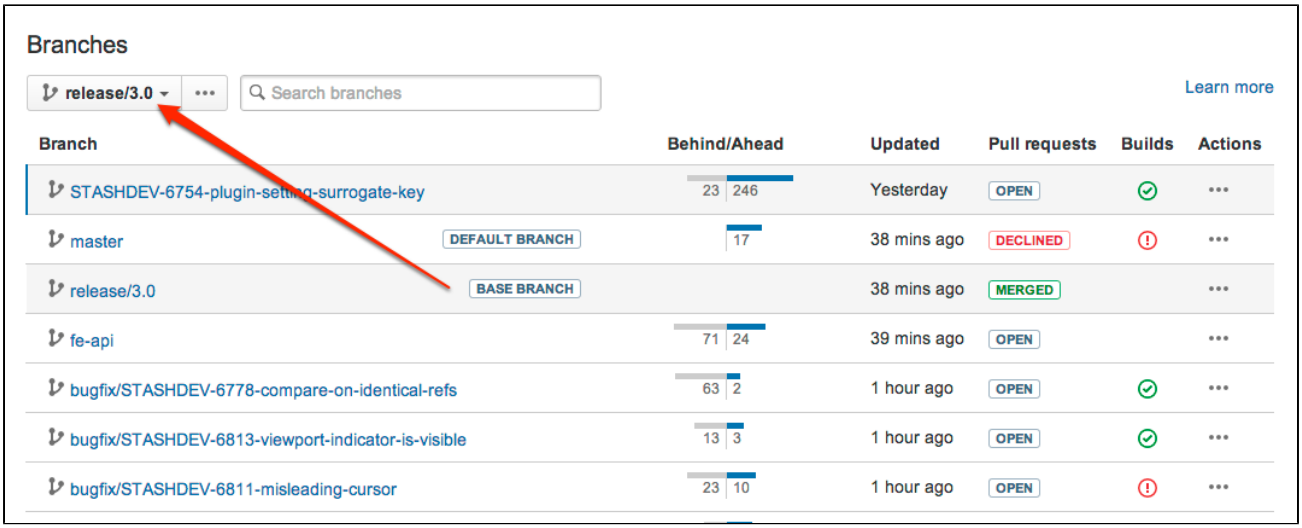
Choose **Keyboard shortcuts** from the Stash Help menu to see shortcuts to help you navigate quickly around the branch listing.

Checking on your branches

The branch listing allows you to:

- See how many commits behind or ahead your branch is compared to a chosen 'base branch'.
- See the latest status for pull requests originating from branches.
- See the build status of branches at a glance.

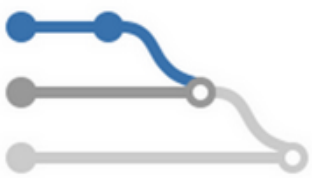
- The **Pull requests** status helps you to track the review and merge work that still needs to be done and can help with branch cleanup. For example, in combination with the **Behind/Ahead** infomation, you can decide whether to remove a feature branch that has already been merged.
- The **Behind/Ahead** column can help you to identify work in progress as well as stale branches. It is calculated for each branch against the base branch.



Read more

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

Automatic branch merging



Stash can automatically merge changes to newer release branches, thus reducing the need for manual maintenance of branches. To be able to do this, Stash has to be able to determine the ordering of branches, and relies on semantic versioning of branch names – for example Stash will order these branch names like this: 1.0.0 < 2.0.0 < 2.1.0 < 2.1.1.

Note that:

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

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

On this page:

- Conditions for automatic merging
  - What happens if the automatic merge fails?
- Branch ordering algorithm
- Ordering examples

Conditions for automatic merging

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

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

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

What happens if the automatic merge fails?

The automatic merge can fail for reasons such as:

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

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

Branch ordering algorithm

Stash is able to automatically merge changes to newer release branches, as long as Stash can determine the ordering of those branches. Ordering is based on semantic versioning in the naming pattern for branches.

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

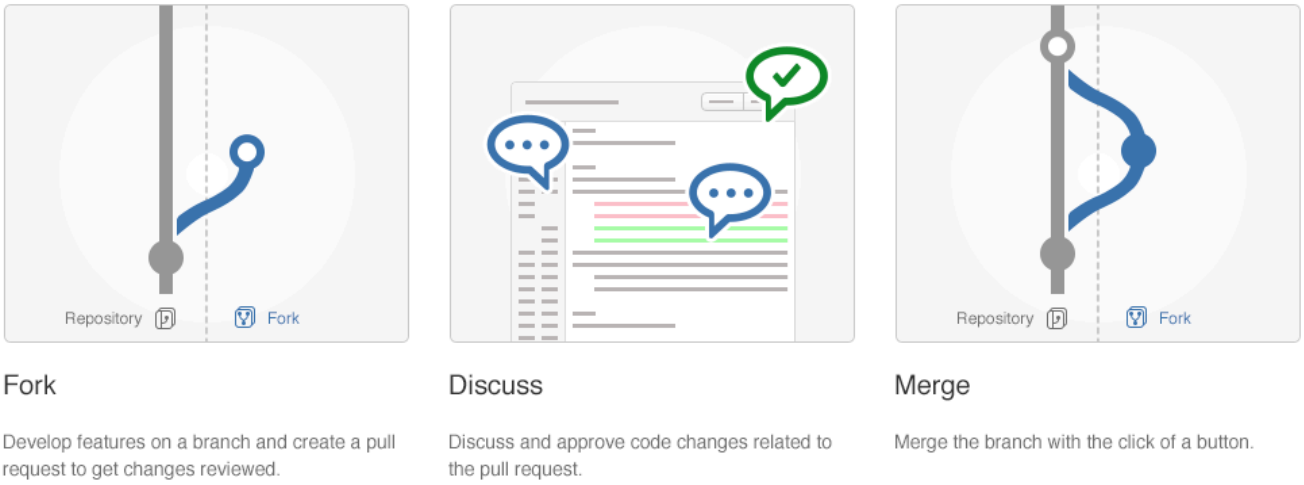
- Branches are selected and ordered on the basis of the name of the branch that started the cascade (i.e. the target of the pull request for the merge).
- Branch names are split into tokens using any of these characters: underscore '_', hyphen '-', plus '+', or period '.'.
- Only branches *matching*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

Ordering examples

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

GOOD	• release/1.0 • release/1.1-rc1 • release/1.1 • release/1.2 • release/2.0	Stash tokenises on the '.' and the '-' of '1.1-rc1' and is able to order these branch names correctly.
GOOD	• release/stash_1.1 • release/stash_1.2 • release/stash_2.0	Stash tokenises on the '.' and the '_' and orders the numeric parts of these branch names correctly.
BAD	• release/1.0 • release/stash_1.1	Stash tokenises on the '.' and the '_' but cannot recognise that 'stash_1.1' should follow '1.0'.

Using forks in Stash



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

You can fork a repository into any other project in Stash for which you have admin access. You can also create personal forks and give other developers access to that using repository permissions.

Creating a fork

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

Simply click **Fork** in the sidebar. You can choose the location for the newly forked repository. Note that when a repository is forked into another project it will get that project's permissions, which may be less restrictive.

When creating the fork you can enable fork syncing to have Stash automatically keep your fork up-to-date with changes in the upstream repository.

On this page:

- Creating a fork
- Issuing a pull request for a fork
- Merging a fork
- Synchronizing with upstream
- Disabling forking
- Pre-receive hooks and forks

Related pages:

- Workflow strategies in Stash
- Controlling access to code
- Creating personal repositories

Issuing a pull request for a fork

Pull requests for forks in Stash work just the way you'd expect. See Using pull requests in Stash.

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

Merging a fork

Once a pull request has been approved by reviewers, it can be merged as usual. See Using pull requests in Stash.

Synchronizing with upstream

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

Disabling forking

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

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

Pre-receive hooks and forks

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

Keeping forks synchronized

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

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

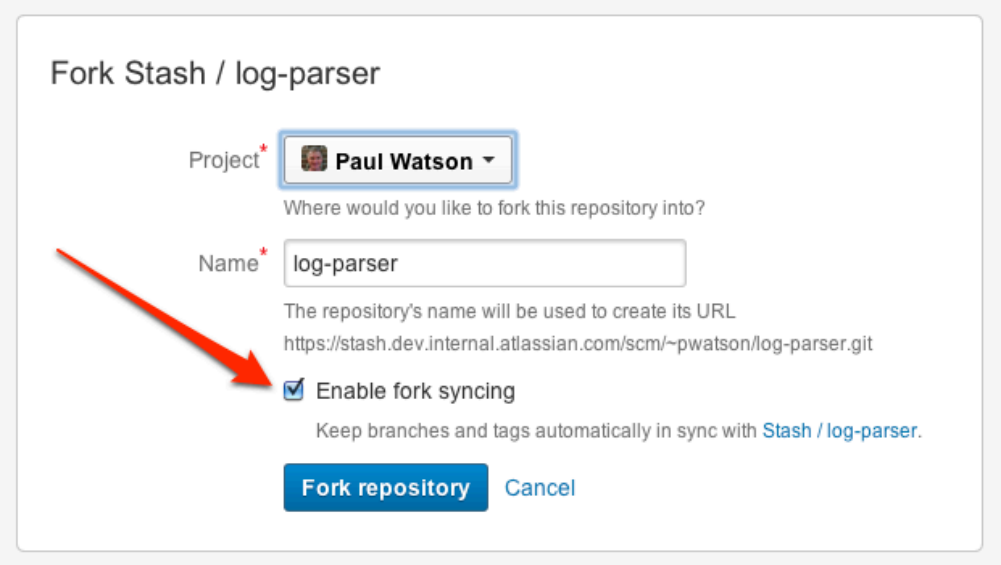
Note that syncing is about pulling recent upstream changes into your fork, whereas pull requests are about pushing your changes back to the upstream repository.

On this page:

- Enabling automatic fork syncing
- What gets synced?
- Manual synchronization strategies

Enabling automatic fork syncing

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



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

What gets synced?

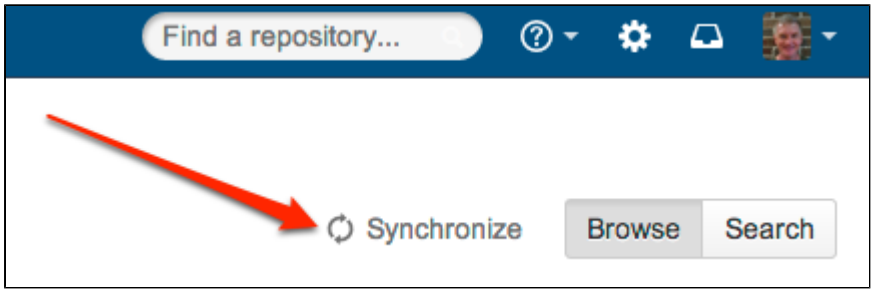
When performing automatic synchronization, Stash updates the fork as follows:

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

Manual syncing

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

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

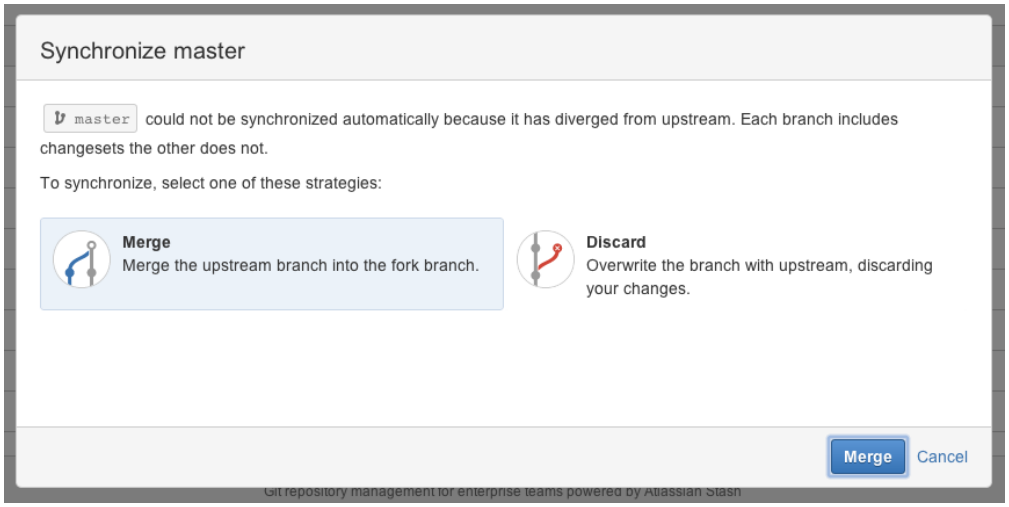


Manual synchronization strategies

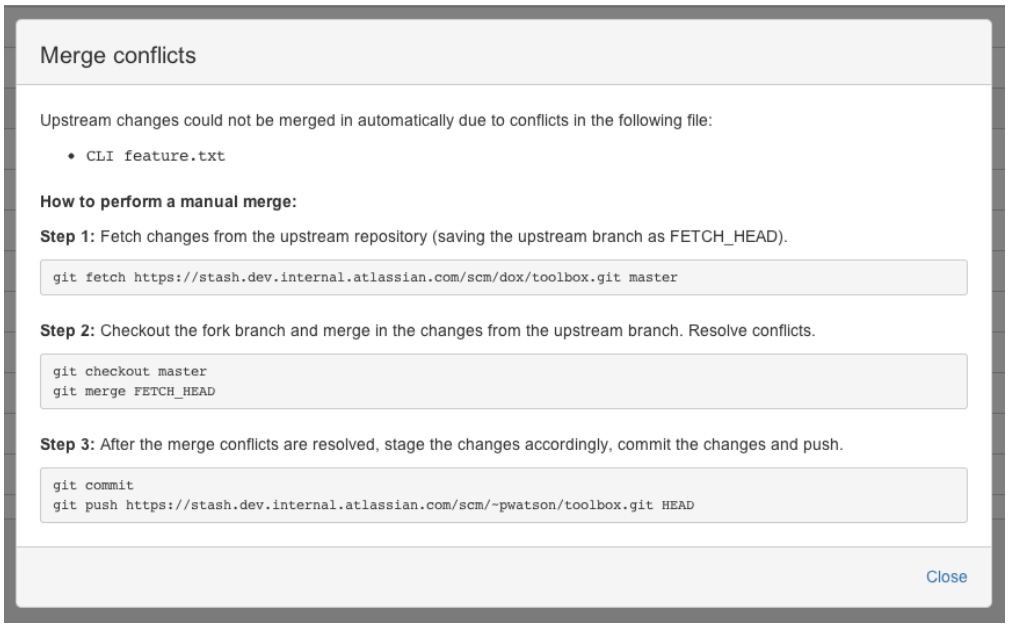
When you initiate a manual synchronization, Stash will ask you to choose one of the following synchronization strategies.

Merge strategy

Merge the upstream branch into the fork branch.



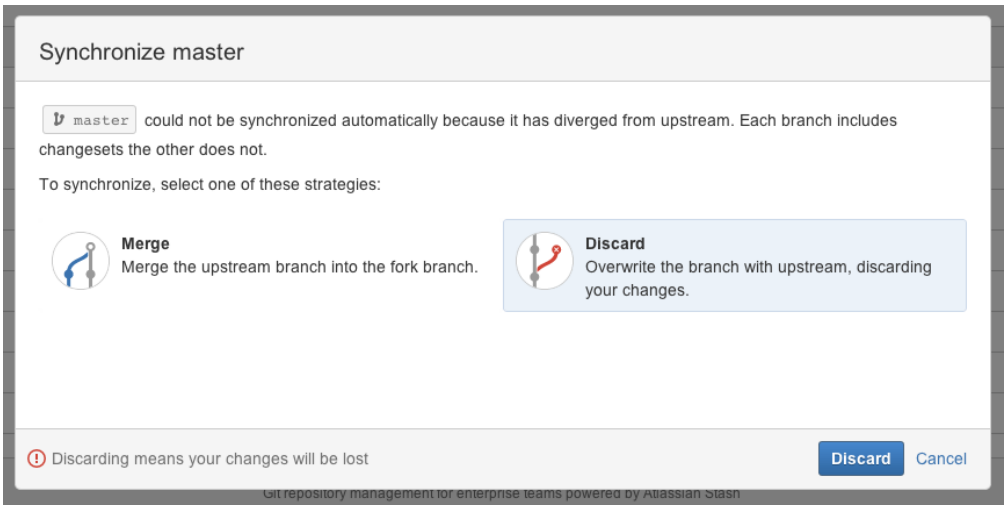
If Stash detects conflicts when trying to perform the merge it will offer hints on how to resolve those:



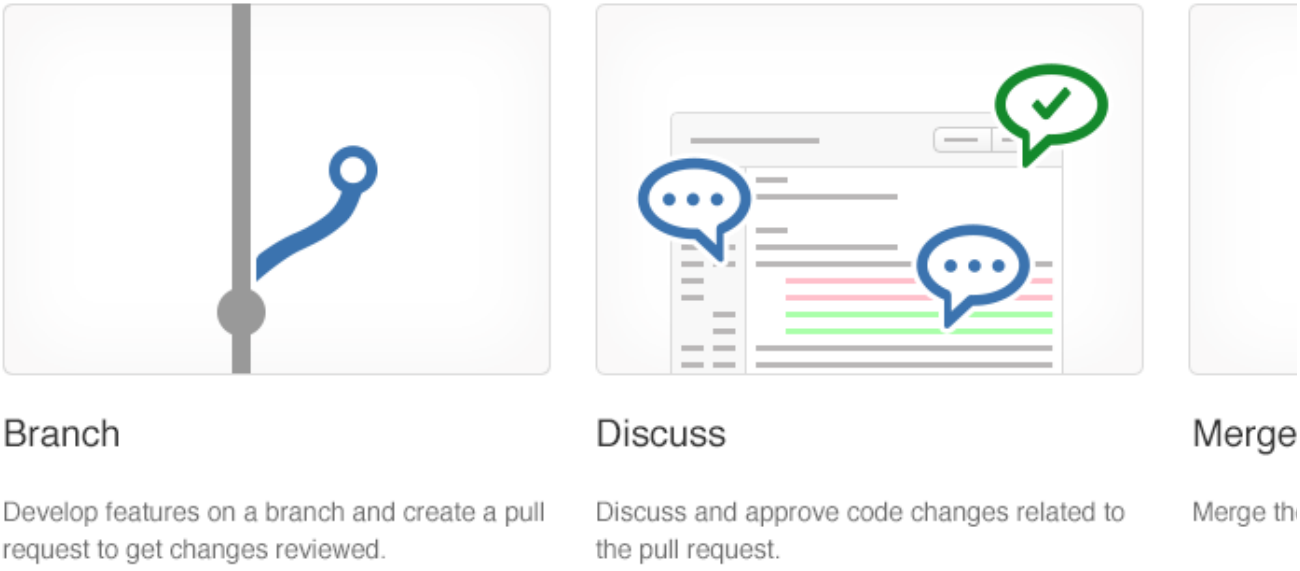
Once the merge is complete, your branch will have incorporated all the commits on the branch in the parent repository, but your branch will still be ahead of the parent (it has your changes on it). This means automatic synchronization for this branch will not occur until your changes are pushed to the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

Discard strategy

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



Using pull requests in Stash



Pull requests in Stash provide the team with a quick and easy way to review changes made on a branch, discuss those changes, and make further modifications before the branch is merged to master or your main development branch.

On this page:

- Creating a pull request
- Editing a pull request
- Discussing a pull request
- Merging a pull request
- Watching and notifications

Related pages:

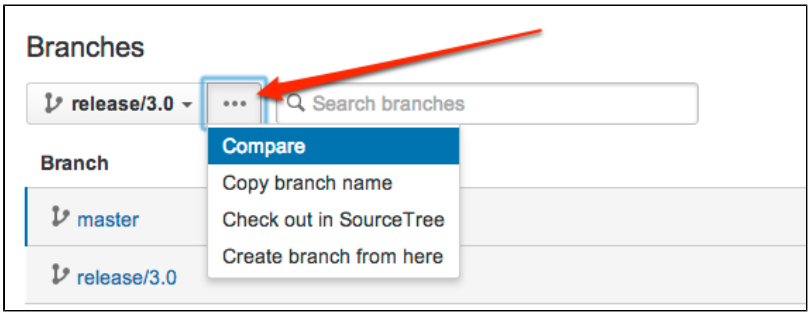
- Checks for merging pull requests

Creating a pull request

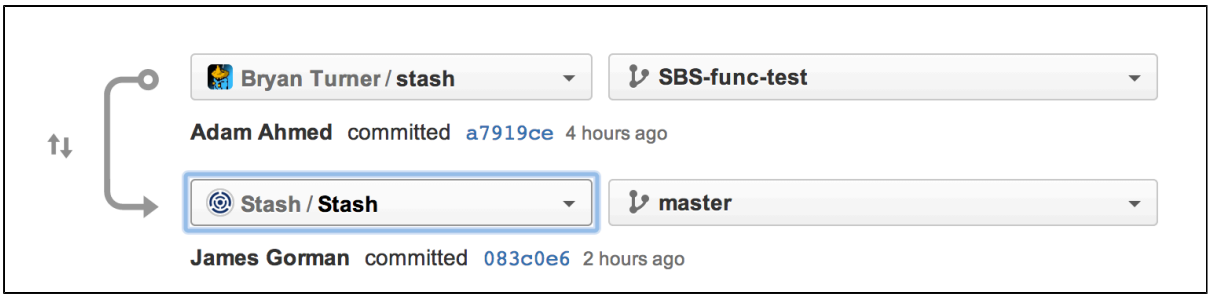
When you are ready to start a discussion about your code changes, simply create a pull request.

To create a pull request:

1. You've pushed your changes to Stash, right?
2. Now, go to the repository in Stash. Either click **Create pull request** in the sidebar, or choose **Compare** fr om the Actions menu (when on the Source, Commits or Branches pages):



3. Choose the source and destination branches. The source branch is where you made your code changes and the destination is the branch you want to merge to. The source and target branches may be located in different forks:

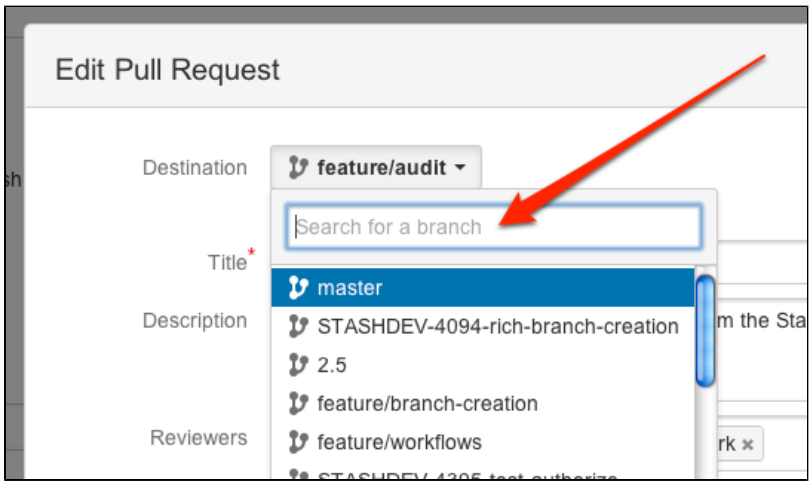


4. Use the Diff and Commits tabs (see below) to compare the source and destination branches, before creating the pull request.
5. Click either **Create pull request**, or **Continue**, and enter a title and description that will help people understand what your pull request is about. You can use mentions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

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

Editing a pull request

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



Discussing a pull request

The most important thing about a pull request is the discussion that it generates. With Stash, you can comment on the pull request as a whole, a particular file, and on specific lines of code in a file. Comment likes are a quick way of amplifying review feedback – effectively saying "also consider this person's feedback". Furthermore, you can attach a task to any comment, so actions identified during the review can be easily tracked and resolved. Read more about pull request tasks below.

To see all the pull requests for a repository, simply click **Pull requests** in the left-hand navigation panel (when viewing the repository).

To help you contribute to the discussion, Stash organises all the information about the pull request into 3 tabs: Overview, Diff and Commits:

Overview

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

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

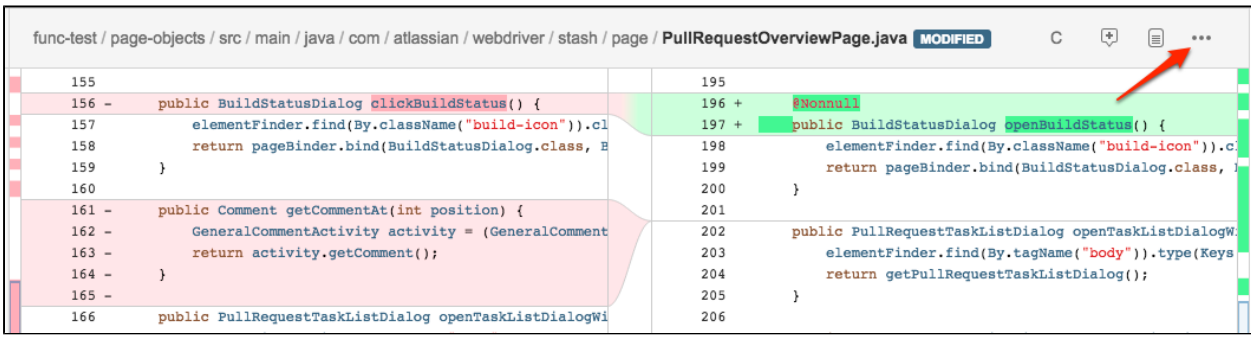
Diff

Diffs for Stash pull requests provide the following advantages:

- The diff highlights the changes that will result when the merge occurs, so you can see exactly what the effect of the merge will be.
- The file tree on the left colour-codes files that have been added, changed or deleted, so you can quickly see the files you may need to review.
- As you'd expect, the diff for a file shows which lines of code have been added, deleted or modified.
- You can comment on the whole file by clicking the icon on the right in the diff header.
- You can comment directly on a line of code right in the diff, by hovering over the line, clicking the icon at

the left, and entering your comment. Your comment will also appear in the activity.

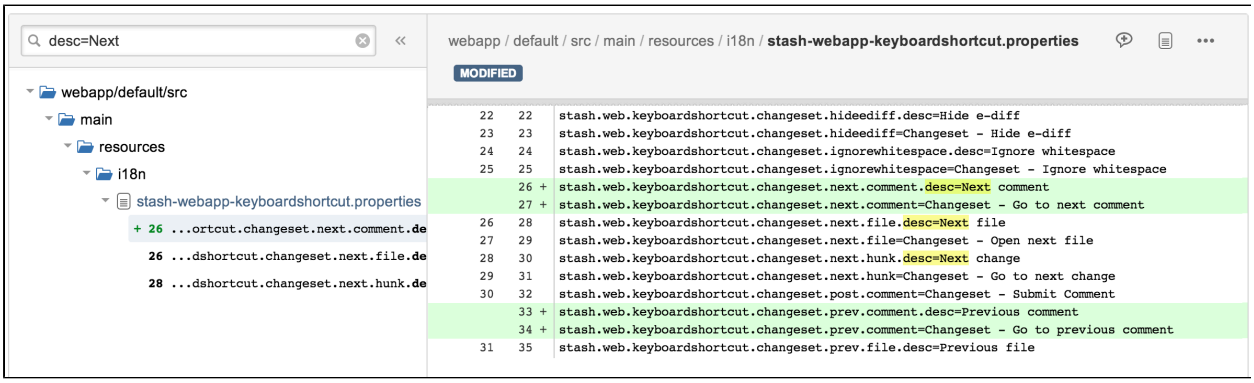
- Comments in the diff are threaded, to allow meaningful and contextual conversations about your code.
- The **Side-by-side diff** option from the Action menu (arrowed below) lets you easily compare the changes that will be merged. Use the N (next) and P (previous) keyboard shortcuts to move between hunks in a diff. Use Shift+N (next) and Shift+P (previous) to move between comments in a diff. The 'map' in each margin of the side-by-side diff provides a visual summary of the diff hunks, and indicates which part of the file you're currently viewing:



- Search across all the files in a pull request, when viewing a pull request diff.

  The search looks at the diff and surrounding code lines to provide context. The file tree on the left displays only the files returned by the search.

  Use the F keyboard shortcut to quickly access code search when viewing a diff in a pull request. The usual J (next) and K (previous) Stash shortcuts let you move between files in the search results. 'Esc' cancels the search.



Commits

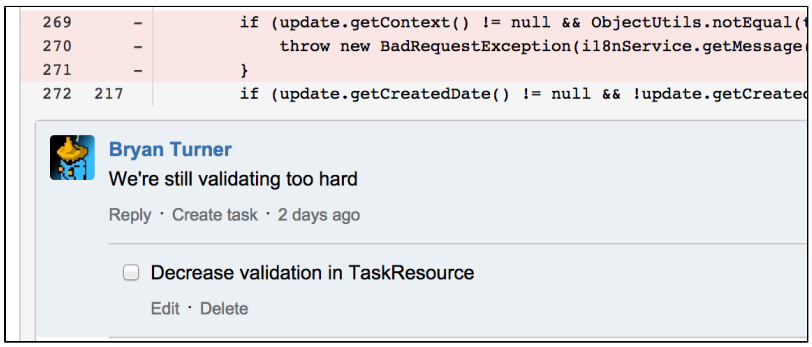
The **Commits** tab lists all the commits that will get merged (those that are greyed out have already been merged). Clicking through to a commit takes you out of the pull request context.

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

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

Pull request tasks

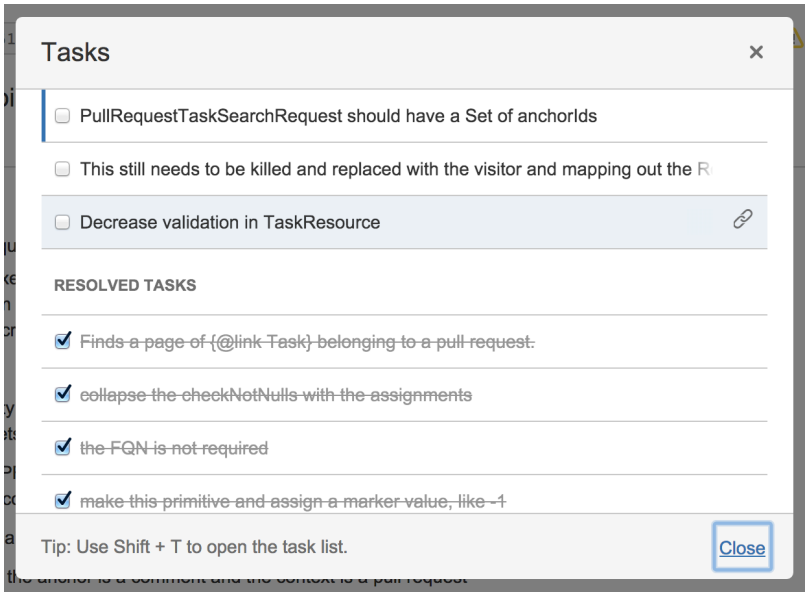
You can attach one or more tasks to any pull request comment, to track required work identified during a review.



Try highlighting some text in the comment, then clicking **Create task** – the task is automatically created and saved with that text.

When that task is done, simply check the box for the task.

To see all the unresolved tasks for a pull request, use Shift+T when viewing the pull request:



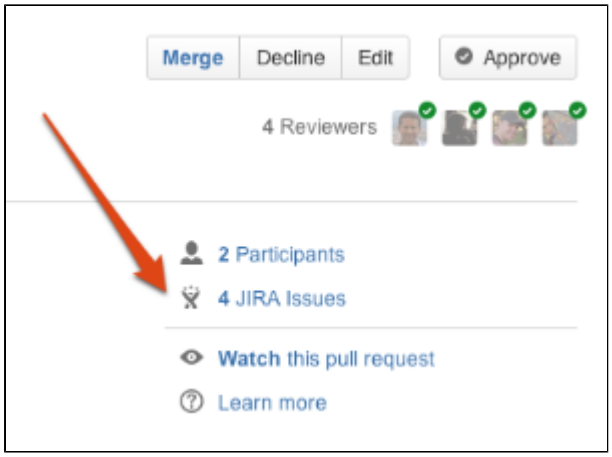
Click the 'link' icon for a task to see the task in the context of the comment and source code.

Note that:

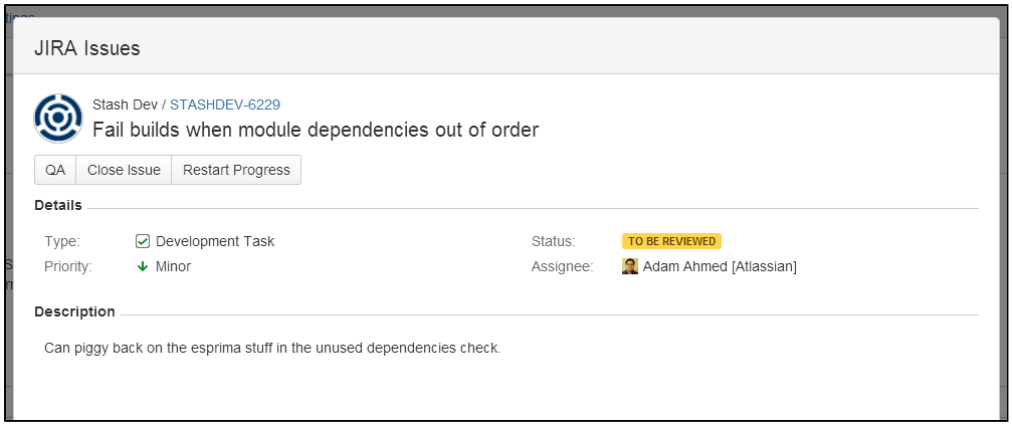
- Anyone with permission to browse a pull request can create a task on any comment, and can browse, resolve and reopen existing tasks in the pull request.
- Repository admins and pull request authors can edit and delete *any* task in the pull request. Reviewers and others can only edit or delete their *own* tasks.
- A Stash administrator can set a merge check that requires all tasks to be resolved before the pull request can be merged. See Checks for merging pull requests.

JIRA issues integration

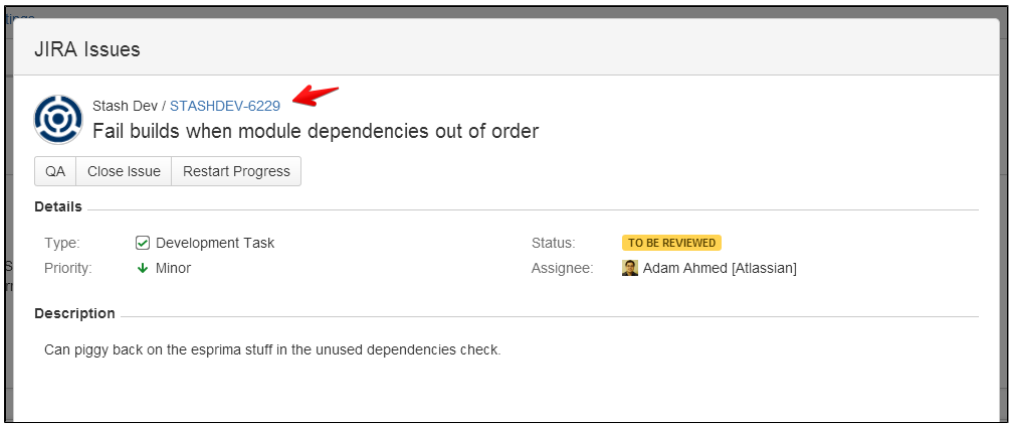
When Stash is integrated with JIRA you can see the related JIRA issues right in the pull request. You'll see either the JIRA issue key, when there is just one, or the number of related issues:



Click an issue key to see details of the issue, such as the description, status and assignee, without having to leave Stash:



Click **the issue key** to see the issue in JIRA. This allows reviewers to gain important insight into the task that is being worked on, by seeing the comments and attachments on the issue. This also gives access into JIRA, so you can easily keep issues updated.



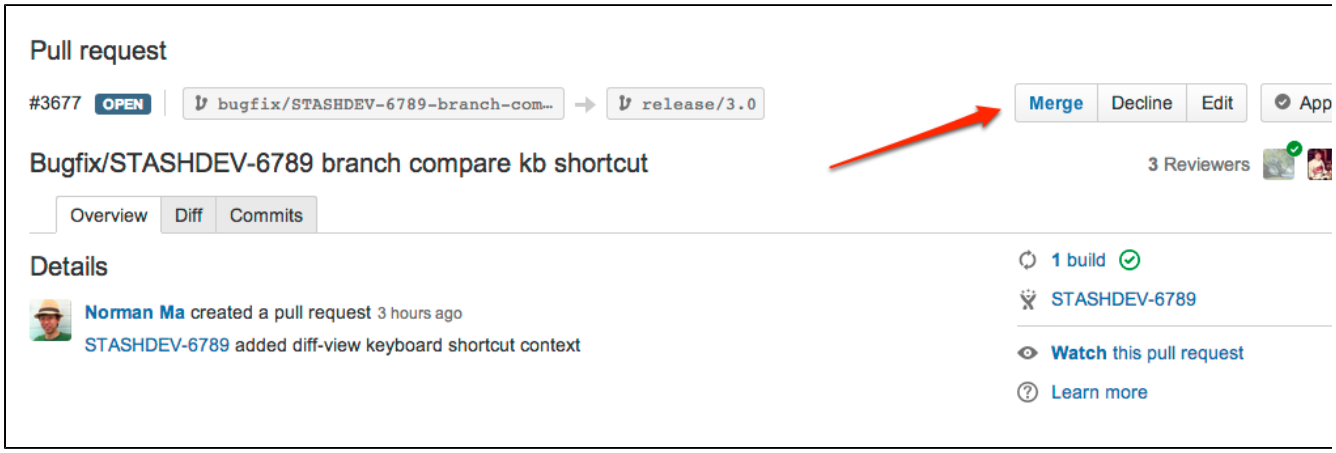
See JIRA integration for a description of all the integrations you get when Stash is linked with JIRA.

Merging a pull request

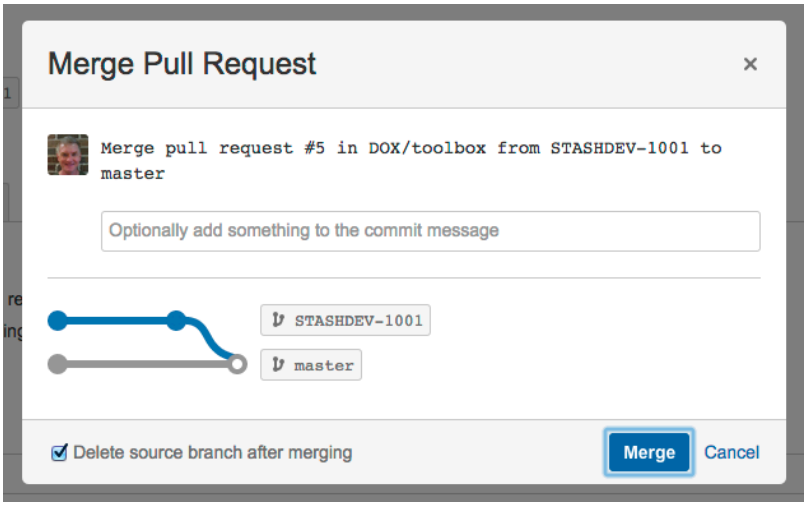
Once you are ready to merge a pull request, and when the reviewers have approved it, simply click **Merge** at the top right of the pull request view. You can merge a pull request if you have write (or admin) permission on the

project.

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



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



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

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

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

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

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

Watching and notifications

You automatically get added as a watcher of a pull request when you are added to the pull request as a

reviewer, or when you perform an action related to the pull request (such as adding a comment):

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

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

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

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

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

See Notifications for details.

Checks for merging pull requests

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

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

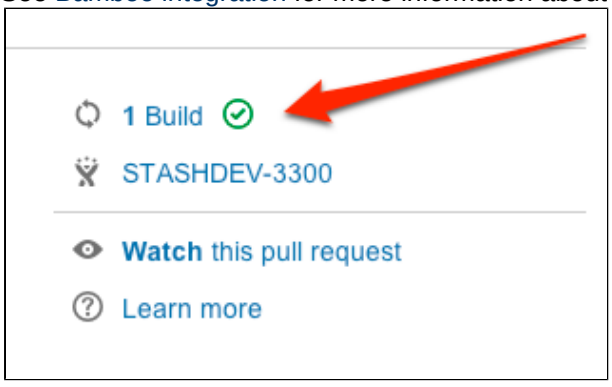
So, to set merge checks for pull requests, go to a repository in a project and choose **Settings** > **Pull requests.** Stash includes the merge checks described below, or you can write your own merge request check plugin.

Requires a minimum number of successful builds

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

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

See Bamboo integration for more information about integrating Stash with your build server.



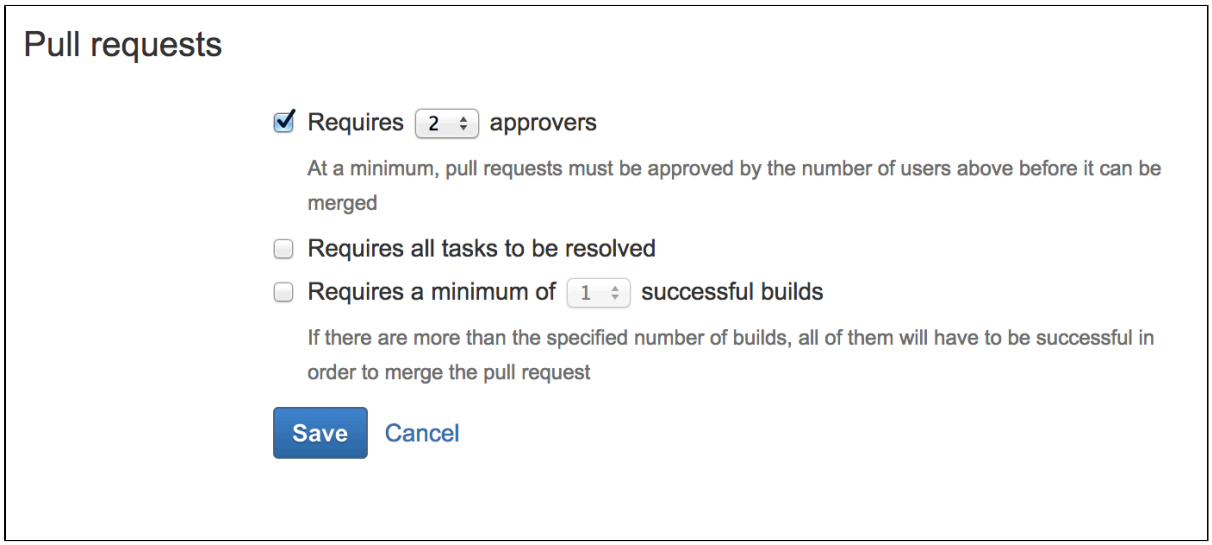
The number of builds, and the latest result, for the head commit of a branch.

Requires a minimum number of approvers

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

Requires all tasks to be resolved

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



Notifications

An email server must be configured in Stash for email notifications to be sent. See Setting up your mail server. Note that if the mail server fails, notifications will be dropped. See also HipChat notifications.

Pull request notifications

Stash sends email notifications to the watchers and reviewers of a pull request when the following events occur:

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

The pull request is reopened	**BATCHED** or **IMMEDIATE**

By default, email notifications are batched. However, in the following situations notifications are sent immediately:

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

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

You don't receive notifications for events you initiate yourself. See also Using pull requests in Stash.

Batched email notifications

Stash sends email notifications to:

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

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

By default, email notifications are batched. However:

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

Using 'mentions' to notify someone

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

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

Markdown syntax guide

By default, Stash uses Markdown as its markup language. You can use markdown in the following places:

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

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

Markdown syntax

The page below contains examples of Markdown syntax. For a full list of all the Markdown syntax, consult the official documentation on John Gruber's Daring Fireball site.

On this page:

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

Headings

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

Paragraphs

```
Each paragraph begins on a new line. Simply press <return> for a new line.

For example,
like this.

You'll need an empty line between a paragraph and any following markdown
construct,
such as an ordered or unordered list, for that to be rendered. Like this:

* Item 1
* Item 2
```

Character styles

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

Unordered list

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

Ordered list

```
1. Step 1
2. Step 2
3. Step 3
    a. Step 3a
    b. Step 3b
    c. Step 3c
```

List in list

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

Quotes or citations

```
Introducing my quote:

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

Inline code characters

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

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

Code blocks

```
Indent every line of the block by at least 4 spaces or 1 tab.

This is a normal paragraph:

    This is a code block.
    With multiple lines.

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

```
This is a code block
```

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

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

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

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

Links to external websites

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

[This link](http://example.net/) has no title attribute.
```

Linking issue keys to JIRA

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

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

Images

Inline image syntax looks like this:

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

For example:

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

Reference image links look like this:

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

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

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

For example:

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

Tables

```
| Day     | Meal    | Price |
| --------|---------|-------|
| Monday  | pasta   | $6    |
| Tuesday | chicken | $8    |
```

Backslash escapes

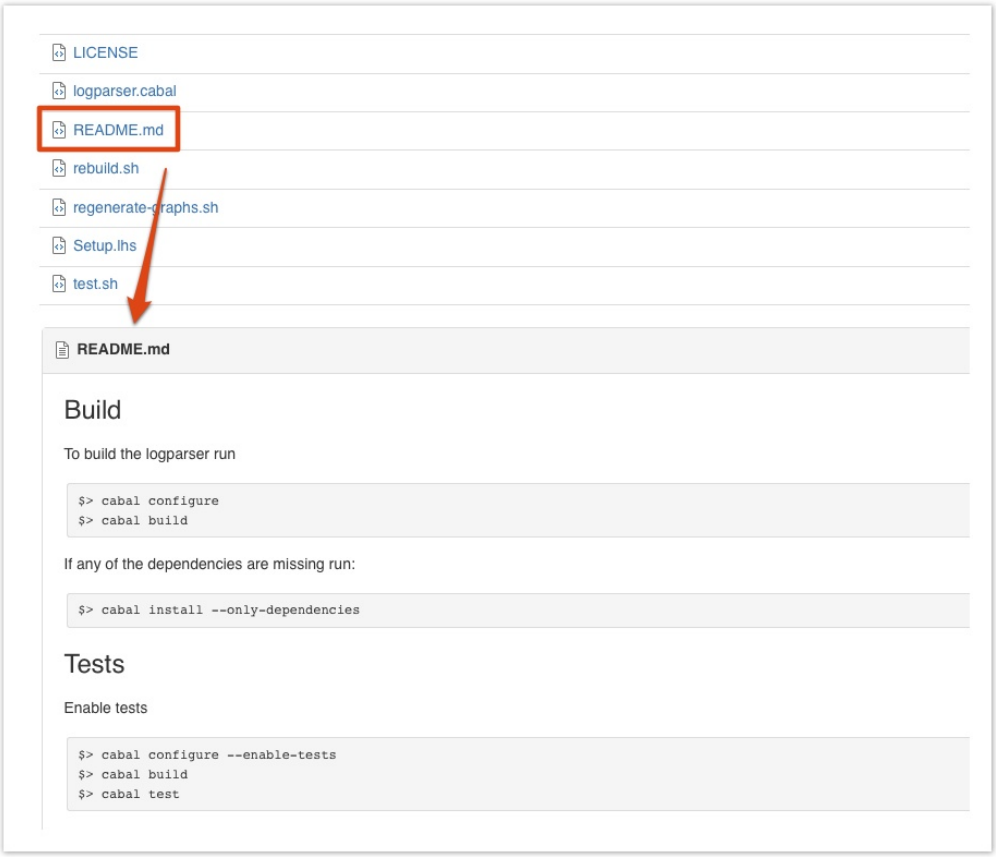
Certain characters can be escaped with a preceding backslash to preserve the literal display of a character instead of its special Markdown meaning. This applies to the following characters:

```
\  backslash
`  backtick
*  asterisk
_  underscore
{} curly braces
[] square brackets
() parentheses
#  hash mark
>  greater than
```

```
+  plus sign
-  minus sign (hyphen)
.  dot
!  exclamation mark
```

README files

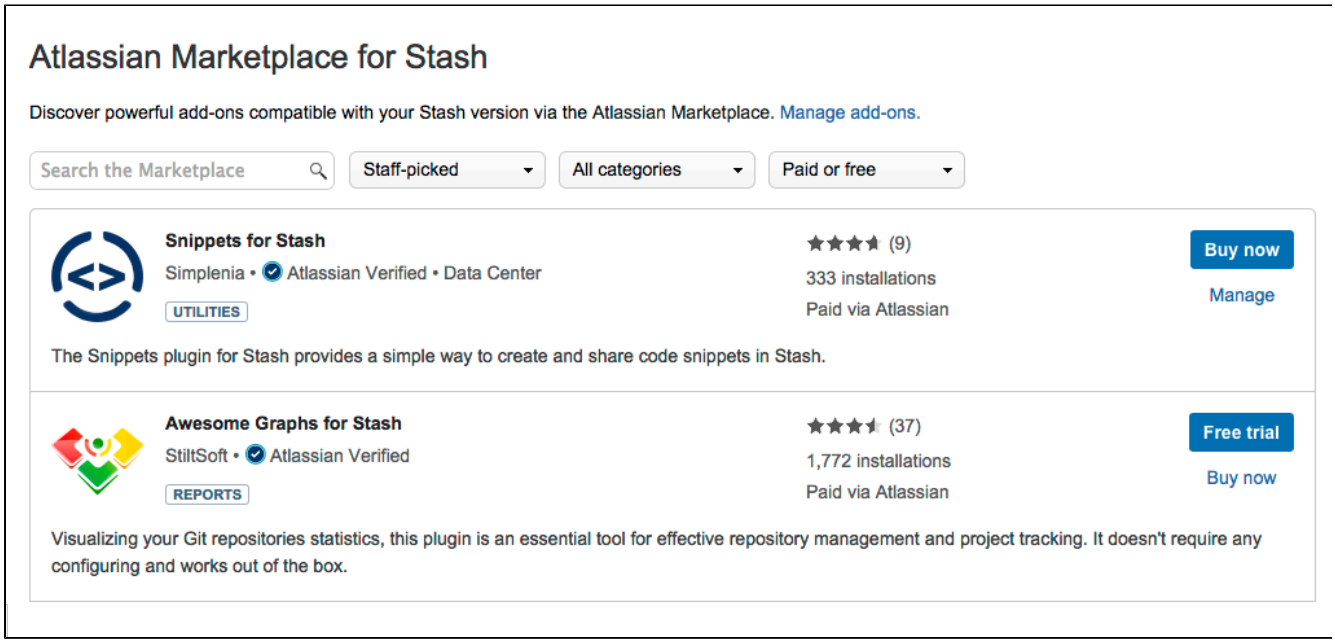
From Stash 1.3, you can document a project right in the repository by creating .md or .txt files. If the ReadMe has the .md extension, any Markdown it contains gets rendered straight to the screen when viewed from the file list of the repository.



Requesting add-ons

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

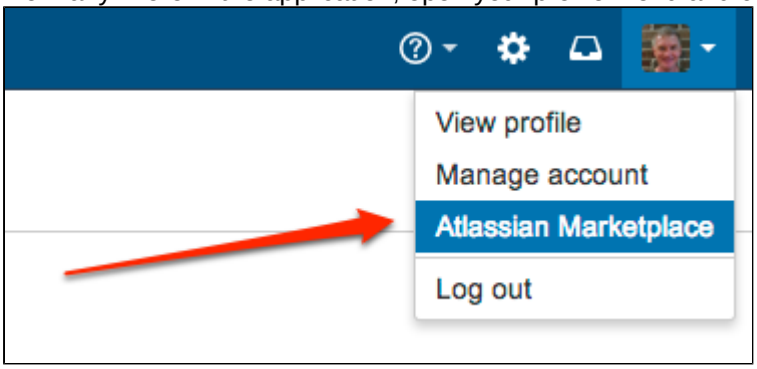
The 'Atlassian Marketplace for Stash' page provides an integrated view of the Atlassian Marketplace from within your Stash instance. The page offers the same features as the Marketplace website, such as searching and category filtering, but tailors the browsing experience to Stash.



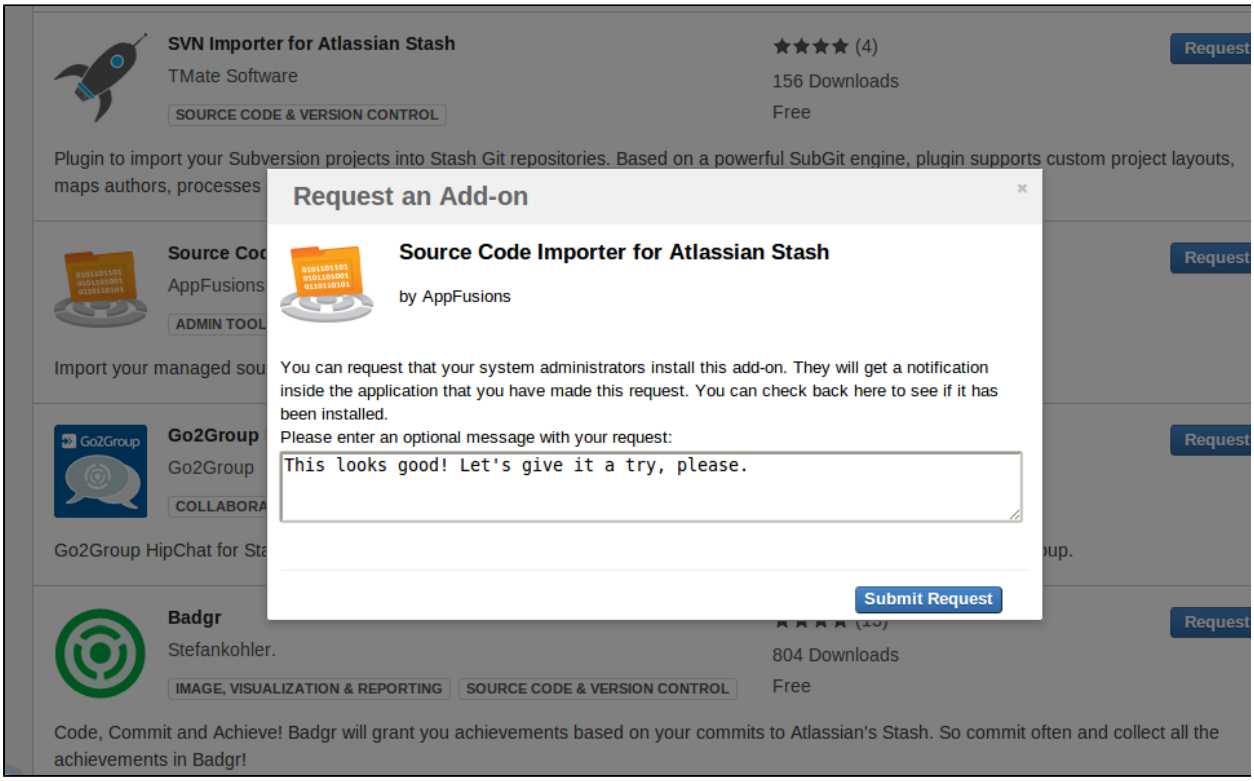
This in-product view of the Marketplace gives day-to-day Stash users, not just administrators, an easy way to discover add-ons that can help them get work done. When you find an add-on of interest, you can submit a request to your administrator for the add-on with just a few clicks.

Submitting an add-on request

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

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


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



5. Click **Submit Request** when done.
6. Click **Close** to dismiss the 'Success!' message dialog box.

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

Updating an add-on request

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

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

Integrating Stash with Atlassian applications

When you integrate Stash with Atlassian applications you get the following benefits:

Application	Integration feature	Compatibility	
JIRA	Related branches, commits and pull requests are all summarized in the Development panel in a JIRA issue.	JIRA 6.2+	Stash 2.10+
JIRA	Create Git branches from within JIRA and JIRA Agile.	JIRA 6.1+	Stash 2.8+
JIRA Agile	Transition JIRA issues from within Stash.	JIRA 5.0+	Stash 2.7+

	See the JIRA issues related to Stash commits and pull requests.	JIRA 5.0+	Stash 2.1+
	See all the code changes commited for the issue (on the JIRA Source tab). Click through to see a changed file, or the full commit , in Stash.	JIRA 5.0.4+	Plugin version bundled in JIRA
		JIRA 5.0–5.0.3	JIRA FishEye/Stash Plugin 5.0.4.1
		JIRA 4.4.x	JIRA FishEye/Stash Plugin 3.4.12
		JIRA 4.3.x	JIRA FishEye/Stash Plugin 3.1.8
HipChat	When Stash is integrated with HipChat, notifications are sent to a HipChat room whenever someone pushes to a repository in Stash.		Stash 2.2+
Bamboo	Bamboo responds to repository events published by Stash to: • Trigger a plan build when a developer pushes to the connected repository. • Create or delete plan branches when a developer creates or removes a branch in the connected repository. When you link a build plan to a Stash repository, build notifications are automatically enabled. See Bamboo integration.	Bamboo 5.6+	Stash 3.1+
	See the latest build status for a commit when viewing Stash commits and pull requests.	Bamboo 4.4+	Stash 2.1+
SourceTree	When you have SourceTree installed, you can: • clone a Stash repository using SourceTree. • check out a branch in SourceTree, when viewing files, commits or branches in a Stash repository.	SourceTree 1.7+	Stash 2.7+

Crowd	When Stash is integrated with Crowd, you can: • use Crowd for user and group management, and for authentication.		

JIRA integration

When Stash is integrated with Atlassian JIRA, you and your team get all these benefits:

- See all the related commits, branches and pull requests in a JIRA issue.
- Create Git branches from within JIRA and JIRA Agile.
- Transition JIRA issues automatically.
- Transition JIRA issues from within Stash.
- Use JIRA issue keys in Stash markdown.
- See the details for JIRA issues in Stash.
- See the JIRA issues related to Stash commits and pull requests.
- Check commits, branches and pull requests for an entire version within JIRA.

You can also use JIRA for delegated management of your Stash users. See External user directories.

Your Stash administrator needs to set up linking with JIRA before you'll see these work.

Check development progress of a version in JIRA

STASH 2.10+
JIRA 6.4+

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

.

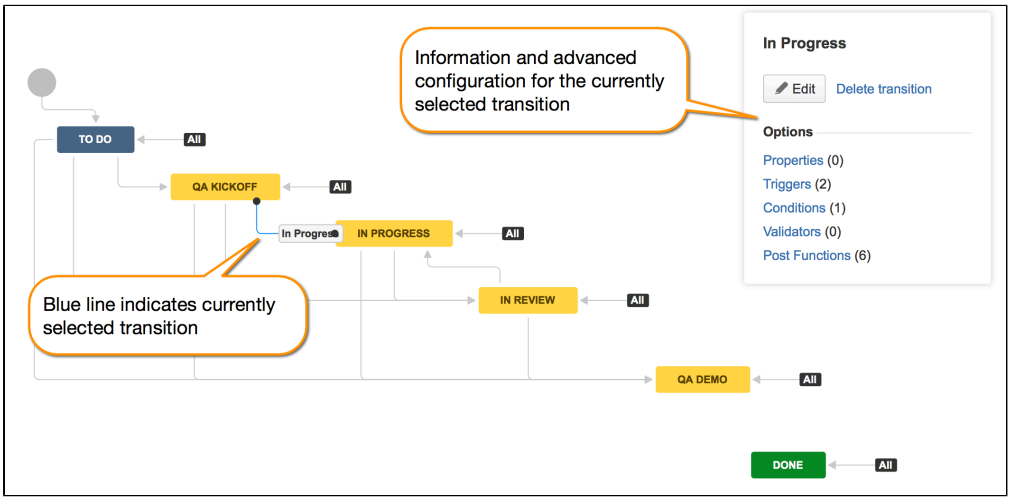
When you are ready, you can also release the version from the Release Hub, which marks the version as complete, moves incomplete issues to other versions, and triggers release builds (if JIRA is connected to Bamboo).

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

Transition JIRA issues automatically

STASH 3.2+
JIRA 6.3.3+

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



The events available in Stash are:

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

Stash events are published by default. We recommend that you use the latest version of JIRA to ensure that duplicate events are handled correctly. JIRA automatically removes duplicate commit events in JIRA 6.3.3+ and duplicate branch creation events in JIRA 6.3.11+.

See all related branches, commits and pull requests in a JIRA issue

STASH 2.10+
JIRA 6.2+

Get visibility into the Stash branches, commits and pull requests related to work on a JIRA issue, right in the context of the issue in JIRA (and JIRA Agile).

.

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

Create Git branches from within JIRA and JIRA Agile

STASH 2.8+
JIRA 6.1+

You can start creating a branch from a JIRA issue. This gives you a faster workflow from picking an issue to starting coding.

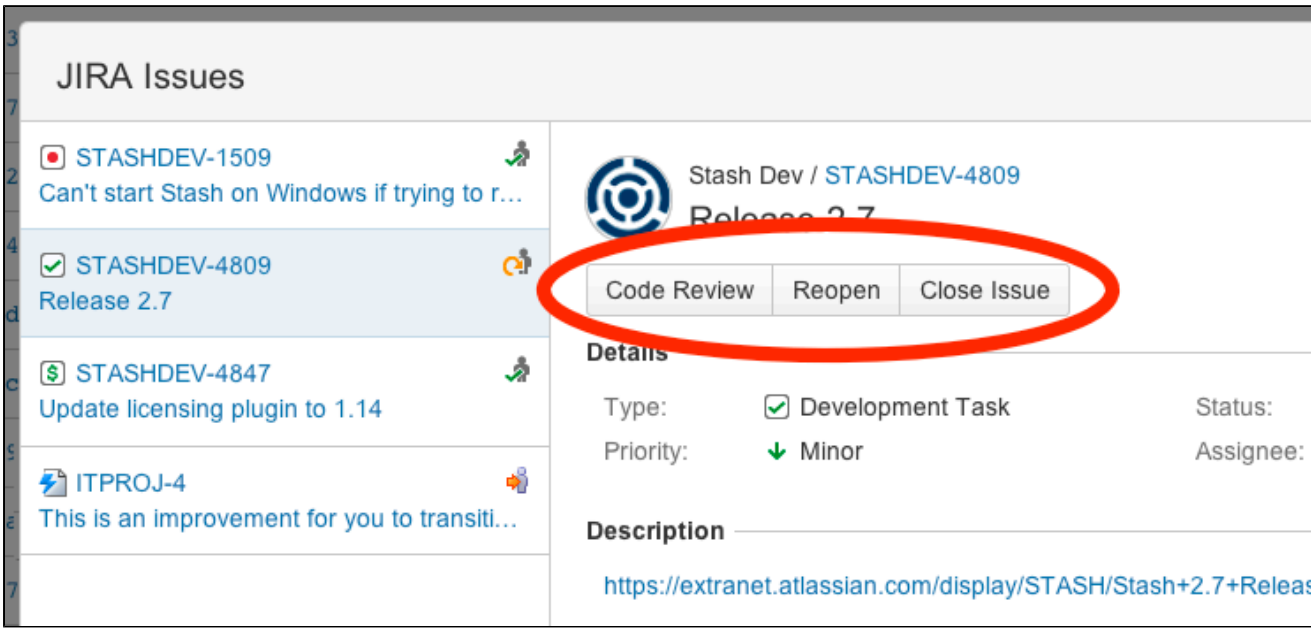
.

Stash will suggest the branch type and branch name, based on the JIRA issue type and summary – you can change these, of course.

Transition JIRA issues from within Stash

STASH 2.8+
JIRA 5.0+

You can easily transition a JIRA issue from within Stash. For example, when creating a pull request you may want to transition the issue into review. Click on a linked JIRA issue anywhere in Stash to see a dialog with the available workflow steps:



Click on a step and complete the fields as required. If there are custom fields that are unsupported by Stash, just click **Edit this field in JIRA** to transition the issue directly in JIRA.

See issues from multiple instances of JIRA

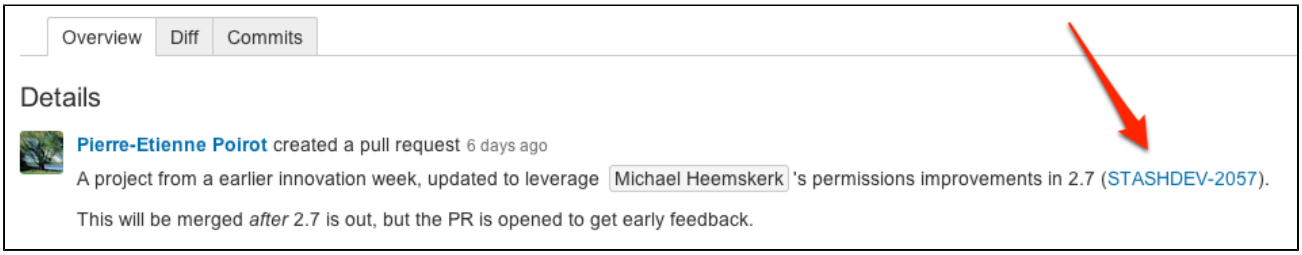
STASH 2.7+

Stash can link to more than one JIRA server at a time, so different teams can work with their own projects in different JIRA instances, or a single team can link to issues across multiple JIRA servers. Read more about linking Stash with JIRA.

Use JIRA issue keys in markdown

STASH 2.7+

When you mention a JIRA issue key in Stash, for example in a pull request description or a comment, the key gets automatically linked:

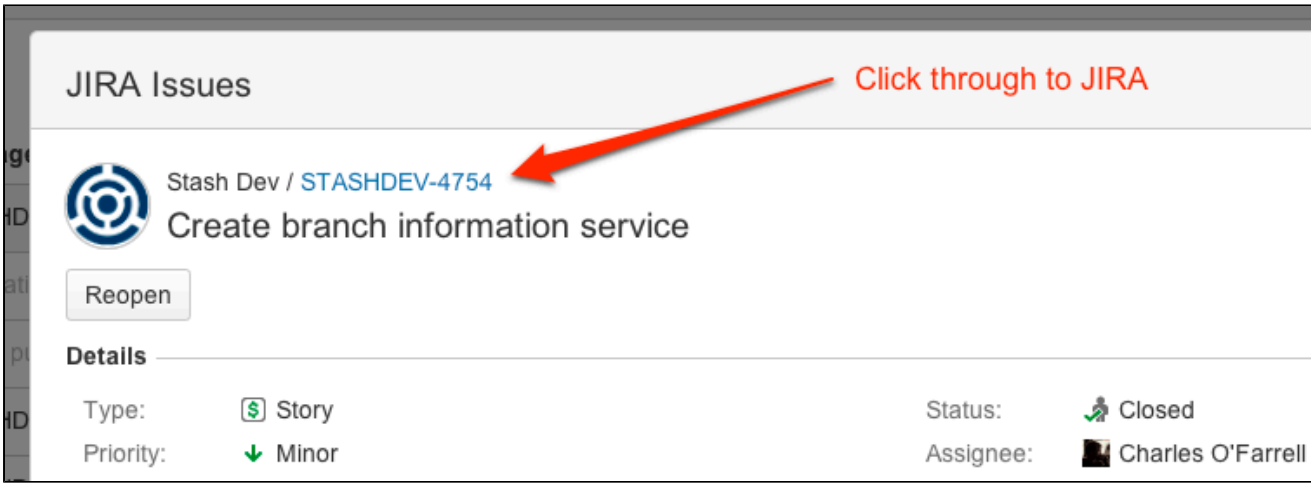


Click on the linked key to see details for the issue.

See the details for JIRA issues

STASH 2.1+
JIRA 5.0+

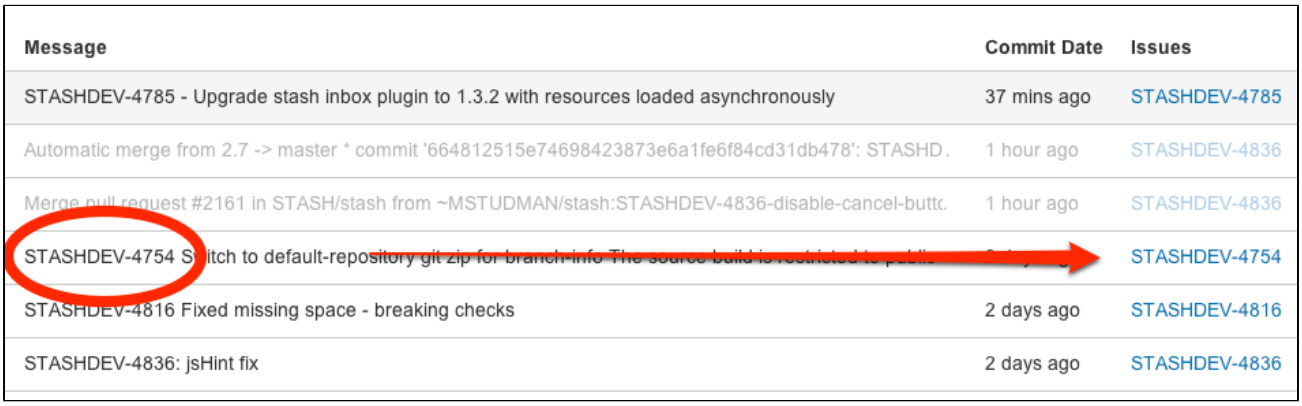
Click a linked issue key anywhere in Stash to see the details of that issue in a dialog. And you can just click the issue key at the top of the dialog to go straight to the issue in JIRA:



See the JIRA issues related to commits and pull requests

STASH 2.1+
JIRA 5.0+

Stash recognises JIRA issue keys in commit messages, and displays the keys as links on the Commits tabs for both the repository and pull requests:

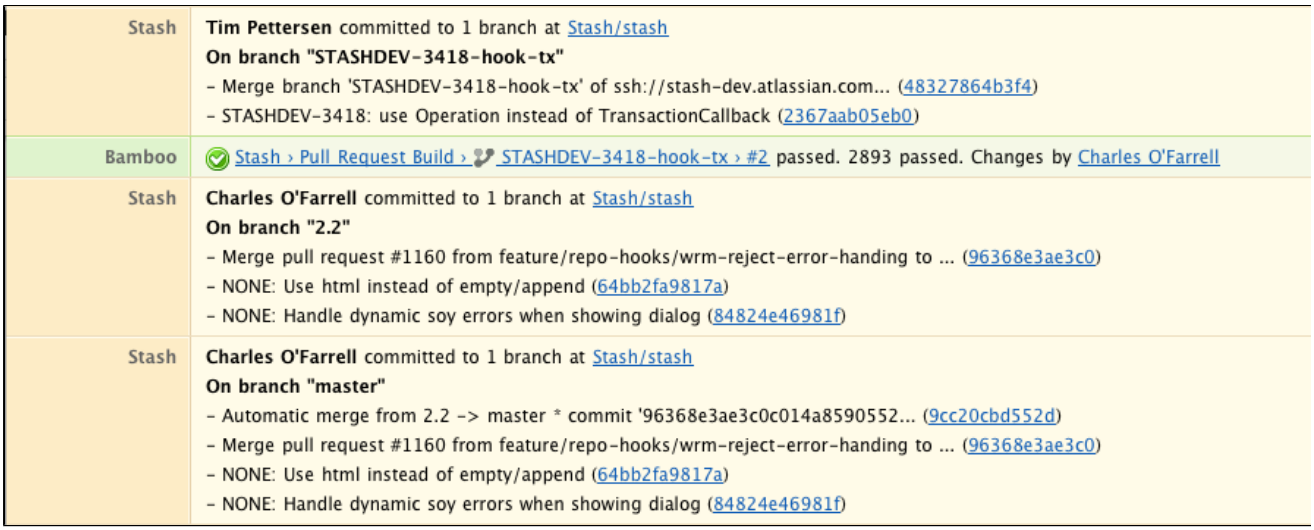


Click on the linked key to see details for the issue.

HipChat notifications

Stash can send an IM notification to a HipChat room whenever someone pushes to a repository. HipChat notifications are implemented as a post receive hook in Stash, and are available to all repos in Stash.

Here's an example of what you might see, from our own HipChat room:



On this page:

- Configure Stash to send notifications to your HipChat server
- Configure HipChat notifications for a repository

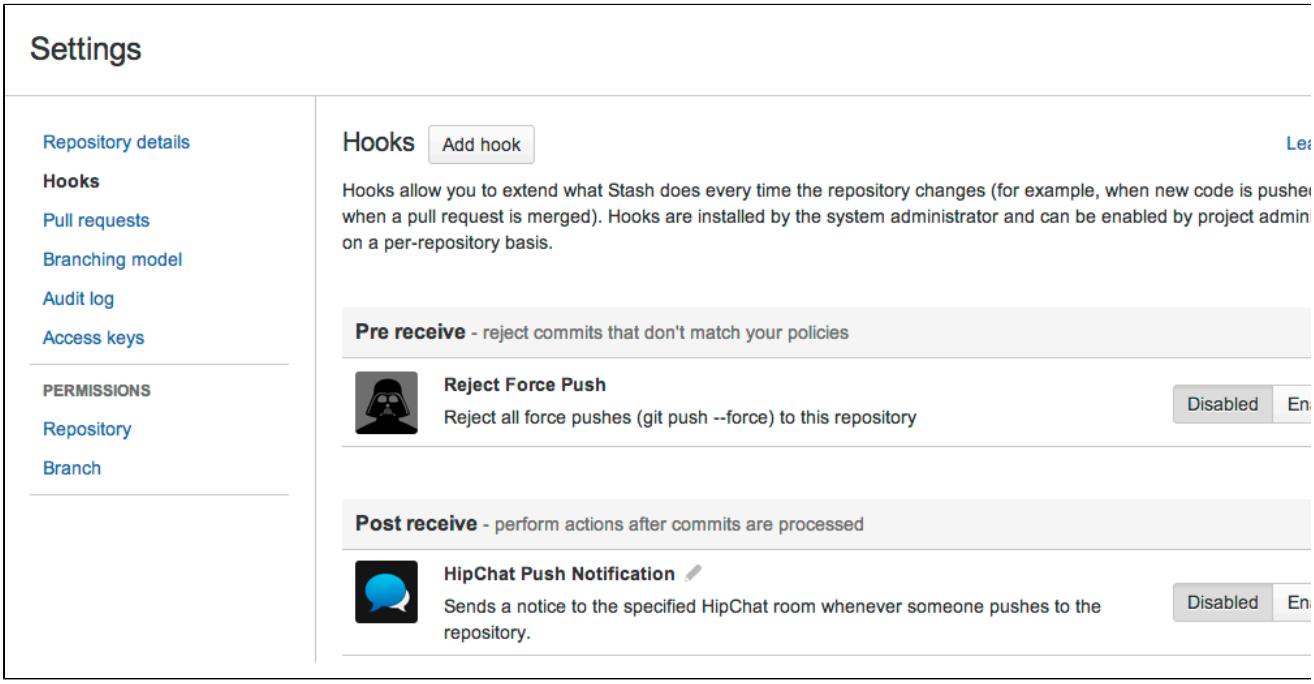
Configure Stash to send notifications to your HipChat server

A Stash admin can configure Stash to send notifications to rooms whether they're hosted by Atlassian on hipchat.com, or by your own organization's HipChat Server instance.

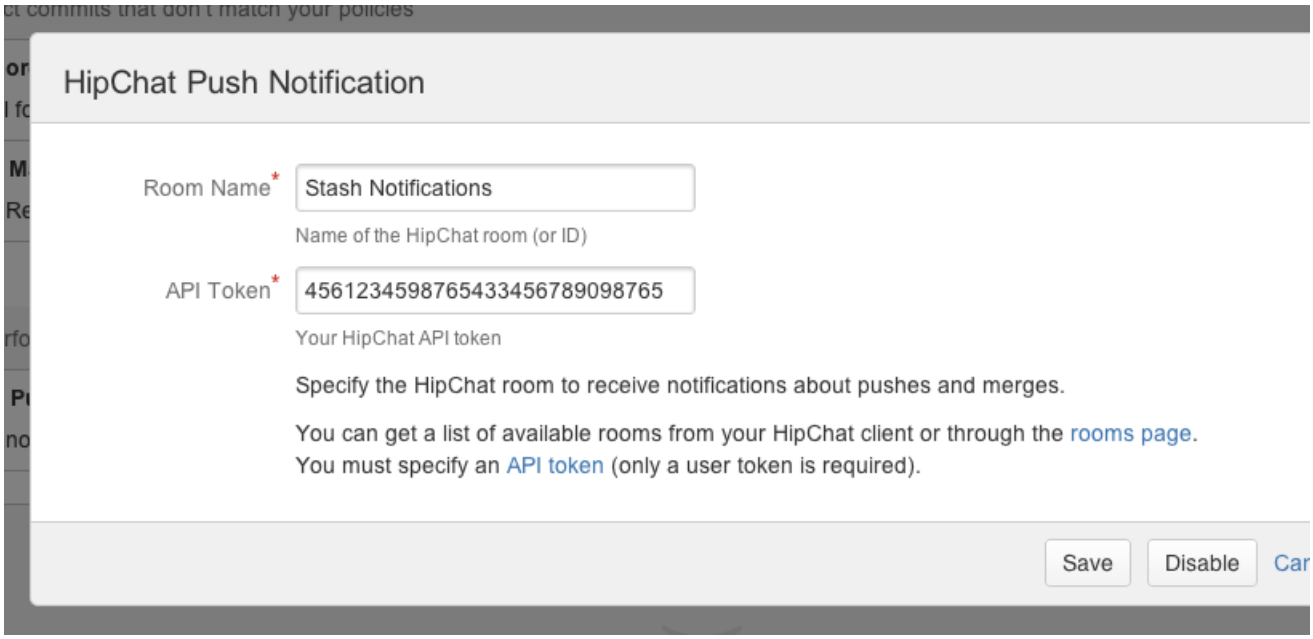
To change the setting for HipChat Server, you just need to tell Stash the fully qualified domain name (FQDN). Go to **Server settings** in the Stash admin area, and scroll down to 'HipChat integration':

Configure HipChat notifications for a repository

You can enable and disable HipChat notifications for a particular repo by going to **Settings** > **Hooks** for the repo:



Click the edit icon for 'HipChat Push Notifications' to specify the name of the HipChat room and the API token:



Talk to your HipChat administrator to get the API token for the HipChat server.

Bamboo integration

When you integrate Stash with Atlassian's Bamboo build and deployment server, commit, branch, build and deployment information is shared for users of both applications.

On this page:

- Benefits of integration
- Configuration

Benefits of integration

When Bamboo (versions 5.6 and later) and Stash (versions 3.1 and later) are integrated, you and your team get all the following advantages:

Stash tells Bamboo when to build

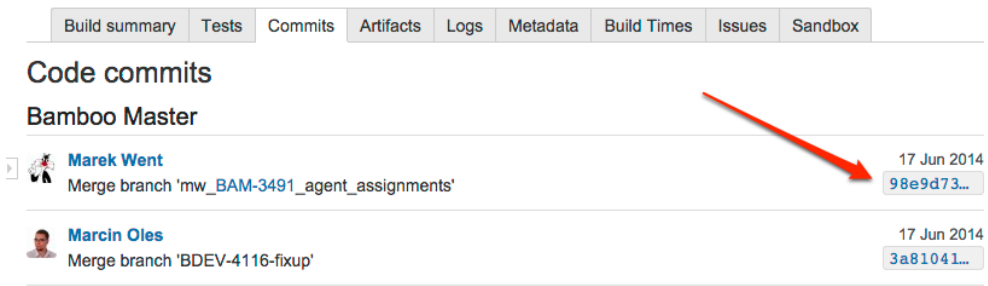
- When a developer pushes to a repository the build is automatically started.

Stash tells Bamboo when to update plan branches to match changes in repository branches

- When a developer pushes a new branch to a repository a branch plan is automatically created.
- When a developer deletes a branch in a repository, the branch plan is automatically deleted or disabled.

Stash commits are displayed in the relevant Bamboo builds

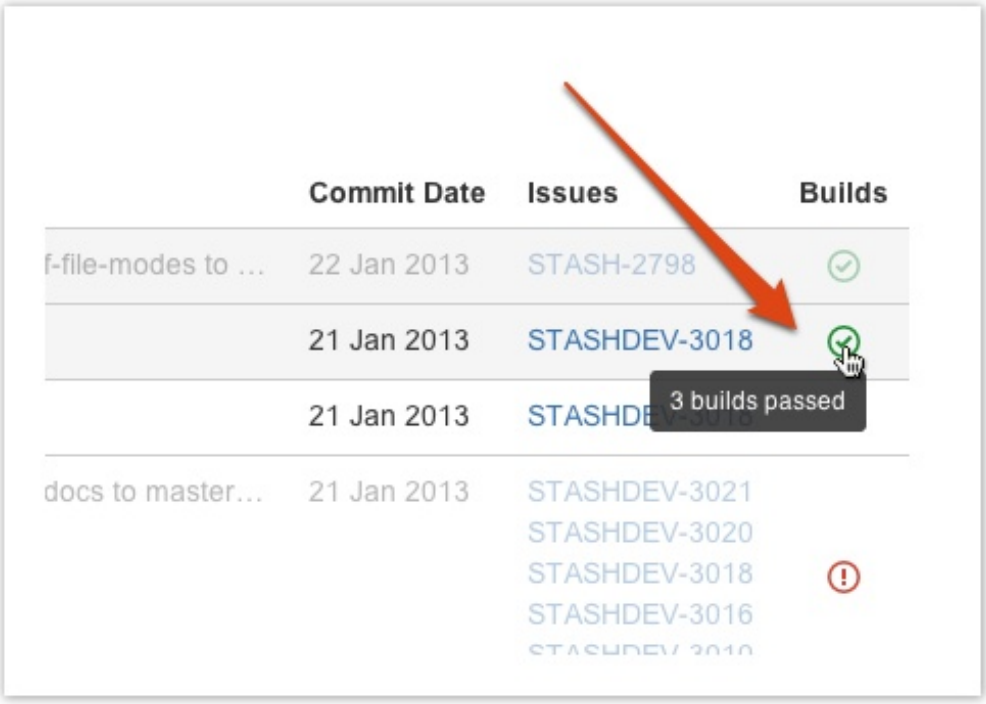
- In Bamboo, you can view all of the commits involved in the build, allowing you to accurately track changes:



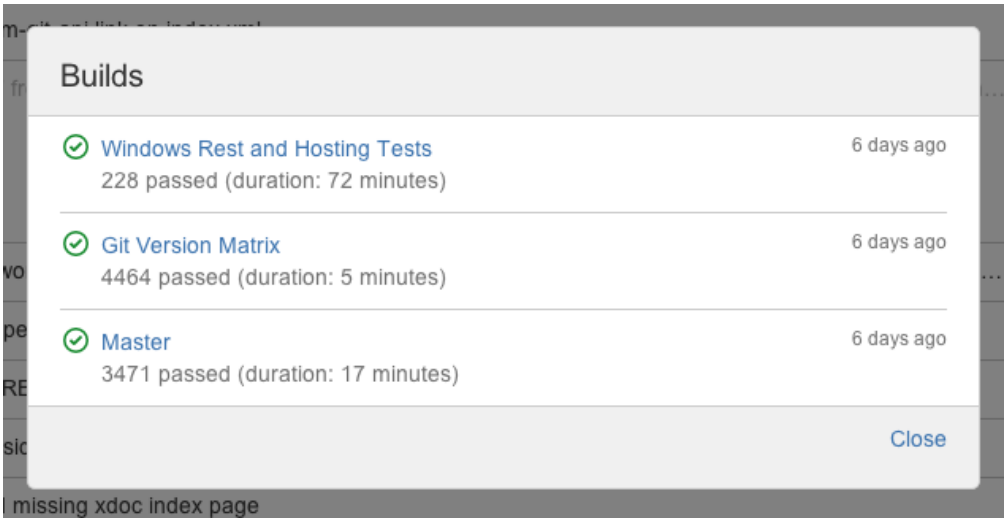
- Simply click on a changeset to go to Stash, where you can see the commit diff for all of the files that are part of the build.

Bamboo notifies Stash automatically about build results

- Build notifications are automatically enabled when you link a build plan to a Stash repository.
- Notifications are sent to all linked Stash servers.
- You see the build results status for a commit when viewing any commit or pull request in Stash, so you can easily check the build status of a branch when deciding whether to merge changes:



- Click a build status icon in Stash to see further details:



Stash displays the overall status of the build results. The status is 'passed' if all the different builds (for example, unit tests, functional tests, deploy to staging) have succeeded, and 'failed' if at least one run failed for any of those.

For example, when viewing the Commits tab for a Stash project, you will see icons that indicate the status of the latest build results. The red 'fail' icon is displayed if there is at least one failed build run for the commit.

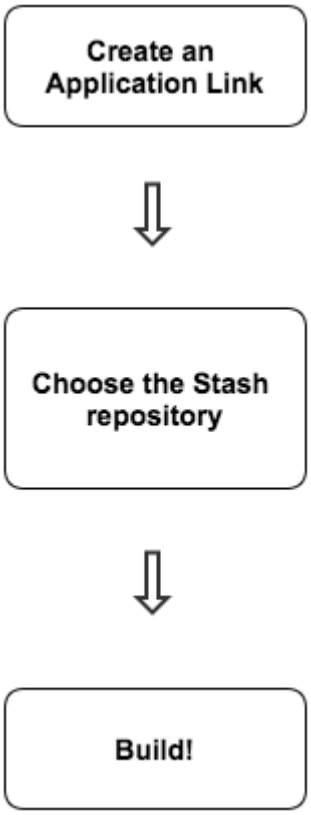
Note that the legacy Stash notification type is deprecated – it is still available in Bamboo 5.6 but will be removed in Bamboo 5.7.

Configuration

There are just a few simple configuration steps to get the integrations described above with Bamboo (versions 5.6 and later) and

Stash (versions 3.1 and later).

Bamboo will be automatically configured to respond to repository events published by Stash, and to notify Stash about build results – you don't have to configure repository polling for new commits anymore in Bamboo, or set up dedicated web hooks in your Stash instance.



1. Create an Application Link

You only need to do this once for each pair of Stash and Bamboo instances.

See Linking to another application.

Once linked, all the Stash repositories are available to your plans in Bamboo.

2. Choose the Stash repository for the Bamboo plan

Create a build plan (if necessary) and specify the repository in the plan (or job) configuration.

See Stash for more information about using Stash source repositories in Bamboo.

3. Build!

Having trouble integrating your Atlassian products with Application Links?
We've developed a guide to troubleshooting Application Links, to help you out. Take a look at it if you need help getting around any errors or roadblocks.

You can also use the Stash Rest API to automatically publish build status from Bamboo, Jenkins or any other build tool to Stash. See the Stash developer documentation to do with updating build status.

Administering Stash

Administration actions that can be performed from the Stash Administration user interface (click the 'cog' icon in the Stash header):

- Supported platforms
- Users and groups
- External user directories
  - Connecting Stash to an existing LDAP directory
  - Connecting Stash to JIRA for user management
  - Delegating Stash authentication to an LDAP directory
  - Connecting Stash to Crowd
- Global permissions
- Setting up your mail server
- Linking Stash with JIRA
- Connecting Stash to an external database
- Migrating Stash to another server
- Specifying the base URL for Stash
- Configuring the application navigator
- Managing add-ons
- Audit logging in Stash

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

- Running the Stash installer
- Automated setup for Stash
- Starting and stopping Stash
- Install Stash from an archive file
- Running Stash as a Linux service
- Running Stash as a Windows service
- Stash config properties
- Proxying and securing Stash
- Enabling SSH access to Git repositories in Stash
- Using diff transcoding in Stash
- Changing the port that Stash listens on
- Moving Stash to a different context path
- Running Stash with a dedicated user
- Stash debug logging
- Data recovery and backups
- Lockout recovery process
- Scaling Stash
- High availability for Stash
- Clustering with Stash Data Center
- Enabling JMX counters for performance monitoring
- Getting started with Stash and AWS
- Disabling HTTP(S) access to Git repositories in Stash

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:

- Even after users have been added to the Stash user directory, they will not be able to log in to Stash until

they have been given global access permissions.
- Permissions can also be applied separately at the level of projects, repositories and branches.

On this page:

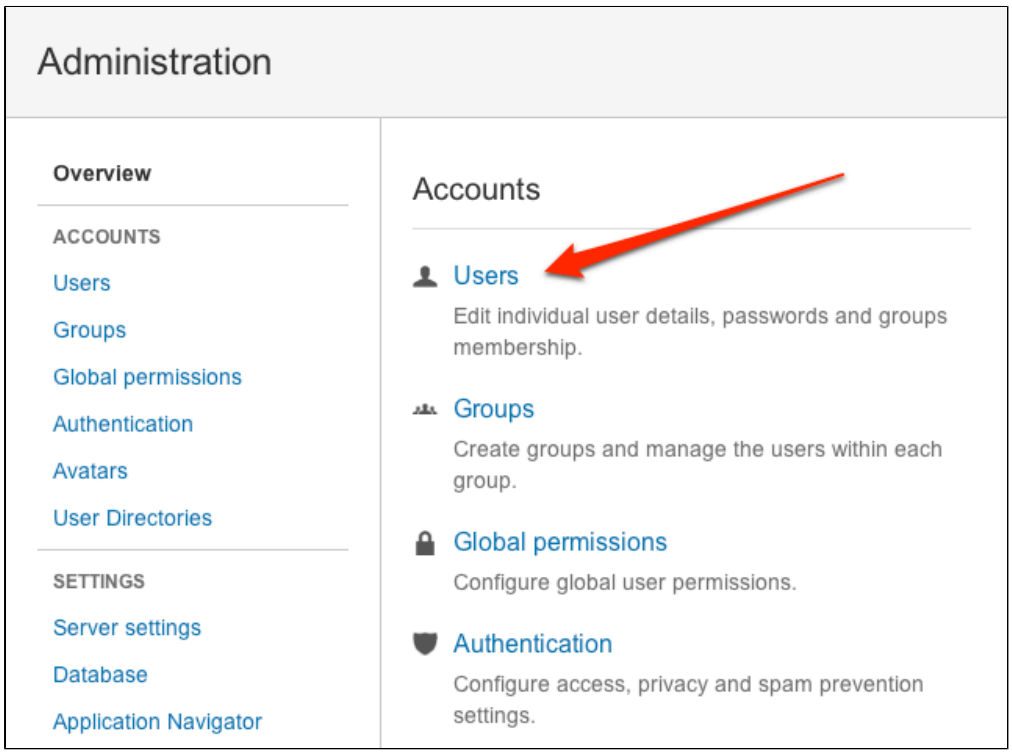
- Creating a user
- Creating a group
- Adding users to groups
  - From the user account page
  - From the group page
- Changing usernames
- Deleting users and groups

Related pages:

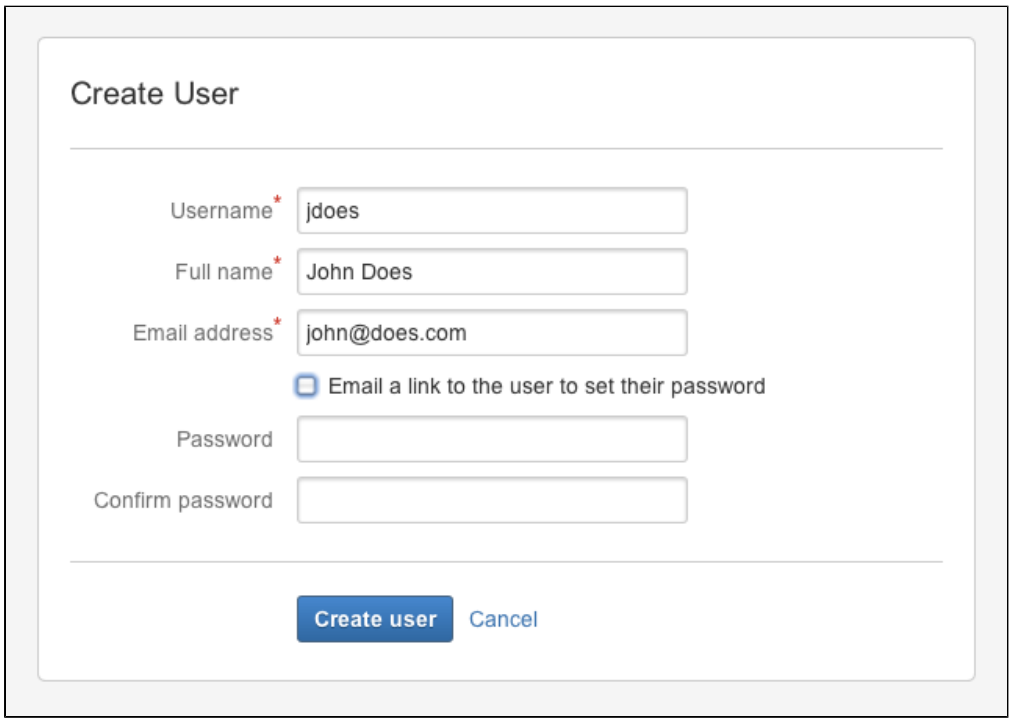
- Getting started with Stash
- External user directories

Creating a user

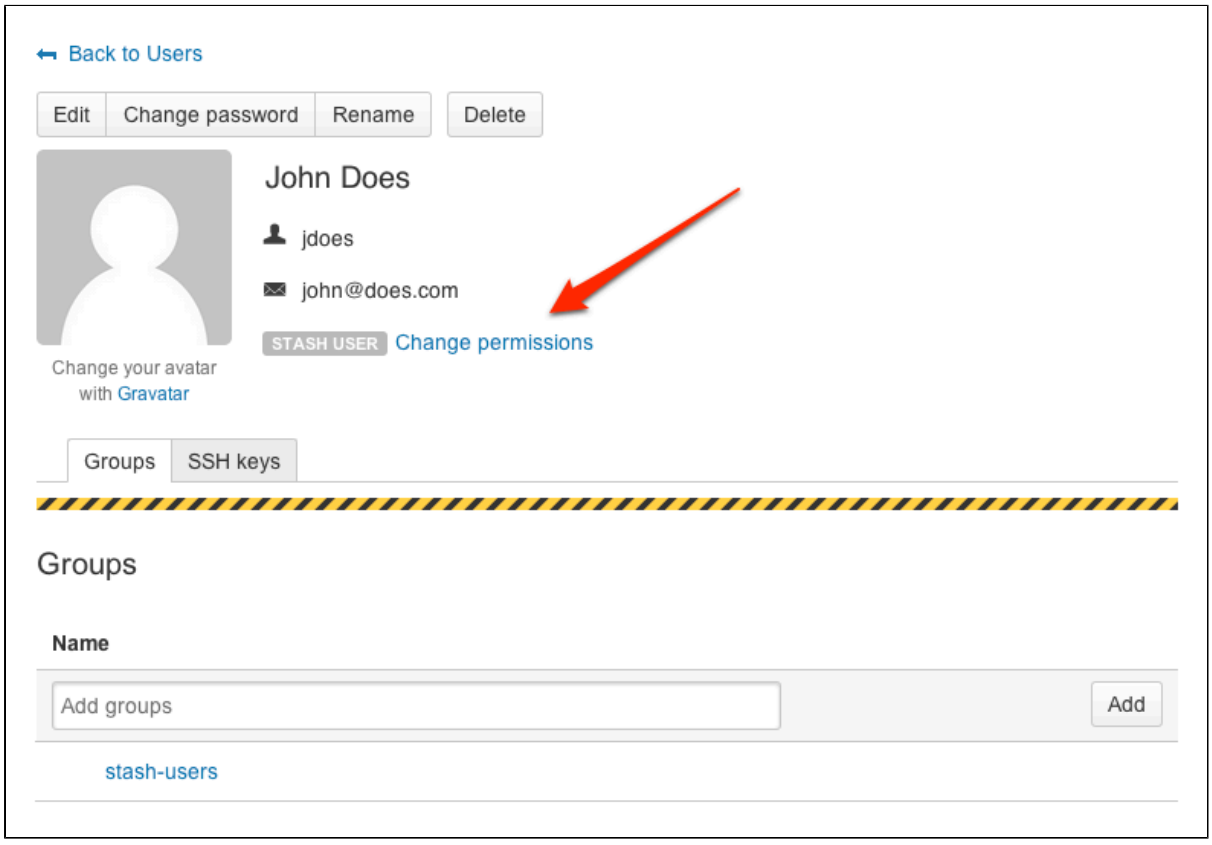
In the administration area, click **Users** (under 'Accounts') and then **Create user** (on the 'Users' screen)



Complete the form. You can either set the user's password now, or have Stash email the user with a link that they can use to set the password themselves:



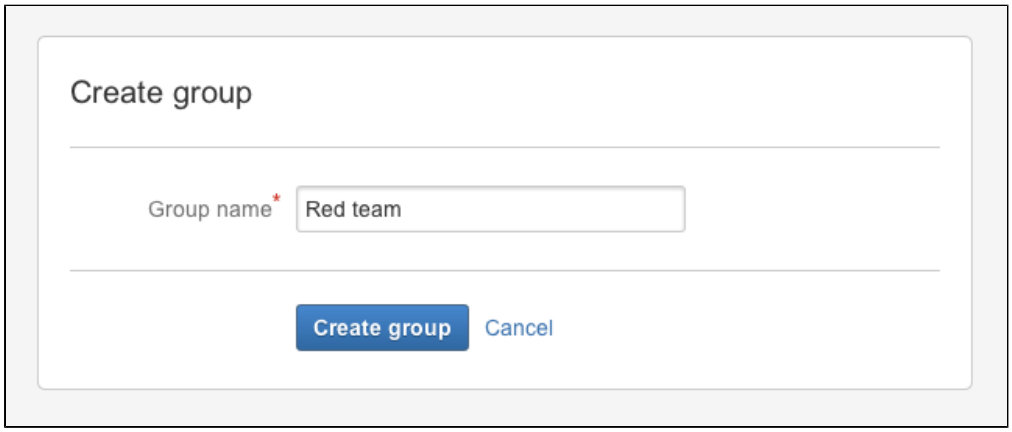
Once you've created the user, click **Change permissions** to set up their access permissions. Note that a user doesn't have access to Stash until global access permissions have been set.



See Global permissions for more information.

Creating a group

In the administration area, click **Groups** (under 'Accounts') and then **Create group.** Enter the name for the new group, and click **Create group** (again):



Now you can add users to your new group (see the next section).

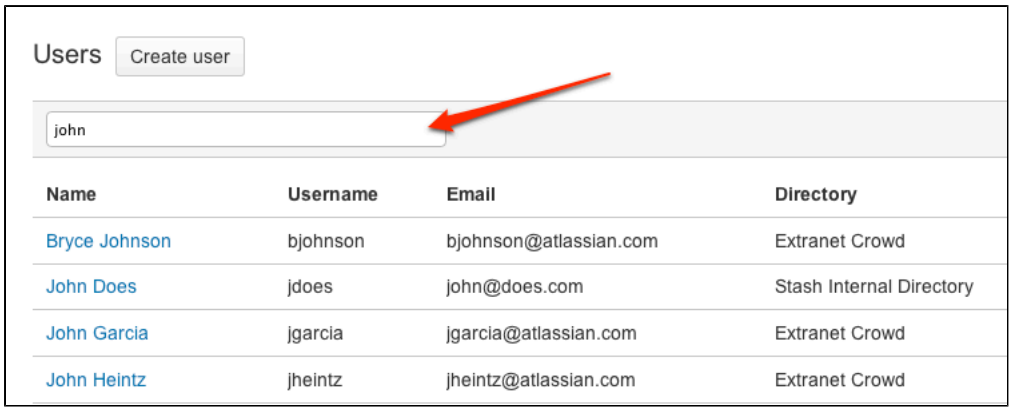
Adding users to groups

You can add users to groups in two ways:

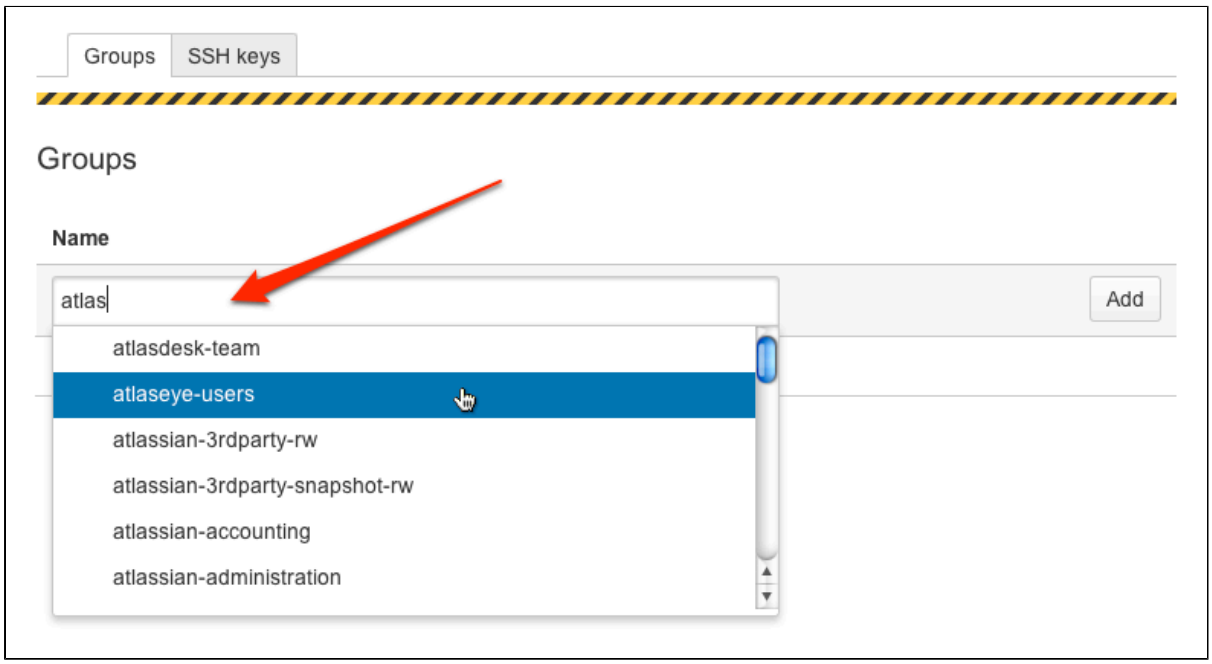
- add a particular user to multiple groups, from the user's account page in the admin area.
- add multiple users to a particular group, from the group's page.

From the user account page

To add a user to a group from the user's account page, click **Users** in the Administration section, and then use the filter to find the user:



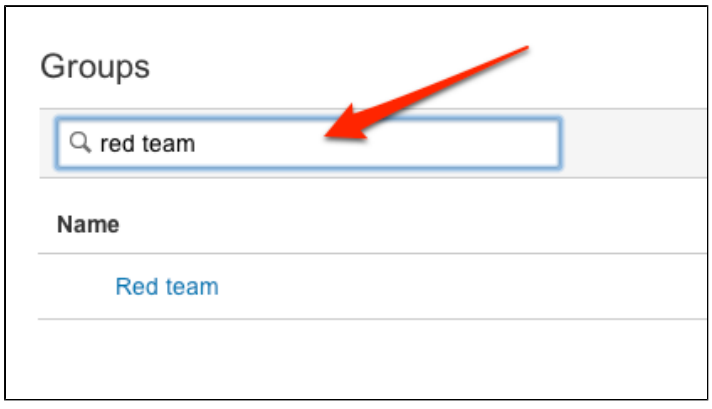
On the account page for the user, use the filter to find a group to which you want to add the user:



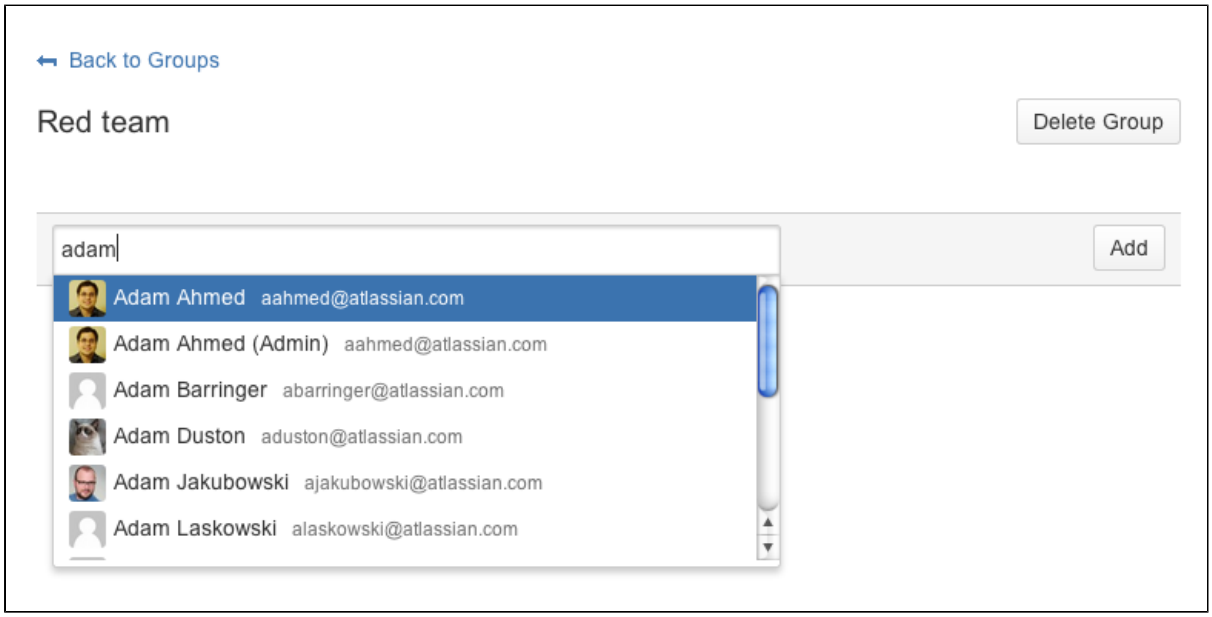
Click **Add** for each group in turn.

From the group page

To add a user to a group from the group's page, click **Groups** (under "Accounts') in the administration area, and use the filter to find the group:



On the page for the group, use the filter to find a user whom you wish to add to the group:



Click **Add** for each user you select, to make them a member of the group.

Changing usernames

You can change the username for a user account that is hosted in Stash's internal user directory. Go to **Users** in the Administration section, and use the filter to find the user. On the account page for the user, click **Rename**.

Deleting users and groups

You can delete a user or group from Stash's internal user directory, or the external directory from which Stash sources users, such as an LDAP, Crowd or JIRA server.

When a user or group is deleted from such a directory, Stash checks to see if that user still exists in another directory:

- If the user or group *does* exist in another directory, Stash assumes the administrator intended to *migrate* t he user or group between directories and we leave their data intact.
- If the user or group *does not* exist in another directory, Stash assumes the intent was to permanently delete them, and we delete the users permissions, SSH keys and 'rememberme' tokens.

Notes

- If an entire directory is deleted Stash *always* assumes it is a migration and does nothing to clean up after users and groups.
- Content which might be of historical interest (comments, pull requests, etc.) is not deleted when a user or group is. Only authentication, authorisation and data which serves no purpose to a user who can no longer log in is removed.
- In some situations, reordering the directories will change the directory that the current user comes from, if a user with the same username happens to exist in both. This behaviour can be used in some cases to create a copy of the existing configuration, move it to the top, then remove the old one. Note, however, that duplicate usernames are not a supported configuration.
- You can enable or disable a directory at any time. If you disable a directory, your configuration details will remain but Stash will not recognise the users and groups in that directory.

Limitations

- You cannot edit, disable or delete the directory that your own user account belongs to. This prevents administrators from locking themselves out of Stash, and applies to internal as well as external directories.
- You cannot remove the internal directory. This limitation aligns with the recommendation that you always keep an administrator or sysadmin account active in the Stash internal directory, so that you can troubleshoot problems with your user directories.
- You have to disable a directory before you can remove it. Removing a directory will remove the details from the database.

External user directories

You can connect Stash to external user directories. This allows you to use existing users and groups stored in an enterprise directory, and to manage those users and groups in one place.

User management functions include:

- **Authentication:** determining which user identity is sending a request to Stash.
- **Authorisation**: determining the access privileges for an authenticated user.
- **User management:** maintaining profile information in user's accounts.
- **Group membership:** storing and retrieving groups, and group membership.

It is important to understand that these are separate components of a user management system. You could use an external directory for any or all of the above tasks.

There are several approaches to consider when using external user directories wth Stash, described briefly below:

- LDAP
- JIRA
- Crowd
- Multiple directories

On this page:

- LDAP
- JIRA
- Crowd
- Multiple directories

Related pages:

- Connecting Stash to an existing LDAP directory
- Delegating Stash authentication to an LDAP directory
- Connecting Stash to Crowd
- Connecting Stash to JIRA for user management
- Users and groups
- External directory lockout recovery

- Stash provides a "read-only" connection to external directories for user management. This means that users and groups, fetched from *any external directory*, can only be modified or updated in the external directory itself, rather than in Stash.
- Connecting Atlassian Stash to your external directory is not sufficient to allow your users to log in to Stash. You must explicitly grant them access to Stash in the global permission screen.
- We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Stash license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Stash will display a warning banner. See this FAQ.
- 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.

- See also this information about deleting users and groups in Stash.

LDAP

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

There are two common ways of using an external LDAP directory with Stash:

- For full user and group management, including for user authentication — see Connecting Stash to an existing LDAP directory for instructions.
- For delegated user authentication only, while using Stash's internal directory for user and group management — see Delegating Stash authentication to an LDAP directory for instructions.

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

JIRA

You can delegate Stash user and group management, as well as user authentication, to an Atlassian JIRA instance. This is a good option if you already use JIRA in your organization. Note that Stash can only connect to a JIRA server running JIRA 4.3 or later.

You should consider using Atlassian Crowd for more complex configurations with a large number of users.

See Connecting Stash to JIRA for user management for configuration instructions.

Crowd

You can connect Stash to Atlassian Crowd for user and group management, as well as for user 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 with multiple 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 Connecting Stash to Crowd for configuration instructions.

Multiple directories

When Stash is connected directly to multiple user directories, where duplicate user names and group names are used across those directories, the effective group memberships that Stash uses for authorisation can be determined using either of these two schemes:

- 'aggregating membership'
- 'non-aggregating membership'.

See Effective memberships with multiple directories for more information about these two schemes.

Note that:

- Aggregating membership is used by default for new installations of Stash.
- Authentication, for when Stash is connected to multiple directories, only depends on the mapped groups in those directories – the aggregation scheme is not involved at all.
- For inactive users, Stash only checks if the user is active in the first (highest priority) directory in which they are found for the purpose of determining authentication. Whether a user is active or inactive does not affect how their memberships are determined.
- When a user is added to a group, they are only added to the first writeable directory available, in priority order.
- When a user is removed from a group, they are only removed from the group in the first directory the user appears in, when non-aggregating membership is used. With aggregating membership, they are removed from the group in *all* directories the user exists in.

A Stash admin can change the membership scheme used by Stash using the following commands:

- To change to *aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d
'{"membershipAggregationEnabled":true}' -u <username>:<password>
<base-url>/rest/crowd/latest/application
```

- To change to non-*aggregating membership*, substitute your own values for `<username>`, `<password>` a nd `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d
'{"membershipAggregationEnabled":false}' -u <username>:<password>
<base-url>/rest/crowd/latest/application
```

Note that these operations are different from how you make these changes in Crowd. Note also that changing the aggregation scheme can affect the authorisation permissions for your Stash users, and how directory update operations are performed.

Connecting Stash to an existing LDAP directory

You can connect Stash to an existing LDAP user directory, so that your existing users and groups in an enterprise directory can be used in Stash. The LDAP directory is used for both user authentication and account management.

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

See also this information about deleting users and groups in Stash.

On this page:

- License considerations
- Synchronisation when Stash is first connected to the LDAP directory
- Authentication when a user attempts to log in
- Connecting Stash
- Server settings
- LDAP schema
- LDAP permission
- Advanced settings
- User schema settings
- Group schema settings
- Membership schema settings

Connecting Atlassian Stash to your external directory is not sufficient to allow your users to log in to Stash. You must explicitly grant them access to Stash in the global permission screen.

We recommend that you use groups instead of invidual accounts when granting permissions.

License considerations

When connecting Stash to an external directory, be careful not to allow access to Stash by more users than your Stash license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Stash will display a warning banner. See this FAQ.

Synchronisation when Stash is first connected to the LDAP directory

When you first connect Stash to an existing LDAP directory, the Stash internal directory is synchronised with the LDAP directory. User information, including groups and group memberships, is copied across to the Stash directory.

When we performed internal testing of synchronisation with an Active Directory server on our local network with 10 000 users, 1000 groups and 200 000 memberships, we found that the initial synchronisation took about 5 minutes. Subsequent synchronisations with 100 modifications on the AD server took a couple of seconds to complete. See the option below.

Note that when Stash is connected to an LDAP directory, you cannot update user details in Stash. Updates must be done directly on the LDAP directory, perhaps using a LDAP browser tool such as Apache Directory Studio.

Option - Use LDAP filters to restrict the number of users and groups that are synchronised

You can use LDAP filters to restrict the users and groups that are synchronised with the Stash internal directory. You may wish to do this in order to limit the users or groups that can access Stash, or if you are concerned that synchronisation performance may be poor.

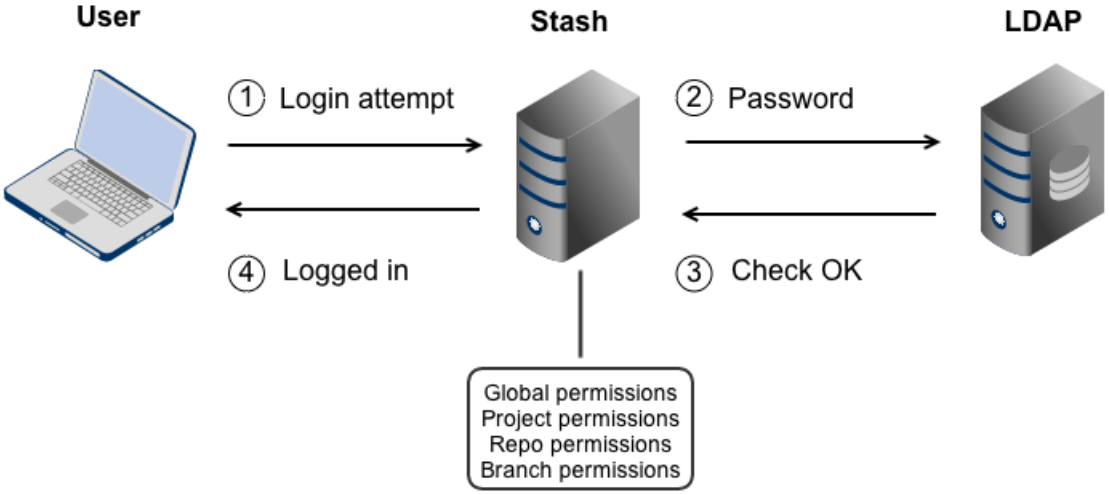
For example, to limit synchronisation to just the groups named "stash_user" or "red_team", enter the following into the **Group Object Filter** field (see Group Schema Settings below):

```
(&(objectClass=group)(|(cn=stash_user)(cn=red_team)))
```

For further discussion about filters, with examples, please see How to write LDAP search filters. Note that you need to know the names for the various containers, attributes and object classes in your particular directory tree, rather than simply copying these examples. You can discover these container names by using a tool such as Apache Directory Studio.

Authentication when a user attempts to log in

When a user attempts to log in to Stash, once synchronisation has completed, Stash confirms that the user exists in it's internal directory and then passes the user's password to the LDAP directory for confirmation. If the password matches that stored for the user, LDAP passes a confirmation back to Stash, and Stash logs in the user. During the user's session, all authorisations (i.e. access to Stash resources such as repositories, pull requests and administration screens) are handled by Stash, based on permissions maintained by Stash in its internal directory.



Connecting Stash

To connect Stash to an LDAP directory:

1. Log in as a user with 'Admin' permission.
2. In the Stash administration area, click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select either **Microsoft Active Directory** or **LDAP** as the directory type.
4. Configure the directory settings, as described in the tables below.
5. Save the directory settings.
6. Define the directory order by clicking the arrows next to each directory on the 'User Directories' screen. The directory order has the following effects:
    - The order of the directories is the order in which they will be searched for users and groups.
    - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Server settings

Setting	Description
Name	Enter a meaningful name to help you identify the LDAP directory server. Examples: • `Example Company Staff Directory` • `Example Company Corporate LDAP`
Directory Type	Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for many of the options on the rest of screen. Examples: • `Microsoft Active Directory` • `OpenDS` • And more.

Hostname	The host name of your directory server. Examples: • `ad.example.com` • `ldap.example.com` • `opends.example.com`
Port	The port on which your directory server is listening. Examples: • `389` • `10389` • `636` (for example, for SSL)
Use SSL	Check this if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.
Username	The distinguished name of the user that the application will use when connecting to the directory server. Examples: • `cn=administrator,cn=users,dc=ad,dc=example,dc=com` • `cn=user,dc=domain,dc=name` • `user@domain.name` Ensure that this is an administrator user for the LDAP engine. For example, in Active Directory the user will need to be a member of the built-in Administrators group.
Password	The password of the user specified above. **Note:** Connecting to an LDAP server requires that this application log in to the server with the username and password configured here. As a result, this password cannot be one-way hashed - it must be recoverable in the context of this application. The password is currently stored in the database in plain text without obfuscation. To guarantee its security, you need to ensure that other processes do not have OS-level read permissions for this application's database or configuration files.

LDAP schema

Setting	Description
Base DN	The root distinguished name (DN) to use when running queries against the directory server. Examples: • `o=example,c=com` • `cn=users,dc=ad,dc=example,dc=com` • For Microsoft Active Directory, specify the base DN in the following format: `dc=domain1,dc=local`. You will need to replace the `domain1` and `local` for your specific configuration. Microsoft Server provides a tool called `ldp.exe` which is useful for finding out and configuring the the LDAP structure of your server.
Additional User DN	This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example: • `ou=Users`
Additional Group DN	This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example: • `ou=Groups`

LDAP permission

Setting	Description

Read Only	LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens.
Read Only, with Local Groups	LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens. However, you can add groups to the internal directory and add LDAP users to those groups.

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called 'nested groups'. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Manage User Status Locally	If true, you can activate and deactivate users in Crowd independent of their status in the directory server.
Filter out expired users	If true, user accounts marked as expired in ActiveDirectory will be automatically removed. For cached directories, the removal of a user will occur during the first synchronisation after the account's expiration date.
Use Paged Results	Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size – that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results.
Follow Referrals	Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup `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.
Naive DN Matching	If your directory server will always return a consistent string representation of a DN, you can enable naive DN matching. Using naive DN matching will result in a significant performance improvement, so we recommend enabling it where possible. This setting determines how your application will compare DNs to determine if they are equal. • If this checkbox is selected, the application will do a direct, case-insensitive, string comparison. This is the default and recommended setting for Active Directory, because Active Directory guarantees the format of DNs. • If this checkbox is not selected, the application will parse the DN and then check the parsed version.

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 `uSNChanged` 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 the application. This setting is only available if the directory type is set to "Microsoft Active Directory".
Synchronisation Interval (minutes)	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.
Read Timeout (seconds)	The time, in seconds, to wait for a response to be received. If there is no response within the specified time period, the read attempt will be aborted. A value of 0 (zero) means there is no limit. The default value is 120 seconds.
Search Timeout (seconds)	The time, in seconds, to wait for a response from a search operation. A value of 0 (zero) means there is no limit. The default value is 60 seconds.
Connection Timeout (seconds)	This setting affects two actions. The default value is 0. • The time to wait when getting a connection from the connection pool. A value of 0 (zero) means there is no limit, so wait indefinitely. • The time, in seconds, to wait when opening new server connections. A value of 0 (zero) means that the TCP network timeout will be used, which may be several minutes.

User schema settings

Setting	Description
User Object Class	This is the name of the class used for the LDAP user object. Example: • `user`
User Object Filter	The filter to use when searching user objects. Example: • `(&(objectCategory=Person)(sAMAccountName=*))` More examples can be found here and here.
User Name Attribute	The attribute field to use when loading the username. Examples: • `cn` • `sAMAccountName` NB: In Active Directory, the 'sAMAccountName' is the 'User Logon Name (pre-Windows 2000)' field. The User Logon Name field is referenced by 'cn'.
User Name RDN Attribute	The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example: • `cn`

User First Name Attribute	The attribute field to use when loading the user's first name. Example: • `givenName`
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: • `sn`
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: • `displayName`
User Email Attribute	The attribute field to use when loading the user's email address. Example: • `mail`
User Password Attribute	The attribute field to use when loading a user's password. Example: • `unicodePwd`
User Unique ID Attribute	The attribute used as a unique immutable identifier for user objects. This is used to track username changes and is optional. If this attribute is not set (or is set to an invalid value), user renames will not be detected — they will be interpreted as a user deletion then a new user addition. This should normally point to a UUID value. Standards-compliant LDAP servers will implement this as 'entryUUID' according to RFC 4530. This setting exists because it is known under different names on some servers, e.g. 'objectGUID' in Microsoft Active Directory.

Group schema settings

Setting	Description
Group Object Class	This is the name of the class used for the LDAP group object. Examples: • `groupOfUniqueNames` • `group`
Group Object Filter	The filter to use when searching group objects. Example: • `(&(objectClass=group)(cn=*))`
Group Name Attribute	The attribute field to use when loading the group's name. Example: • `cn`
Group Description Attribute	The attribute field to use when loading the group's description. Example: • `description`

Membership schema settings

Setting	Description
Group Members Attribute	The attribute field to use when loading the group's members. Example: • `member`

User Membership Attribute	The attribute field to use when loading the user's groups. Example: • `memberOf`
Use the User Membership Attribute, when finding the user's group membership	Check this if your directory server supports the group membership attribute on the user. (By default, this is the '`memberOf`' attribute.) • If this checkbox is selected, your application will use the group membership attribute on the user when **retrieving the list of groups to which a given user belongs**. This will result in a more efficient retrieval. • If this checkbox is not selected, your application will use the members attribute on the group ('`member`' by default) for the search. • If the **Enable Nested Groups** checkbox is seleced, your application will ignore the **Use the User Membership Attribute** option and will use the members attribute on the group for the search.
Use the User Membership Attribute, when finding the members of a group	Check this if your directory server supports the user membership attribute on the group. (By default, this is the '`member`' attribute.) • If this checkbox is selected, your application will use the group membership attribute on the user when **retrieving the members of a given group**. This will result in a more efficient search. • If this checkbox is not selected, your application will use the members attribute on the group ('`member`' by default) for the search.

Connecting Stash to JIRA for user management

⚠ *This page does not apply to JIRA Cloud; you can't use JIRA Cloud to manage your Stash users.*

You can connect Stash to an existing Atlassian JIRA instance to delegate Stash user and group management, and authentication. Stash provides a "read-only" connection to JIRA for user management. This means that users and groups, fetched from JIRA, can only be modified or updated in that JIRA server, rather than in Stash.

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

Also on this page:

- Server settings
- JIRA server permissions
- Advanced settings

⚠ You need to be an administrator in JIRA and a system administrator in Stash to perform the following tasks.

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** (case-sensitive) 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 and RFC 4632.

6. Click **Save**.
7. Define the directory order, on the 'User Directories' screen, by clicking the blue up- and down-arrows next to each directory. The directory order has the following effects:
    - The order of the directories is the order in which they will be searched for users and groups.
    - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

2. Set up Stash to connect to JIRA

1. Log in to Stash as a user with 'Admin' permission.
2. In the Stash administration area click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Atlassian JIRA**.
4. Enter settings, as described below.
5. Test and save the directory settings.
6. Define the directory order, on the 'User Directories' screen, by clicking the arrows for each directory. The directory order has the following effects:
    - The order of the directories is the order in which they will be searched for users and groups.
    - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

3. Set up Stash users and groups in JIRA

In order to use Stash, users must be a member of the `Stash-users` group or have Stash global permissions. 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 those groups as members of one or both of the above Stash groups.
    - Option 2: Log in to Stash using your JIRA account and go to the administration area. Click **Global permissions** (under 'Accounts'). Assign the appropriate permissions to the relevant JIRA groups. See Global permissions.

> Connecting Atlassian Stash to JIRA for user management is not sufficient, by itself, to allow your users to log in to Stash. You must also grant them access to Stash by using one of the above 2 options.
>
> We recommend that you use groups instead of individual accounts when granting permissions. However , be careful not to add more users to those groups that your Stash license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Stash will display a warning banner. See this FAQ.
>
> See also this information about deleting users and groups in Stash.

Server settings

Setting	Description
Name	A meaningful name that will help you to identify this JIRA server amongst your list of directory servers. Examples: • `JIRA Server` • `My Company JIRA`

Server URL	The web address of your JIRA server. Examples: • `http://www.example.com:8080` • `http://jira.example.com`
Application Name	The name used by your application when accessing the JIRA server that acts as user manager. Note that you will also need to define your application to that JIRA server, via the '**Other Applications**' option in the 'Users, Groups & Roles' section of the 'Administration' menu.
Application Password	The password used by your application when accessing the JIRA server that acts as user manager.

JIRA server permissions

Setting	Description
Read Only	The users, groups and memberships in this directory are retrieved from the JIRA server that is acting as user manager. They can only be modified via that JIRA server.

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Before enabling nested groups, please check to see if nested groups are enabled on the JIRA server that is acting as user manager. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Enable Incremental Synchronisation	Enable or disable incremental synchronisation. Only changes since the last synchronisation will be retrieved when synchronising a directory..
Synchronisation Interval (minutes)	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.

Delegating Stash authentication to an LDAP directory

You can configure Stash to use an LDAP directory for delegated user authentication while still using Stash for user and group management.

You can either create new user accounts manually in the LDAP directory, or use the option to automatically crea te a user account when the user attempts to log in, as described in the Copy users on login section below.

See also this information about deleting users and groups in Stash.

To connect Stash to an LDAP directory for delegated authentication:

1. Log in to Stash as a user with 'Admin' permission.
2. Go to the Stash administration area and click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Internal with LDAP Authentication** as the directory type.
4. Configure the directory settings, as described in the tables below.
5. Save the directory settings.
6. Define the directory order by clicking the arrows for each directory on the 'User Directories' screen. The directory order has the following effects:
    - The order of the directories is the order in which they will be searched for users and groups.
    - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Connecting Atlassian Stash to your external directory is not sufficient to allow your users to log in to Stash. You must explicitly grant them access to Stash in the global permission screen.

We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Stash license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Stash will display a warning banner. See this FAQ.

On this page:

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

Server settings

Setting	Description
Name	A descriptive name that will help you to identify the directory. Examples: • `Internal directory with LDAP Authentication` • `Corporate LDAP for Authentication Only`
Directory Type	Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for some of the options on the rest of screen. Examples: • `Microsoft Active Directory` • `OpenDS` • And more.
Hostname	The host name of your directory server. Examples: • `ad.example.com` • `ldap.example.com` • `opends.example.com`
Port	The port on which your directory server is listening. Examples: • `389` • `10389` • `636` (for example, for SSL)
Use SSL	Check this box if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.
Username	The distinguished name of the user that the application will use when connecting to the directory server. Examples: • `cn=administrator,cn=users,dc=ad,dc=example,dc=com` • `cn=user,dc=domain,dc=name` • `user@domain.name`

Password	The password of the user specified above.

Manually creating users

Move the delegated authentication directory to the top of the User Directories list and create the user manually (go to **Administration** > **Users** > **Create user**). Using this manual method you must currently create a temporary password when creating users. There is an improvement request to address this:

STASH-3424 - Disable "Change password" field from admin and user page when delegated authentication is used CLOSED

> If you intend to *change* the authentication directory of your users from `Stash Internal Directory` to `Delegated LDAP Authentication` you must select the option to "Copy User on Login" since you can't create a new user that has the same username as another user in another directory.

Copying users on login

The settings described in the table below relate to when a user attempts to authenticate with Stash. This authentication attempt can occur either:

- when using the Stash login screen.
- when issuing a Git clone or push command at the command line, for a repository managed by Stash.

Setting	Description
Copy User on Login	This option affects what will happen when a user attempts to log in. If this box is checked, the user will be created automatically in the internal directory that is using LDAP for authentication when the user first logs in and their details will be synchronised on each subsequent log in. If this box is not checked, the user's login will fail if the user wasn't already manually created in the directory. If you check this box the following additional fields will appear on the screen, which are described in more detail below: • Default Group Memberships • Synchronise Group Memberships • User Schema Settings (described in a separate section below)
Default Group Memberships	This field appears if you check the **Copy User on Login** box. If you would like users to be automatically added to a group or groups, enter the group name(s) here. To specify more than one group, separate the group names with commas. Each time a user logs in, their group memberships will be checked. If the user does not belong to the specified group(s), their username will be added to the group(s). If a group does not yet exist, it will be added to the internal directory that is using LDAP for authentication. Please note that there is no validation of the group names. If you mis-type the group name, authorisation failures will result – users will not be able to access the applications or functionality based on the intended group name. Examples: • `confluence-users` • `bamboo-users,jira-users,jira-developers`

Synchronise Group Memberships	This field appears if you select the **Copy User on Login** checkbox. If this box is checked, group memberships specified on your LDAP server will be synchronised with the internal directory each time the user logs in. If you check this box the following additional fields will appear on the screen, both described in more detail below: • Group Schema Settings (described in a separate section below) • Membership Schema Settings (described in a separate section below)

LDAP schema

Setting	Description
Base DN	The root distinguished name (DN) to use when running queries against the directory server. Examples: • `o=example,c=com` • `cn=users,dc=ad,dc=example,dc=com` • For Microsoft Active Directory, specify the base DN in the following format: `dc=domain1,dc=local`. You will need to replace the `domain1` and `local` for your specific configuration. Microsoft Server provides a tool called `ldp.exe` which is useful for finding out and configuring the the LDAP structure of your server.
User Name Attribute	The attribute field to use when loading the username. Examples: • `cn` • `sAMAccountName`

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called 'nested groups'. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Use Paged Results	Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size – that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results.
Follow Referrals	Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup `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.

User schema settings

Note: this section is only visible when **Copy User on Login** is enabled.

Setting	Description
Additional User DN	This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example: • `ou=Users`

User Object Class	This is the name of the class used for the LDAP user object. Example: • `user`
User Object Filter	The filter to use when searching user objects. Example: • `(&(objectCategory=Person)(sAMAccountName=*))`
User Name RDN Attribute	The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example: • `cn`
User First Name Attribute	The attribute field to use when loading the user's first name. Example: • `givenName`
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: • `sn`
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: • `displayName`
User Email Attribute	The attribute field to use when loading the user's email address. Example: • `mail`

Group schema settings

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

Setting	Description
Additional Group DN	This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example: • `ou=Groups`
Group Object Class	This is the name of the class used for the LDAP group object. Examples: • `groupOfUniqueNames` • `group`
Group Object Filter	The filter to use when searching group objects. Example: • `(objectCategory=Group)`
Group Name Attribute	The attribute field to use when loading the group's name. Example: • `cn`
Group Description Attribute	The attribute field to use when loading the group's description. Example: • `description`

Membership schema settings

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

Setting	Description
Group Members Attribute	The attribute field to use when loading the group's members. Example: • `member`
User Membership Attribute	The attribute field to use when loading the user's groups. Example: • `memberOf`
Use the User Membership Attribute, when finding the user's group membership	Check this box if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.) • If this box is checked, your application will use the group membership attribute on the user when **retrieving the members of a given group**. This will result in a more efficient retrieval. • If this box is not checked, your application will use the members attribute on the group ('member' by default) for the search.

Connecting Stash to Crowd

You can configure Stash to use Atlassian Crowd for user and group management, and for authentication and authorisation.

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.

See also this information about deleting users and groups in Stash.

> Connecting Atlassian Stash to your external directory is not sufficient to allow your users to log in to Stash. You must explicitly grant them access to Stash in the global permission screen.
>
> We recommend that you use groups instead of individual accounts when granting permissions. However , be careful not to add more users to those groups that your Stash license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Stash will display a warning banner. See this FAQ.

On this page:

- Server settings
- Crowd permissions
- Advanced settings
- Single sign-on (SSO) with Crowd
- Using multiple directories

To connect Stash to Crowd:

1. Log in as a user with 'Admin' permission.
2. In the Stash administration area, click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Atlassian Crowd**.
4. Enter settings, as described below.
5. Test and save the directory settings.
6. Define the directory order, on the **Directories** tab, by clicking the blue up- and down-arrows next to each

directory. The directory order has the following effects:

- The order of the directories is the order in which they will be searched for users and groups.
- Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Server settings

Setting	Description
Name	A meaningful name that will help you to identify this Crowd server amongst your list of directory servers. Examples: • `Crowd Server` • `Example Company Crowd`
Server URL	The web address of your Crowd console server. Examples: • `http://www.example.com:8095/crowd/` • `http://crowd.example.com`
Application Name	The name of your application, as 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 adding an application.
Application Password	The password which the application will use when it authenticates against the Crowd framework as a client. This must be the same as the password you have registered in Crowd for this application. See the Crowd documentation on adding an application.

Crowd permissions

Stash offers **Read Only** permissions for Crowd directories. The users, groups and memberships in Crowd directories are retrieved from Crowd and can only be modified from Crowd. You cannot modify Crowd users, groups or memberships using the Stash administration screens.

For local Stash directories, **Read Only** and **Read/Write** permissions are available.

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Before enabling nested groups, please check to see if the user directory or directories in Crowd support nested groups. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Synchronisation Interval (minutes)	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.

Single sign-on (SSO) with Crowd

Once the Crowd directory has been set up, you can enable Crowd SSO integration by adding the following setting to `shared/stash-config.properties` in the Stash home directory (create this file if it doesn't exist yet):

stash-config.properties

```
# Whether SSO support should be enabled or not. Regardless of this setting SSO
authentication
# will only be activated when a Crowd directory is configured in Stash that is
configured
# for SSO.
plugin.auth-crowd.sso.enabled=true
```

Please note that you will need to correctly set up the domains of the applications involved in SSO. See Crowd SSO Domain examples.

In addition to this property, Crowd SSO integration can be tuned using the system properties described on Stash config properties.

Using multiple directories

When Stash is connected to Crowd you can map Stash to multiple user directories in Crowd.

For Crowd 2.8, and later versions, there are two different membership schemes that Crowd can use when multiple directories are mapped to an integrated application, and duplicate user names and group names are used across those directories. The schemes are called 'aggregating membership' and 'non-aggregating membership' and are used to determine the effective group memberships that Stash uses for *authorisation*. See Effective memberships with multiple directories for more information about these two schemes in Crowd.

Note that:

- *Authentication*, for when Stash is mapped to multiple directories in Crowd, only depends on the mapped groups in those directories – the aggregation scheme is not involved at all.
- For inactive users, Stash only checks if the user is active in the first (highest priority) directory in which they are found to determine *authentication*. The membership schemes described above are not used when Crowd determines if a user should have access to Stash.
- When a user is added to a group, they are only added to the first writeable directory available, in priority order.
- When a user is removed from a group, they are only removed from the group in the first directory the user appears in, when non-aggregating membership is used. With aggregating membership, they are removed from the group in *all* directories the user exists in.

An administrator can set the aggregation scheme that Stash uses when integrated with Crowd. Go to the **Direct ories** tab for the Stash instance in Crowd, and check **Aggregate group memberships across directories** to use the 'aggregating membership' scheme. When the checkbox is clear 'non-aggregating membership' is used.

Note that changing the aggregation scheme can affect the authorisation permissions for your Stash users, and how directory update operations are performed.

Global permissions

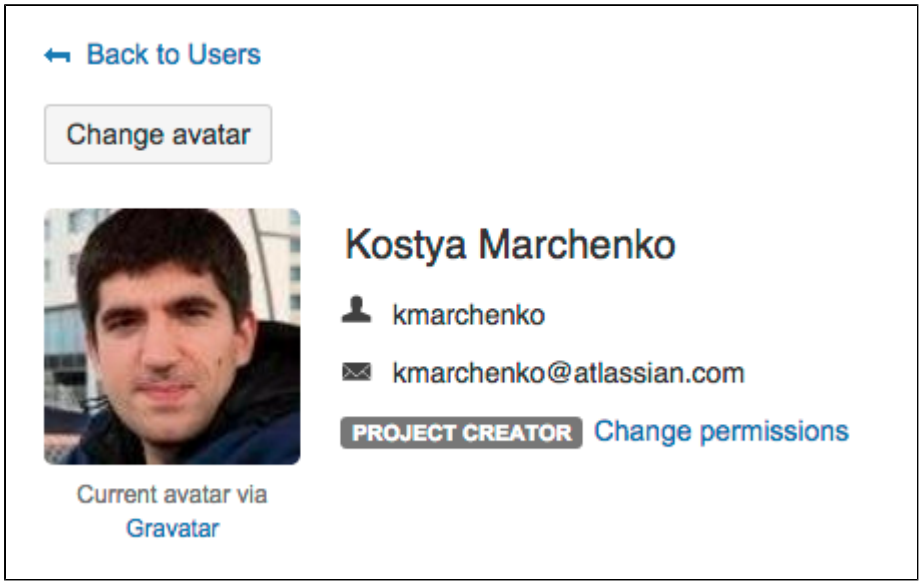
Stash uses four levels of account permissions to control user and group access to Stash projects and to the Stash server configuration.

	Login / Browse	Create projects	Manage users / groups	Manage global permissions	Edit application settings	Edit server config
Stash User	✓	✗	✗	✗	✗	✗
Project Creator	✓	✓	✗	✗	✗	✗

Administrator	✓	✓	✓	✓	✓	✗
System Administrator	✓	✓	✓	✓	✓	✓

User accounts that have not been assigned "Stash User" permission or higher, either directly or through group membership, will not be able to log in to Stash. These users are considered unlicensed and do not count towards your Stash license limit.

A user's permission level is displayed on the user's page seen from the admin area.



Note that you can also apply access permissions to projects.

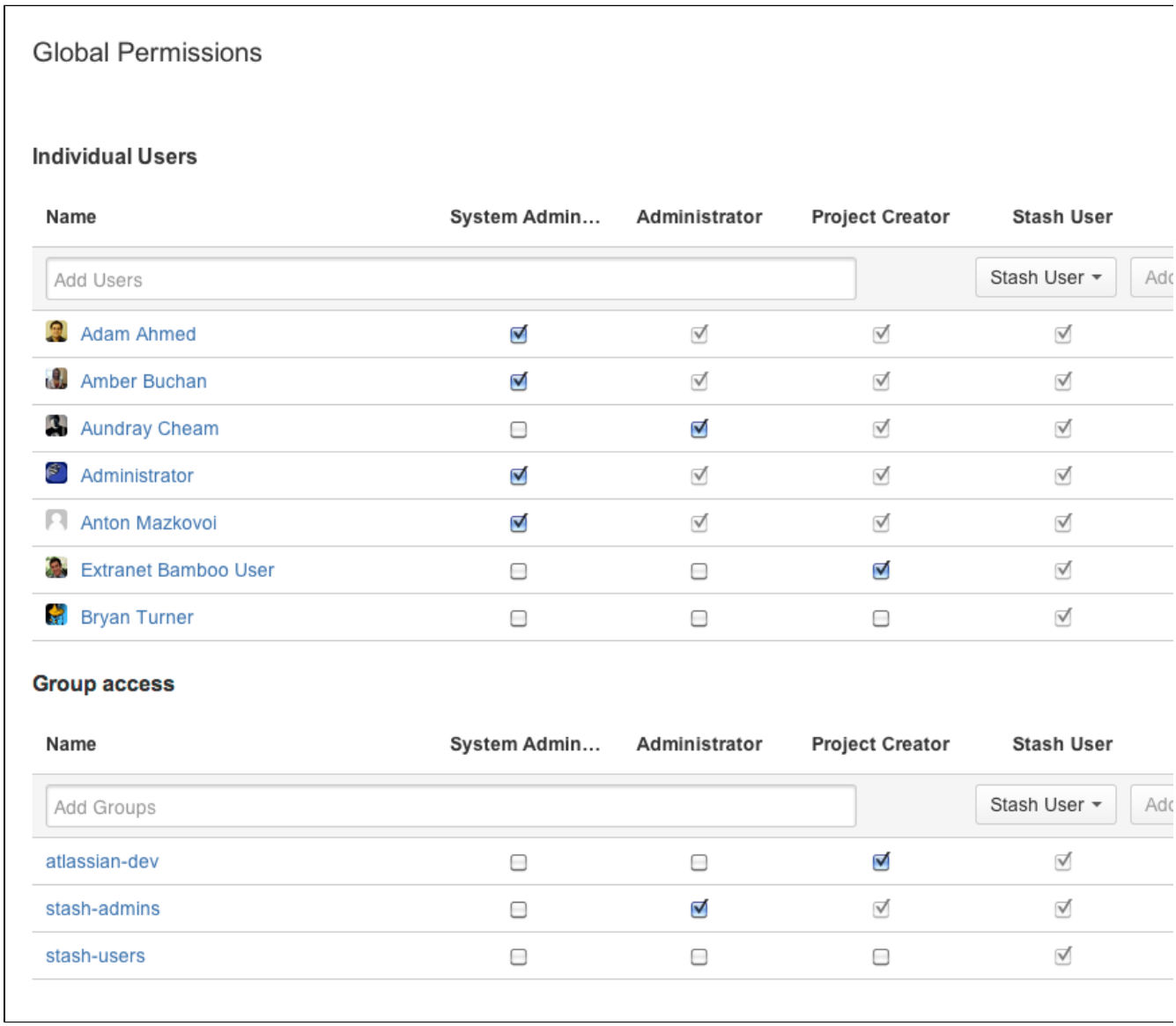
Related pages:

- Getting started with Stash
- Users and groups
- Using project permissions

To edit the account permissions for an existing Stash user or group:

1. Click the 'cog' menu in the header, to go to the admin area.
2. Click **Global permissions** (under 'Accounts').
3. Select, or clear, the permission checkboxes as required.
4. Click in the **Add Users** or **Add Groups** field to set permissions for additional users or groups.

You can remove all permissions for a user or group by clicking the X at the right-hand end of the row (when you hover there). This will remove that user or group.



Setting up your mail server

Setting up Stash to use your SMTP mail server:

- allows Stash to send notifications about events to do with pull requests. See Using pull requests in Stash.

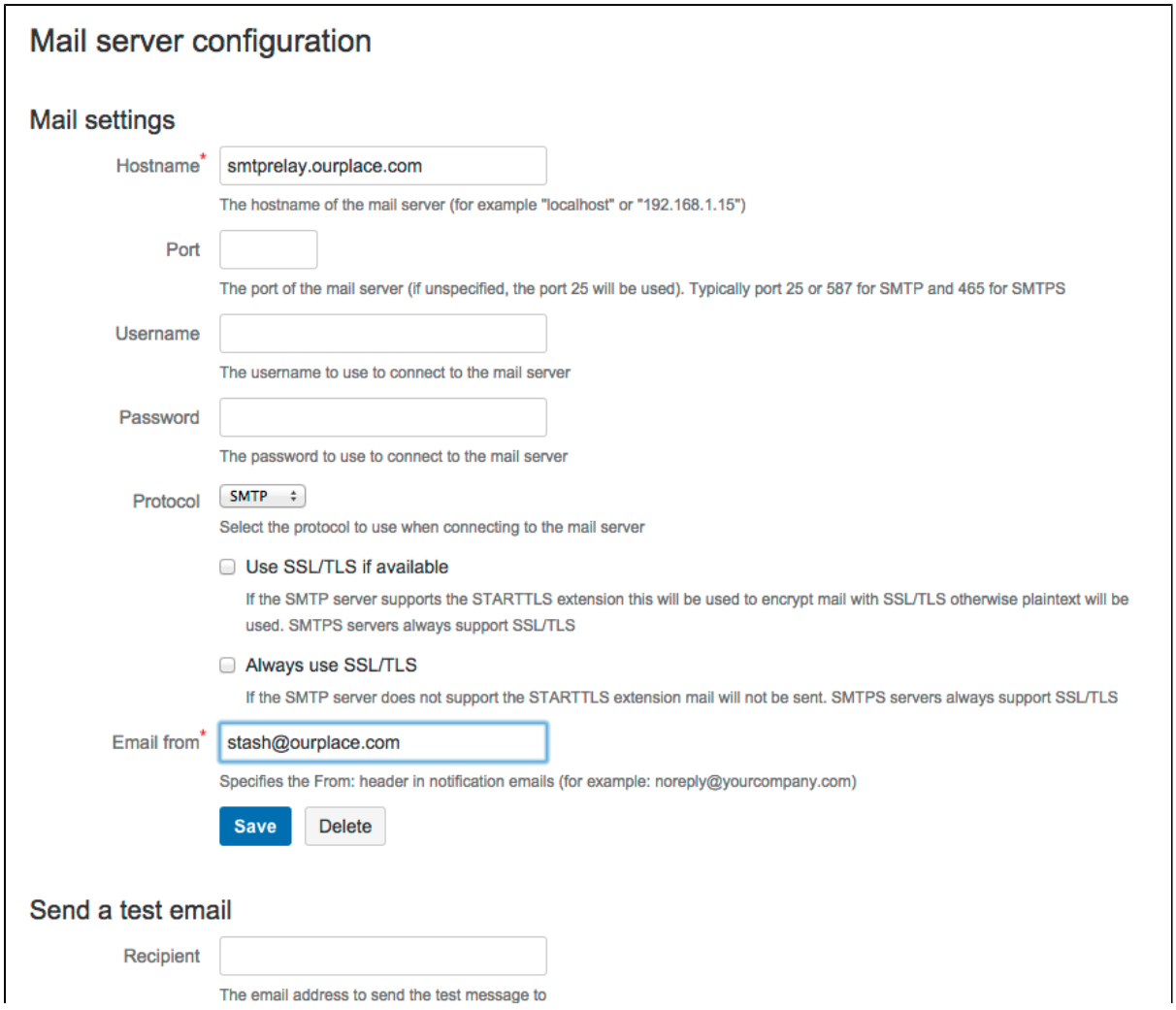
Note that if the mail server fails, notifications will be dropped.
- allows Stash to email a link to a newly created user, which the user can use to generate their own password.
- allows a user to reset his or her password if they forget it.

To configure a mail server for Stash, go to the administration area and click **Mail server** (under 'Settings'). See Supported platforms for the mail clients supported by Stash.

Complete the form and click **Save**.

Hostname	The hostname of the mail server (for example "localhost" or "192.168.1.15").
Port	The port of the mail server (if unspecified, the port 25 will be used).
Username	The username to use to connect to the mail server.
Password	The password to use to connect to the mail server.
Protocol	Use either SMTP or SMTPS when connecting to the mail server. When using SMTP, you can specify that: • SSL/TLS is used if supported by the mail server, otherwise mail is sent in plaintext. • mail should only be sent if the mail server supports SSL/TLS. See Securing email notifications below.
Email from	Specifies the 'From' header in notification emails (for example: noreply@yourcompany.com).

Anonymous user
If you wish to set up the outgoing mail server as an anonymous user, simply leave the username and password fields empty. However, in Chrome, these fields may be auto-populated, leading to an error – as a workaround, try using a different browser.



Securing email notifications

Stash 3.6 and later versions support the following protocols:

- SMTP, where mail is not encrypted.
- SMTP encrypted by SSL/TLS using the STARTTLS extension, where the protocol conversation is upgraded only if SSL/TLS is supported by the mail server, but otherwise remains as plaintext.
- SMTP, where STARTTLS support is required on the mail server, otherwise mail is not sent.
- SMTPS (where the whole protocol conversation uses SSL/TLS).

Note that if you use either SMTP with STARTTLS, or SMTPS, and connect to a self-signed mail server, you may need to import the server's cert ificate and set up a custom cacerts file for Stash (just as you do for any outbound SSL/TLS connection to a self-signed server). See this Stash knowledge base article for information about how to do that.

Configuring the mail server to use Gmail

If you wish to connect to a Gmail account for email notifications in Stash, refer to the Configuring the Mail Server to Use Gmail guide.

In particular, note that Gmail won't show images in the email because of the way that Google loads images on their servers. For Google Apps, a Stash administrator can solve the problem by adding the Stash domain name to a whitelist – see https://support.google.com/a/answer/3299041?hl=en for more information.

Linking Stash with JIRA

See JIRA integration for a description of all the integrations you get when Stash is linked with JIRA.

You can also use JIRA for delegated user management. See External user directories.

This page describes how to link Stash to JIRA.

On this page:

- Linking Stash with JIRA
- Restrictions for JIRA integration
- Restrictions when linking Stash with JIRA Cloud
- Known issues with JIRA integration
- Troubleshooting integration with JIRA

Linking Stash with JIRA

You can integrate Stash with one or more instances of JIRA by means of 'application links'. You set up application links either:

- during the Stash install process, using the Setup Wizard, or
- at any time after installation, as described below.

To link Stash to a JIRA server:

1. Click **Application Links** (under 'Settings') in the Stash admin area.
2. Enter the URL for the JIRA instance you want to link to and click **Create new link**.
3. Complete the application link wizard to connect Stash to your JIRA server. You *must* make use of the automatic link-back from JIRA to Stash to get full integration (you'll need JIRA system administrator global permission for that).

When you create a new application link between JIRA and an instance of Stash, 2-legged (2LO) and 3-legged OAuth (3LO) are enabled by default. 2LO is required for information from Stash to be included in the summaries in the Development panel of the JIRA issue; 3LO checks that a JIRA user has authenticated with Stash before they get to see the information in any of the details dialogs.

An older application link between JIRA and Stash will need to have 2LO authentication explicitly enabled.

Click here to see how to enable 2-legged OAuth...

An existing application link between JIRA and Stash (that perhaps used Trusted Apps authentication) needs to have 2-legged authentication (2LO) enabled for both outgoing and incoming authentication, so that information from Stash can be included in the JIRA Development panel summaries.

When updating an older application link to use OAuth, 3-legged authentication is applied by default, but you need to explicitly enable 2LO. Enable 2-legged authentication for the application link from within JIRA as follows:

1. Go to the JIRA admin area and click **Add-ons** > **Application Links**.
2. Click **Edit** for the app link with the Stash instance.
3. For both **Outgoing Authentication** and **Incoming Authentication**:
    a. Click **OAuth**
    b. Check **Allow 2-legged OAuth**.
    c. Click **Update**.

The application link update process will involve logging you into Stash for a short time to configure that end of the link, before returning you to JIRA.

Note that:

- The following system plugins must be enabled in Stash. These are bundled and enabled by default in Stash 2.10 (and later):
    - Atlassian Navigation Links Plugin (com.atlassian.plugins.atlassian-nav-links-plugin)
    - Stash Dev Summary Plugin (com.atlassian.stash.stash-dev-summary-plugin).
- Users who can see summarized data in the Development panel may not have permission to see all the information that contributed to those summaries and which is visible in the details dialogs (for example, for branches, commits and pull requests). That is, the details dialogs respect the access permissions that users have in the connected applications.
- Users must have the 'View Development Tools' permission within JIRA to see the Development panel within an issue.
- When Stash 2.10 or later is linked with JIRA 6.2 or later, you won't see the Source tab at the bottom of the JIRA View Issue screen any more.
- Application links must have Trusted Applications and Basic Access authentication disabled. The Development panel in the JIRA issue view only supports OAuth authentication.
- Stash only begins scanning commit messages for JIRA issue keys on the first push after you created the application link to JIRA – the scan may take a short time.

More detailed information about application links can be found on Configuring Application Links.

Restrictions for JIRA integration

- The display of details for JIRA issues, for example when viewing a pull request, relies on the JIRA 5.0 REST API. Issue details are not displayed when Stash in integrated with JIRA versions earlier than 5.0.
- Transitioning JIRA issues requires OAuth AppLinks. If only a Basic Auth applink is set up, users will be able to view issue details, but will not be able to transition issues.
- JIRA permissions are respected, so a user who is not permitted to transition an issue in JIRA will not see the transition buttons in Stash.
- If Stash is linked with multiple JIRA instances and the JIRA projects happen to have the same key, only the issue from the instance marked as Primary will be displayed. See Making a Primary Link for Links to the Same Application Type.

Restrictions when linking Stash with JIRA Cloud

In addition, there are port restrictions, and other limitations, when linking Stash with JIRA Cloud. Pleases see Integrating JIRA Cloud with Stash.

Known issues with JIRA integration

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

- There is no checking for project or issue-key validity; Stash may link to issues that do not actually exist:
  STASH-2470 - JIRA Integration: Check for issue validity before linking issues OPEN

We apologise for the inconvenience. Please watch the issue to keep track of our progress.

Troubleshooting integration with JIRA

> **Having trouble integrating your Atlassian products with Application Links?**
> We've developed a guide to troubleshooting Application Links, to help you out. Take a look at it if you need help getting around any errors or roadblocks.

There are a few scenarios where the integration of Stash with JIRA can produce an error or may not function as intended:

Unable to see the Development panel within a JIRA issue

You must have the 'View Development Tools' permission in order to see the Development panel. See the section (JIRA) Managing Global Permissions > Granting global permissionsfor details.

The application link is misconfigured

This can result if authentication for the application link has not been set up. See Troubleshooting JIRA Integration.

You don't have permission to access the JIRA project

If you don't have permission to access the JIRA project then Stash is unable to display issues.

The JIRA server is of an unsupported version

Stash can integrate with JIRA 4.3.x, or later. Some features require higher versions of JIRA to function properly. See Integrating Stash with Atlassian applications for details.

The JIRA issue key is invalid

Stash doesn't check for invalid issue keys, such as UTF-8. An error will result if Stash tries to connect to an issue that doesn't exist.

The JIRA issue keys are of a custom format

Stash assumes that JIRA issue keys are of the default format (that is, two or more uppercase letters (`[A-Z][A-Z]+`), followed by a hyphen and the issue number, for example STASH-123). By default, Stash will not recognise custom JIRA issue key formats. See Using custom JIRA issue keys with Stash for details.

The Application Link is created with OAuth only without the option to create a link using Trusted Applications

Stash allows a user with global permissions of "Administrator" to create an OAuth only application link. You need to log in with a user having "System Administrator" privileges to create an application link using Trusted Applications.

JIRA FishEye-Stash Plugin compatibility

Some aspects of Atlassian JIRA's support for Stash is built into the FishEye/Stash Plugin that is bundled with

JIRA. The plugin allows you to see all of your code changes in one place, even if you're running multiple Atlassian FishEye and Stash servers.

The information on this page applies to Stash vesions 2.1 and later.

On this page:

- Supported JIRA versions
- Plugin upgrade guide

Related pages:

- JIRA integration
- Linking Stash with JIRA
- Stash releases

Supported JIRA versions

If you're using a version of JIRA earlier than **5.0.4** you may need to upgrade the FishEye/Stash plugin in JIRA to get support for Atlassian Stash.

JIRA Version	Compatibility	FishEye/Stash Plugin URL
5.0.4+	Works straight out of the box!	NA
5.0-5.0.3	Requires JIRA FishEye/Stash Plugin 5.0.4.1	https://maven.atlassian.com/content/repositories/atlassian-contrib/com/atlassian/jira
Limited support for JIRA 4.4.x and earlier		
JIRA 4.3+ allows for showing commits associated with issues in JIRA. However, viewing issues within Stash is r		
4.4.x	Requires JIRA FishEye/Stash Plugin 3.4.12	https://maven.atlassian.com/content/repositories/atlassian-contrib/com/atlassian/jira
4.3.x	Requires JIRA FishEye/Stash Plugin 3.1.8	https://maven.atlassian.com/contrib/com/atlassian/jira/plugins/jira-fisheye-plugin/3.1
Earlier versions	JIRA-to-Stash integration is unsupported	NA

Plugin upgrade guide

To upgrade the FishEye/Stash 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.

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/Stash Plugin

in JIRA 4.3 for more details.

Using custom JIRA issue keys with Stash

Stash assumes that JIRA issue keys are of the default format (that is, two or more uppercase letters (`[A-Z][A-Z]+`), followed by a hyphen and the issue number, for example STASH-123). By default, Stash will not recognise custom JIRA issue key formats.

You can use custom JIRA issue key formats with Stash, however note that integrations with JIRA can depend on using the default issue key format in both JIRA and Stash. See Integrating using custom JIRA issue keys for more details.

Configure Stash to recognize custom issue key formats by editing `<Stash installation directory>/bin/setenv.sh` (on Windows, edit `<Stash installation directory>/bin/setenv.bat` instead).

To override the default issue key format, use the JVM_SUPPORT_RECOMMENDED_ARGS property, like this:

Stash 2.8, and later:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-Dintegration.jira.key.pattern=\"(<Some different regex>)\""
```

Stash versions up to 2.7.x:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-Dstash.jira.key.pattern=\"(<Some different regex>)\""
```

Notice that the parameter name was changed for Stash 2.8.

You will need to restart Stash.

For example, to use lowercase letters in issue keys, use a regex with the parameter like this:

Stash 2.8, and later:

```
"-Dintegration.jira.key.pattern=\"((?<!([a-z]{1,10})-?)[a-z]+-\d+)\""
```

Stash versions up to version 2.7.x:

```
"-Dstash.jira.key.pattern=\"((?<!([a-z]{1,10})-?)[a-z]+-\d+)\""
```

See also Reindex JIRA issue keys.

As always, please back up your home directory (and perhaps the database) before performing any manual operation on Stash. Consider testing this change on another copy of Stash before using it in production.

Connecting Stash to an external database

This page provides information about using Stash with an external database.

Stash ships with an embedded database that it uses straight out-of-the-box, with no configuration required. This is great for evaluation purposes, but for production installations we recommend that you use one of the supporte d external databases.

Please refer to Supported platforms for the versions of external databases supported by Stash.

If you just want to change the password for the external database, see How do I change the external database password.

Instructions for connecting Stash to the supported external databases:

- Connecting Stash to PostgreSQL
- Connecting Stash to Oracle
- Connecting Stash to SQL Server
- Connecting Stash to MySQL

> MySQL is **not** supported for Stash Data Center instances. MySQL is supported for Stash Server (standalone) instances, but not recommended. See Connecting Stash to MySQL for more information.

Why would I want to use an external database?

Stash ships with an embedded database that is great for evaluation purposes, but for production installations we recommend that you make use of one of the supported external databases, for the following reasons:

- **Improved protection against data loss**: The Stash built-in database, which runs HSQLDB, is susceptible to data loss during system crashes. External databases are generally more resistant to data loss during a system crash. HSQLDB is not supported in production environments and should only be used for evaluation purposes.
- **Performance and scalability**: If you have a large number of users on your Stash instance, running the database on the same server as Stash may slow it down. We recommend that for large installations, Stash and the DBMS are run on separate machines. When using the embedded database, the database will always be hosted and run on the same server as Stash, which will limit performance.
- **Unified back-up**: Use your existing DBMS tools to back up your Stash database alongside your organisation's other databases.
- **Stash Data Center support**: If you want to upgrade your instance to Stash Data Center, either now or in the future, to take advantage of the performance-at-scale and high availability benefits of running Stash in clustered mode, then you **must** use an external database. HSQLDB is not supported in Stash Data Center.

Using the Database Migration Wizard

> The Database Migration Wizard is not supported in Stash Data Center instances while more than one cluster node is running. To migrate databases for a Stash Data Center instance, you should perform the migration before starting multiple cluster nodes.

You can use the Database Migration Wizard to migrate the Stash data:

- from the embedded database to a supported external DBMS.
- to another instance of the same DBMS.
- from one DBMS to another supported DBMS (for example, from MySQL to PostgreSQL).

You need to have created the DBMS (such as PostgreSQL) that you wish to migrate the Stash data to before running the Migration Wizard.

To run the Database Migration Wizard:

1. Log in to Stash.
2. In the administration area, click **Database** (under 'Settings').
3. Click **Migrate database** and follow the instructions for running the migration.

Notes about database migration

- **Back up the database and Stash home directory:**
  Before starting the database migration process you should back up your Stash home directory. If you

intend to migrate from one external database to another, you should also backup the existing database before proceeding. See Data recovery and backups for more information.

- **Stash will be unavailable during the migration:**
  Stash will not be available to users during the database migration operation. In addition, running the migration when people are using Stash can sometimes cause the migration to time out waiting for all activity in Stash that uses the database to complete. For these reasons we recommend that you run the database migration outside of normal usage periods.

- **Migration will usually take less that 30 minutes:**
  The duration of the migration process depends on the amount of data in the Stash database being migrated. For new installations of Stash, containing very little data, the migration process typically takes just a few seconds. If you have been using Stash for some time, its database will contain more data, and the migration process will therefore take longer. If Stash has been linked to a JIRA instance, and there are hundreds of thousands of commits in Stash with JIRA keys in the commit messages, the migration may take tens of minutes.

- **We strongly recommend using a new clean database for the new Stash database:**
  In case of a migration failure, Stash may have partially populated the target database. If the target database is new (therefore empty) and set aside for Stash's exclusive use, it's very easy to clean up after a failed migration; just drop the target database and use a clean target database instance for the next attempt.

- **Ensure your** Stash home directory **is secured against unauthorised access**:
  - After the migration, the connection details (including the username and password) for the database are stored in the `stash-config.properties` file.
  - Migration will create a dump file of the contents of your database in the Stash home `export` directory. This is used during the migration and is kept for diagnostic purposes in the case of an error. You may remove this after migration but it may reduce Atlassian Support's ability to help you in the case of migration issues.
  - You can edit the database password if needed after migration.

Connecting Stash to MySQL

This page describes how to connect Stash to a MySQL or MariaDB database. The procedure for MySQL and MariaDB is the same, except where noted below. See Connecting Stash to an external database for general information.

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

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

Stash Server does its best to work around the MySQL / MariaDB behavior - see issues STASH-4517, STASH-4701 and others for example. However, under very heavy load you will generally get better performance with any of the other database engines supported by Stash (such as PostgreSQL which is also freely available) than you will with MySQL or MariaDB. Please see Connecting Stash to an external database for instructions on migrating your data to one of these other engines.

MySQL and MariaDB are not supported in Stash Data Center
Stash Data Center does not support any version of MySQL or MariaDB. With Stash Data Center you must use one of the other database engines supported by Stash (such as PostgreSQL which is also freely available). Please see Connecting Stash to an external database for instructions on migrating your data to one of these other engines.

MySQL 5.6.x compatibility
Note that Stash Server is not compatible at all with versions of MySQL 5.6 earlier than 5.6.16 because of bugs in its query optimizer (#68424, #69005). Please watch STASH-3164 for further updates on this. Stash Server does support versions of MySQL 5.6 from 5.6.16 on.

See Supported platforms for the versions of MySQL and MariaDB supported by Stash Server.

The overall process for using a MySQL or MariaDB database with Stash is:

1. Install MySQL or MariaDB where it is accessible to Stash. It is assumed here that you already have MySQL or MariaDB installed and running. See the MySQL documentation at http://dev.mysql.com/doc/.
2. Create the database and user on the MySQL / MariaDB server for Stash to use.
3. Download and install the JDBC driver.
4. Migrate Stash to the MySQL / MariaDB database.

Create the Stash database

Before you can use Stash with MySQL or MariaDB, you must set up the MySQL or MariaDB server as follows:

Step	Notes
Create database	Create a database on MySQL or MariaDB for Stash to use.
Create database user	Create a Stash user on the database.
Character encoding	Configure the database to use `utf8` character set encoding. Note that Stash on MySQL and MariaDB does not support 4 byte UTF-8 characters.
Collation	Configure the database to use `utf8_bin` collation (to ensure case sensitivity).
Logging format	If MySQL or MariaDB is using binary logging, configure the database to use a binary logging format of either `MIXED` or `ROW`. Refer to the MySQL documentation. Note that Stash sets the MySQL / MariaDB transaction isolation level to `READ-COMMITTED` when it connects to the database. Packages of MySQL or MariaDB in some Linux distributions may be configured with `binlog_fmt=statement` by default. Before using such packages with Stash you must change this to either `mixed` or `row`. See this KB article for more information.
Connection timeout	Stash requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be see mingly random connection errors.

Here is an example of how to do that. When Stash and MySQL / MariaDB run on the same physical computer (accessible through `localhost`), run the following commands (replacing `stashuser` and `password` with your own values):

```
mysql> CREATE DATABASE stash CHARACTER SET utf8 COLLATE utf8_bin;
mysql> GRANT ALL PRIVILEGES ON stash.* TO 'stashuser'@'localhost' IDENTIFIED BY
'password';
mysql> FLUSH PRIVILEGES;
mysql> QUIT
```

This creates an empty MySQL / MariaDB database with the name `stash`, and a user that can log in from the host that Stash is running on who has full access to the newly created database. In particular, the user should be allowed to create and drop tables, indexes and other constraints.

If the MySQL / MariaDB database and Stash servers are on the same physical computer, you can use `localhost` and *not set a password* by omitting `IDENTIFIED BY 'password'` from the 2nd MySQL statement above (if you trust the security *within* this computer).

If the MySQL / MariaDB database and Stash servers are on different computers, just replace the `localhost` part of the GRANT ALL statement above with the hostname of the machine that Stash is running on. See the documentation at http://dev.mysql.com/doc/refman/5.1/en/account-names.html.

Note that Stash will generally require about 25–30 connections to the database. The maximum number of connections is a configurable system property – see Database pool.

Download and install the JDBC driver

The JDBC drivers for MySQL / MariaDB are *not* bundled with Stash (due to licensing restrictions). You need to download and install the driver yourself, after you have installed Stash.

1. Download the MySQL Connector/J JDBC driver from the download site.

   > **The MariaDB Java Client is not compatible with Stash**
   > The MySQL Connector/J must be used for both MySQL and MariaDB. The MariaDB Java Client is ***not*** compatible with Stash and is not supported.

2. Expand the downloaded zip/tar.gz file.
3. Copy the mysql-connector-java-5.1.XX-bin.jar file from the extracted directory to your `<Stash home directory>/lib` directory (for Stash 2.1 or later).
4. Stop, and then restart Stash. See Starting and stopping Stash.

Migrate Stash to the MySQL / MariaDB database

You can migrate Stash to the MySQL or MariaDB database created above, either from the embedded database or from another external database.

The migration process makes a backup of your existing Stash database in `exports` under the Stash home directory. See Data recovery and backups for further information about backing up Stash.

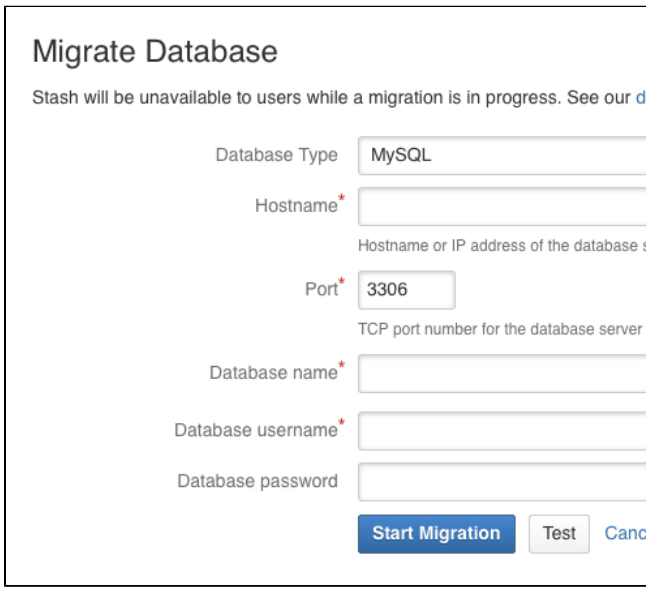
Run the migration as follows:

1. In the administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **MySQL** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

See these notes about database migration.

Hostname	The host name or IP address of the computer running the database server.

Port	The TCP port with which Stash can connect to the database server. The default value is the default port that MySQL or MariaDB runs against. You can change that if you know the port that your MySQL or MariaDB instance is using.
Database name	The name of the database that Stash should connect to.
Database username	The username that Stash should use to access the database.
Database password	The password that Stash should use to access the database.



Connecting Stash to Oracle

This page describes how to connect Stash to a Oracle database.

The overall process for using a Oracle database with Stash is:

- Install Oracle where it is accessible to Stash.
- Create a database and user on the Oracle server for Stash to use.
- Install Stash on Windows, or on Linux or Mac. See Getting started.
- Either:
  - at Stash install time, run the Setup Wizard to connect Stash to the Oracle database, or
  - at a later time, migrate Stash to the Oracle database. See Using the Database Migration Wizard .

It is assumed here that you already have Oracle installed and running. For information about installing Oracle and creating Oracle databases, see the Oracle documentation pages. For the versions of Oracle supported by Stash see Supported platforms.

On this page:

Prerequisites
Connect Stash to the Oracle database
Install the JDBC driver

Related pages:

- Connecting Stash to an external database
- Connecting Stash to MySQL
- Connecting Stash to PostgreSQL
- Connecting Stash to SQL Server

Prerequisites

Backup

If you are migrating your data from the internal Stash database, back up the Stash home directory.

If you are migrating your Stash data from a different external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See Data recovery and backups.

Create the Stash database

Before you can use Stash with Oracle, you must set up Oracle as follows:

- Ensure that you have a database instance available for Stash (either create a new one or use an existing one)
  The character set of the database must be set to either `AL32UTF8` or `UTF8`, to support storage of Unicode data as per the Oracle documentation.
  Note that it is important to the proper operation of Stash that the database store its data in a case-sensitive manner. By changing the values of the NLS_COMP and/or NLS_SORT variables, it is possible to cause Oracle to perform its searches in a case-insensitive manner. We therefore strongly recommend that those variables be left at their default values.
- Create a user that Stash will connect as (e.g. **`stash`**).
  Remember the database user name; it will be used to configure Stash's connection to the database in subsequent steps.
  When you create a user in Oracle, a schema is automatically created.
  It is strongly recommended that you create a new database user for use by Stash rather than sharing one that is used by other applications or people.
- Grant the Stash user `connect` and `resource` roles only. The `connect` role is required to set up a connection, while `resource` role is required to allow the user to create objects in its own schema.
- Create a local `all_objects` view to the user's schema, so that there is no possibility that a table with the same name as one of the Stash tables in another schema will cause any conflicts.
- Note that Stash requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be seemingly random connection errors.

The format of the command to create a user in Oracle is:

```
CREATE USER <user>
  IDENTIFIED BY <password>
  DEFAULT TABLESPACE USERS
  QUOTA UNLIMITED ON USERS;
GRANT CONNECT, RESOURCE to <user>;
CREATE VIEW <user>.all_objects AS
  SELECT *
  FROM sys.all_objects
  WHERE owner = upper('<user>');
```

Here is a simple example, using SQL*Plus, of how one might create a user called **`stash`** with password **`jdHyd6Sn21`** in tablespace **`users`**, and grant the user a minimal set of privileges. When you run the command on your machine, remember to replace the user name, password and tablespace names with your own values.

```
CREATE USER stash
  IDENTIFIED BY jdHyd6Sn21
  DEFAULT TABLESPACE USERS
  QUOTA UNLIMITED ON USERS;
GRANT CONNECT, RESOURCE to stash;
CREATE VIEW stash.all_objects AS
  SELECT *
  FROM sys.all_objects
  WHERE owner = upper('stash');
```

This creates an empty Oracle schema with the name `stash`, and a user that can log in from the host that Stash is running on and who has full access to the newly created schema. In particular, the user is allowed to create sessions and tables.

Stash will generally require about 25–30 connections to the database. The maximum number of connections is a configurable system property – see Database pool.

Connect Stash to the Oracle database

You can now connect Stash to the Oracle database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate to Oracle, either from the embedded Stash database or from another external database.

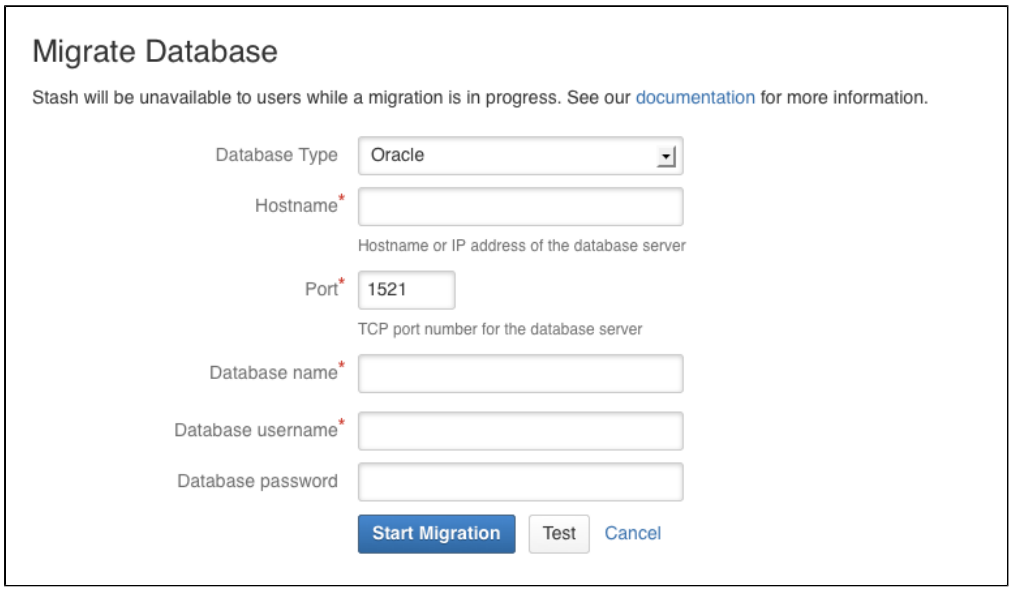
When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **Oracle** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next,** and follow the instructions in the Stash Setup Wizard.

When migrating to Oracle

1. In the Stash administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **Oracle** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Hostname	The host name or IP address of the computer running the Oracle server.
Port	The TCP port with which Stash can connect to the database server. The default value is the default port that Oracle runs against. You can change that if you know the port that your Oracle instance is using.
Database name	The system identifier of the Oracle instance that Stash should connect to. Stash does not support connecting to `Oracle servers` which are using `SIDs or TNS Alias` to identify themselves; it requires `the fully qualified Service Name` instead.
Database username	The username that Stash should use to access the database.
Database password	The password that Stash should use to access the database.



Install the JDBC driver

This section is only relevant to some distributions of Stash, for example if you are running Stash via the Atlassian Plugin SDK, or have built Stash from source.

If the Oracle JDBC driver is *not* bundled with Stash, you will need to download and install the driver yourself.

1. Download the appropriate JDBC driver from the Oracle download site.
2. Copy the downloaded jar file to your `<Stash home directory>/lib` directory (for Stash 2.1 or later).
3. Stop, then restart, Stash. see Starting and stopping Stash.

Connecting Stash to PostgreSQL

This page describes how to connect Stash to a PostgreSQL database.

The overall process for using a PostgreSQL database with Stash is:

- Install PostgreSQL where it is accessible to Stash.
- Create a database and user on the PostgreSQL server for Stash to use.
- Install Stash on Windows, or on Linux or Mac. See Getting started.
- Either:
  - at Stash install time, run the Setup Wizard to connect Stash to the PostgreSQL database, or
  - at a later time, migrate Stash to the PostgreSQL database. See Using the Database Migration Wizard.

It is assumed here that you already have PostgreSQL installed and running. For more information about PostgreSQL installation and operation, refer to the PostgreSQL documentation. For additional information review this page on tuning.

PostgreSQL has the idea of schemas. When you create a PostgreSQL database, a 'public' schema is created and set as the default for that database. It is possible to create a different schema (e.g. 'stash') and set that as the default schema. Stash will use whatever schema is set as the default for the logged-in user. Stash does not provide a way for a user to nominate the schema to use; it uses schema that is set as the PostgreSQL default.

See Supported platforms for the versions of PostgreSQL supported by Stash.

On this page:

Prerequisites
Connect Stash to the PostgreSQL database
Install the JDBC driver

Related pages:

- Connecting Stash to an external database
- Connecting Stash to MySQL
- Connecting Stash to Oracle
- Connecting Stash to SQL Server

Prerequisites

Backup

If you are migrating your Stash data from the HSQL internal database, back up the Stash home directory.

If you are migrating your Stash data from another external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See Data recovery and backups.

Create the Stash database

Before you can use Stash with PostgreSQL, you must:

- Create a role for Stash to use when it connects to the database.
  We strongly recommend that this role be established for Stash's use exclusively; it should not be shared by other applications or people.
- Create a database in which Stash can store its data.
  The database must be configured to use the UTF-8 character set.
  During normal operation, Stash will acquire 25–30 connections to the database. The maximum number of connections is a configurable system property – see Database pool.
- Note that Stash requires the datebase to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be seemingly random connection errors.

Here is an example of how to create a user called **`stashuser`** with password **`jellyfish`**, and a database called **`stash`**, which is configured for use by `stashuser`. Using a PostgreSQL client application like psql or pgAdmin, run the following commands, replacing the user name, password, and database name with your own values.

```
CREATE ROLE stashuser WITH LOGIN PASSWORD 'jellyfish' VALID UNTIL 'infinity';

CREATE DATABASE stash WITH ENCODING='UTF8' OWNER=stashuser CONNECTION LIMIT=-1;
```

If the server that is hosting the PostgreSQL database is not the same server as Stash, then please ensure that the Stash server can connect to the database server. Please also refer to the PostgreSQL documentation on how to set up pg_hba.conf. If the `pg_hba.conf` file is not set properly, remote communication to the PostgreSQL server will fail.

Connect Stash to the PostgreSQL database

You can now connect Stash to the PostgreSQL database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate Stash to PostgreSQL, either from the embedded HSQL database or from another external database.

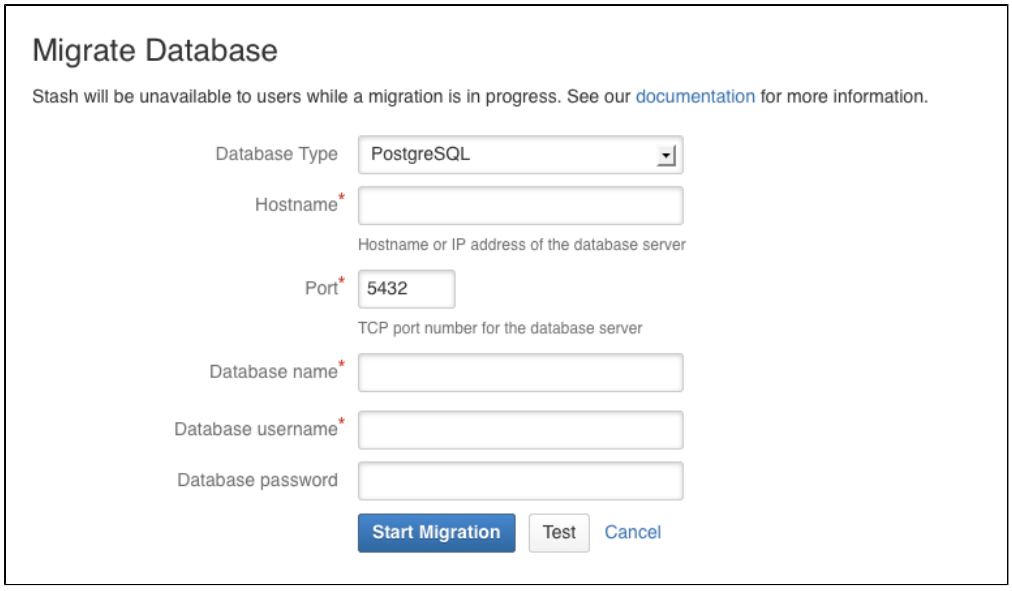
When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **PostgreSQL** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow the instructions in the Stash Setup Wizard.

When migrating to PostgreSQL

1. In the Stash administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **PostgreSQL** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Hostname	The host name or IP address of the computer running the database server.
Port	The TCP port with which Stash can connect to the database server. The default value is the default port that PostgreSQL runs against. You can change that if you know the port that your PostgreSQL instance is using.
Database name	The name of the database that Stash should connect to.
Database username	The username that Stash should use to access the database.
Database password	The password that Stash should use to access the database.



Install the JDBC driver

This section is only relevant to some distributions of Stash, for example if you are running Stash via the Atlassian Plugin SDK, or have built Stash from source.

If the PostgreSQL JDBC driver is *not* bundled with Stash, you will need to download and install the driver yourself.

1. Download the appropriate JDBC driver from the Postgres download site.
2. Copy the downloaded jar file to your `<Stash home directory>/lib` directory (for Stash 2.1 or later).
3. Stop, then restart, Stash. See Starting and stopping Stash.

Connecting Stash to SQL Server

This page describes how to connect Stash to a Microsoft SQL Server database.

The overall process for using a SQL Server database with Stash is:

- Install SQL Server where it is accessible to Stash.
- Create a database and user on the SQL Server server for Stash to use.
- Install Stash on Windows, or on Linux or Mac. See Getting started.
- Either:
  - at Stash install time, run the Setup Wizard to connect Stash to the SQL Server database, or
  - at a later time, migrate Stash to the SQL Server database. See Using the Database Migration Wizard.

It is assumed here that you already have SQL Server installed and running.

- SQL Server documentation is available at http://msdn.microsoft.com/en-us/library/bb545450.aspx.
- JDBC documentation is available at http://msdn.microsoft.com/en-us/library/ms378672.aspx.

See Supported platforms for the versions of SQL Server supported by Stash.

On this page:

- Prerequisites
- Connect Stash to the SQL Server database
- Use Integrated Authentication or 'Windows Authentication Mode' (Optional)
- Install the JDBC driver

Related pages:

- Transitioning from jTDS to Microsoft's JDBC driver
- Connecting Stash to an external database
- Connecting Stash to MySQL
- Connecting Stash to Oracle
- Connecting Stash to PostgreSQL

Prerequisites

Back up your current database

If you are migrating your data from the internal Stash database, back up the Stash home directory.

If you are migrating your Stash data from a different external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See Data recovery and backups.

Create the Stash database

Before you can use Stash with SQL Server, you must set up SQL Server as follows:

Step	Notes
Create a database	e.g. `stash`. Remember this database name for the connection step below.
Set the collation type	This should be case-sensitive, for example, 'SQL_Latin1_General_CP1_CS_AS' (CS = Case Sensitive).
Set the isolation level	Configure the database to use the isolation level, Read Committed with Row Versioning.
Create a database user	e.g. `stashuser`. This database user should **not** be the database owner, but **should** be in the `db_owner` role. It needs to be in this role during setup *and* at all points when Stash is running due to the way Stash interacts with the database. See SQL Server Startup Errors. Remember this database user name for the connection step below.
Set database user permissions	The Stash database user has permission to connect to the database, and to create and drop tables, indexes and other constraints, and insert and delete data, in the newly-created database.
Enable TCP/IP	Ensure that TCP/IP is enabled on SQL Server and that SQL Server is listening on the correct port (which is 1433 for a default SQL Server installation). Remember this port number for the connection step below.
Check the authentication mode	Ensure that SQL Server is operating in the appropriate authentication mode. By default, SQL Server operates in 'Windows Authentication Mode'. However, if your user is not associated with a trusted SQL connection, 'Microsoft SQL Server, Error: 18452' is received during Stash startup, and you will need to change the authentication mode to 'Mixed Authentication Mode'. *Stash instances running on Windows are also able to support SQL Server databases running in 'Windows Authentication Mode'. This is described at the bottom of this page and it has to be manually configured: Connecting Stash to SQL Server - Use Integrated Authentication (Optional)*
Check that SET NOCOUNT is off	Ensure that the SET NOCOUNT option is turned off. You can do that in SQL Server Management Studio as follows: 1. Navigate to **Tools** > **Options** > **Query Execution** > **SQL Server** > **Advanced**. Ensure that the **SET NOCOUNT** option is cleared. 2. Now, go to the **Server** > **Properties** > **Connections** > **Default Connections** properties box and clear the **no count** option.

Note that Stash will generally require about 25–30 connections to the database.

Note also that Stash requires the datebase to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be seemingly random connection errors.

Here is an example of how to create and configure the SQL Server database from the command line. When Stash and SQL Server run on the same physical computer (accessible through `localhost`), run the following commands (replacing `stashuser` and `password` with your own values):

```
SQL Server> CREATE DATABASE stash
SQL Server> GO
SQL Server> USE stash
SQL Server> GO
SQL Server> ALTER DATABASE stash SET ALLOW_SNAPSHOT_ISOLATION ON
SQL Server> GO
SQL Server> ALTER DATABASE stash SET READ_COMMITTED_SNAPSHOT ON
SQL Server> GO
SQL Server> ALTER DATABASE stash COLLATE SQL_Latin1_General_CP1_CS_AS
SQL Server> GO
SQL Server> SET NOCOUNT OFF
SQL Server> GO
SQL Server> USE master
SQL Server> GO
SQL Server> CREATE LOGIN stashuser WITH PASSWORD=N'password',
DEFAULT_DATABASE=stash, CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
SQL Server> GO
SQL Server> ALTER AUTHORIZATION ON DATABASE::stash TO stashuser
SQL Server> GO
```

This creates an empty SQL Server database with the name `stash`, and a user that can log in from the host that Stash is running on who has full access to the newly created database. In particular, the user should be allowed to create and drop tables, indexes and other constraints.

Connect Stash to the SQL Server database

You can now connect Stash to the SQL Server database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate to SQL Server, either from the embedded database or from another external database.

When running the Setup Wizard at install time

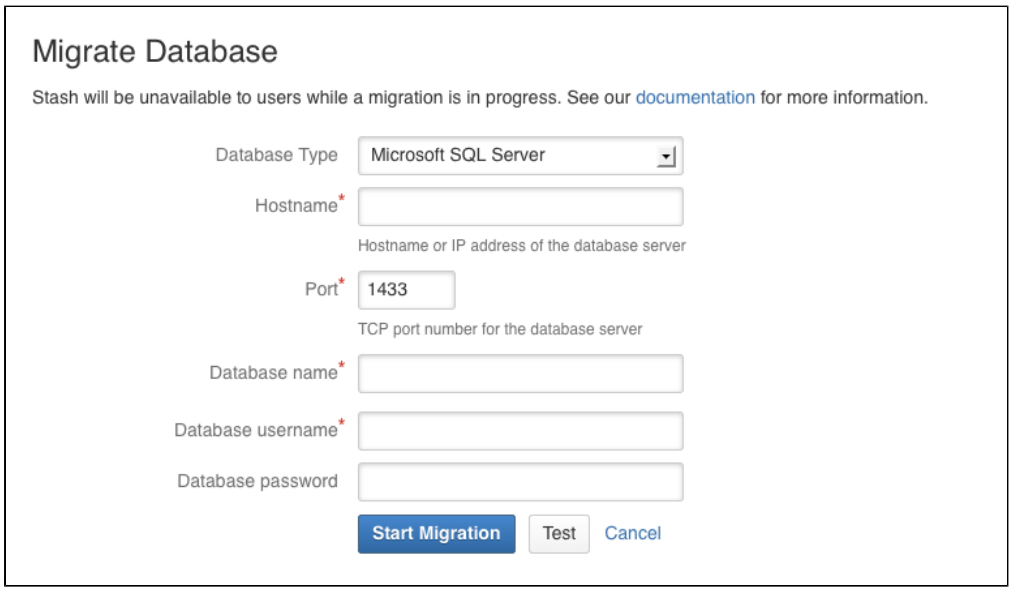
1. Select **External** at the 'Database' step.
2. Select **SQL Server** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow the instructions in the Stash Setup Wizard.

When migrating to SQL Server

1. In the Stash administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **SQL Server** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Hostname	The host name or IP address of the computer running the database server.

Port	The TCP port with which Stash can connect to the database server. The default value of 1433 is the default port that SQL Server runs against. You can change that if you know the port that your SQL Server instance is using.
Database name	The name of the database that Stash should connect to.
Database username	The username that Stash should use to access the database.
Database password	The password that Stash should use to access the database.



> **Named Instances**
> If you have a named instance on your server, you will need to manually edit the `stash-config.properties` file as described on the Connecting to named instances in SQL Server from Stash Knowledge Base article.

Use Integrated Authentication or 'Windows Authentication Mode' (Optional)

Windows authentication is only available for Stash instances running on Windows. It cannot be used on Linux because Microsoft does not provide shared objects for it. You will either need to run Stash on Windows, allowing you to use Windows security, or you will need to enable mixed-mode authentication for SQL Server if you are running Stash on Linux. Unfortunately, there are no other options at this time.

Integrated authentication uses a native DLL to access the credentials of the logged-in user to authenticate with SQL Server. The native DLLs for both 32- and 64-bit systems are included in the distribution; there is no need to download the entire package from Microsoft.

Stash does not currently support configuring the system to use integrated authentication from the UI (Vote for it! STASH-3035 - Add support for integrated authentication for Microsoft SQL Server OPEN). This means you can't currently

migrate to SQL Server with integrated authentication, nor can you configure Stash to use SQL Server with integrated authentication during initial setup. However, if Stash has already been configured to use SQL Server (for example, when the Setup Wizard was run at first use), you can enable integrated authentication by directly modifying Stash's configuration, as follows:

1. Based on the JVM being used to run Stash, rename either the `x64` or `x86` DLL to `sqljdbc_auth.dll` in `lib/native`. Note that running on Windows x64 does *not* require the use of the `x64` DLL; you should only use the `x64` DLL if you are also using a 64-bit JVM.
2. In `setenv.bat`, a `JVM_LIBRARY_PATH` variable has already been defined. Simply remove the leading `rem` . Note that if you are putting the native DLL in an alternative location, you may need to change the value to point to your own path. The value of the `JVM_LIBRARY_PATH` variable will automatically be included in the command line when Tomcat is run using `start-stash.bat`.
3. Edit the `%STASH_HOME%\stash-config.properties` file to include `;integratedSecurity=true` in the `jdbc.url` line. Note that `jdbc.user` and `jdbc.password` will no longer be used to supply credentials *but they must still be defined* – Stash will fail to start if these properties are removed.
4. Ensure the Stash process or service is running as the correct user to access SQL Server. (Note that this user is generally a Windows Domain User Account, but should not be a member of any administrators groups, that is local, domain, or enterprise.)

> It is also possible to configure integrated authentication over Kerberos, rather than using the native DLLs. Details for that are included in the JDBC documentation.

Install the JDBC driver

This section is only relevant to some distributions of Stash, for example if you are running Stash via the Atlassian Plugin SDK, or have built Stash from source.

If the SQL Server JDBC driver is *not* bundled with Stash, you will need to download and install the driver yourself.

1. Download the appropriate JDBC driver from the Microsoft download site.
2. Install the driver file to your `<Stash home directory>/lib` directory (for Stash 2.1 or later).
3. Stop, then restart, Stash. See Starting and stopping Stash.

If Stash was configured to use Microsoft SQL Server by manually entering a JDBC URL, please refer to this guide.

Transitioning from jTDS to Microsoft's JDBC driver

This page describes how to change from using jTDS to using the Microsoft SQL Server JDBC driver to access Microsoft SQL Server.

What do I have to do?

If Stash was configured to use Microsoft SQL Server by following the steps outlined in Connecting Stash to SQL Server, *no change is necessary* . However, If Stash was configured to use Microsoft SQL Server by **manually entering a JDBC URL**, the system will lock on startup if the driver class and URL are not manually updated.

How to proceed

In the Stash home directory, `stash-config.properties` must be edited to change the JDBC driver and URL. The existing configuration should look similar to this:

```
jdbc.driver=net.sourceforge.jtds.jdbc.Driver
jdbc.url=jdbc:jtds:sqlserver://localhost:1433;databaseName=stash;
jdbc.user=stashuser
jdbc.password=secretpassword
```

⚠ The JDBC URL above is in the format constructed by Stash when Connecting Stash to SQL Server and will

automatically be updated to a URL compatible with Microsoft's driver, with no change required on the administrator's part. If the URL contains additional properties, such as domain=, it will need to be manually updated.

To use Microsoft's SQL Server driver, the settings above would be updated to this:

```
jdbc.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
jdbc.url=jdbc:sqlserver://localhost:1433;databaseName=stash;
jdbc.user=stashuser
jdbc.password=secretpassword
```

The exact values to use in the new URL are beyond the scope of this documentation; they must be chosen based on the jTDS settings they are replacing.

Additional Information for the curious

The new JDBC driver class is: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

The JDBC URL format for the jTDS driver is documented on SourceForge at http://jtds.sourceforge.net/faq.html#urlFormat.

The JDBC URL format for Microsoft's SQL Server driver is documented on MSDN at http://msdn.microsoft.com/en-us/library/ms378428.aspx, with documentation for additional properties at http://msdn.microsoft.com/en-us/library/ms378988.aspx.

Why change drivers?

Click here to find all the technical details...

Recent releases of Hibernate, which Stash uses to simplify its persistence layer, have introduced a requirement that the JDBC drivers and connection pools used be JDBC4-compliant. JDBC4 was introduced with Java 6.

The jTDS driver used by releases prior to Stash 2.1 is a JDBC3 driver, compatible with Java 1.3, and therefore cannot be used with newer versions of Hibernate. While jTDS 1.3.0 implements JDBC4, and JDBC4.1 which is provided by Java 7, it *requires* a Java 7 runtime environment. Upgrading Stash to that version was a non-starter, as it would require raising the minimum Java version for the product to Java 7.

Instead, the decision was made to replace jTDS with Microsoft's own SQL Server driver. Microsoft's driver is actively maintained, where jTDS is only recently seeing its first updates in over 3 years, and supports all the features of SQL Server, including SQL Server 2012.

Stash attempts to automatically update jTDS JDBC URLs to values compatible with Microsoft's JDBC driver. However, for installations using custom JDBC URLs–for example, to use domain authentication–such automatic updating is not possible; the URL, which was manually entered, must be manually updated.

Migrating Stash to another server

This page describes how to move your Stash installation from one physical machine to a different machine. For most scenarios, the overall procedure involves the following 4 steps, although your situation may not require all of these:

1. Prepare for the migration.
2. Move the Stash data.
3. Move the Stash installation to the new location, and update the value of the `STASH_HOME` environment variable.
4. Update the Stash stash-config.properties file. This will be necessary if you were unable to use the Migration Wizard in Step 2.

See also the Stash upgrade guide. You can upgrade Stash either before or after you migrate Stash. This page *does not* describe any aspect of the upgrade procedure.

On this page:

1. Prepare for the migration
2. Move the Stash data to a different machine
3. Move Stash to a different machine
4. Update the Stash configuration

Related pages:

- Supported platforms
- Connecting Stash to an external database
- Stash upgrade guide

1. Prepare for the migration

In preparation for migrating Stash to another server, check that you have done the following:

- Confirm that the operating system, database, other applicable platforms and hardware on the new machine will comply with the requirements for Stash.
- Check for any known migration issues in the Stash Knowledge Base.
- Alert users to the forthcoming Stash service outage.
- Ensure that users will not be able to update existing Stash data during the migration. You can do this by temporarily changing the access permissions for Stash.
- Make sure you have created a user in Stash (not in your external user directory) that has System Administrator global permissions so as to avoid being locked out of Stash in case the new server does not have access to your external user directory.

2. Move the Stash data to a different machine

This section gives a brief overview of how to move the Stash data to a different machine. You do not need to do anything in this section if you will continue to use the embedded database - the Stash data is moved when you move the Stash installation.

The Stash data includes the data directories (including the Git repositories), log files, installed plugins, temporary files and caches.

You can move the Stash data:

- from the embedded database to a supported external DBMS.
- to another instance of the same DBMS.
- from one DBMS to another supported DBMS (for example, from MySQL to PostgreSQL).

You can also move the actual DBMS. Atlassian recommends that for large installations, Stash and the DBMS run on separate machines.

There are 2 steps:

1. Create and configure the DBMS in the new location. Please refer to Connecting Stash to an external database, and the relevant child page, for more information.
2. Either:
    - If the new location is currently visible to Stash, use the Stash Database Migration Wizard. Please refer to Connecting Stash to an external database, and the relevant child page, for more information.
    - If the new location is not currently visible to Stash (perhaps because you are moving to a new hosting provider), you need to perform a database export and then import the backup to the new DBMS. Please refer to the vendor documentation for your DBMS for detailed information.
      You will also need to update the stash-config.properties file in the `<Stash home directory>` as described below.

3. Move Stash to a different machine

This section describes moving the Stash installation to a different machine.

1. Stop Stash. See Starting and stopping Stash.

2. Make an archive (such as a zip file) of the Stash home directory. The home directory contains data directories (including the Git repositories), log files, installed plugins, SSH fingerprints, temporary files and caches. The home directory location is defined:
    - on Windows, by the `STASH_HOME` environment variable, or by the `STASH_HOME` line of `<Stash installation directory>/bin/setenv.bat`.
    - on Linux and Mac, by the `STASH_HOME` line of `<Stash installation directory>/bin/setenv.sh`.
3. Copy the archive of the Stash home directory to the new machine and unzip it to its new location there.
    - For production environments the Stash home directory should be secured against unauthorised access. See Stash home directory.
    - When moving the Stash home directory from Windows to Linux or Mac, make sure that the files in the `<Stash home directory>/git-hooks` and `<Stash home directory>/shared/data/repositories/<repoID>/hooks` directories have the executable file permission set.
4. Set up an instance of Stash in the new location by doing one of the following:
    - Make an archive of the old Stash installation directory and copy it across to the new machine.
    - Install the same version of Stash from scratch on the new machine.
5. Redefine the value for `STASH_HOME`, mentioned in Step 2. above, in the new `<Stash installation directory>`, using the new location for your copied home directory. See Stash home directory for more information.
6. If you are continuing to use the Stash embedded database, or you used the Migration Wizard to move the Stash data, you should now be able to start Stash on the new machine and have all your data available. See Starting and stopping Stash. Once you have confirmed that the new installation of Stash is working correctly, revert the access permissions for Stash to their original values.
7. If you moved the Stash data by performing a database export and import, carry on to Step 4. below to update the stash-config.properties file in the `<Stash home directory>`.

4. Update the Stash configuration

If you moved the Stash data by performing a database export, you must update the stash-config.properties file in the `<Stash home directory>` with the changed configuration parameters for the database connection.

The configuration parameters are described in Stash config properties.

Once the configuration parameters are updated, you should be able to start Stash on the new machine and have all your data available. See Starting and stopping Stash. Once you have confirmed that the new installation of Stash is working correctly, revert the access permissions for Stash to their original values.

Specifying the base URL for Stash

This is the base URL for this installation of Stash. All links *which are not from a web request* (for example in Stash email notifications), will be prefixed by this URL. If you are experiencing trouble with setting an `https` base URL, please ensure you have configured Tomcat with SSL correctly.

To specify Stash's base URL:

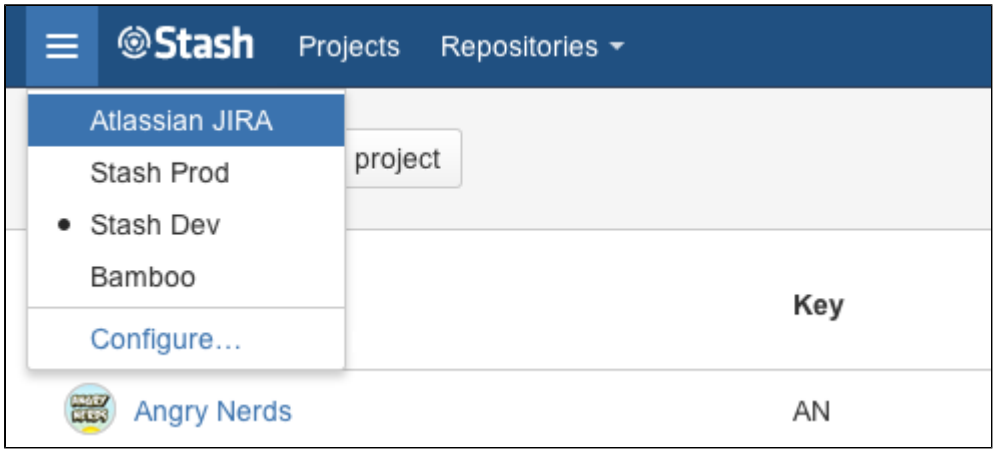
1. In the Stash administration area, click **Server settings** (under 'Settings').
2. In the **Base URL** field, type the URL address of your Stash server (for example, "https://stash.mycompany.com").
3. Click **Save**.

Related pages:

- Administering Stash

Configuring the application navigator

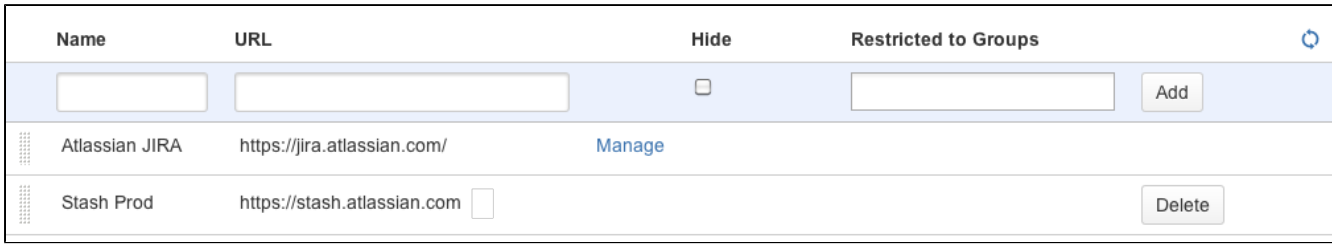
The application navigator, on the left of the Stash header, allows you to switch to your other applications, such as Atlassian JIRA and Bamboo – or any other web application – all from the Stash header:



Users only see the application navigator when links are set up – if there are no links, only administrators can see it.

Stash administrators can configure which apps appear in the navigator – just click **Configure** in the application navigator, or go to the Stash admin area and click **Application Navigator**:

- Linked applications are automatically configured in the application navigator, and can't be deleted. Click **M anage** to configure those in the source application.
- Specify new links, as required by your users, by entering a **Name** and **URL**.
- Restrict the visibility of links to particular user groups, or hide the link completely. Click in a row, under the Groups column header, to edit those properties for existing rows.
- Use the 'handles' at the left to change the link order when seen in Stash.



Managing add-ons

An add-on is an installable component that supplements or enhances the functionality of Stash in some way. For example, the Custom Navigation Plugin enables you to configure custom navigation tabs specific to a repository. Other add-ons are available for adding graphs to Stash, importing SVN source control projects into Stash, and accessing Atlassian support from Stash.

Stash comes with many pre-installed add-ons (called system add-ons). You can install more add-ons, either by acquiring the add-on from the Atlassian Marketplace or by uploading it from your file system. This means that you can install add-ons that you have developed yourself. For information about developing your own add-ons for Stash, see the Stash Developer Documentation.

On this page:

- About the Universal Plugin Manager (UPM)
- Administering add-ons in Stash Server
- Add-ons for Stash Data Center

About the Universal Plugin Manager (UPM)

You administer add-ons for Stash using the Universal Plugin Manager (UPM). The UPM is itself an add-on that

exposes add-on administration pages in the Stash Administration Console. UPM works across Atlassian applications, providing a consistent interface for administering add-ons in Stash, Crucible, Confluence, FishEye, JIRA and Bamboo.

UPM comes pre-installed in recent versions of all Atlassian applications, so you do not normally need to install it yourself. However, like other add-ons, the UPM software is subject to regular software updates. Before administering add-ons in Stash, therefore, you should verify your version of the UPM and update it if needed.

Administering add-ons in Stash Server

You can update UPM, or any add-on, from the UPM's own add-on administration pages. Additionally, you can perform these tasks from the UPM administration pages:

- Install or remove add-ons
- Configure add-on settings
- Discover and install new add-ons from the Atlassian Marketplace
- Enable or disable add-ons and their component modules

It shows only those plugins that are supported in your version of the product, so that you do not install incompatible plugins.

If the add-on request feature is enabled in your Atlassian application, non-administrative users can also discover add-ons on the Atlassian Marketplace. Instead of installing the add-ons, however, these users have the option of requesting the add-ons from you, the administrator of the Atlassian application.

For more information on administering the add-on request feature or performing other common add-on administration tasks, see the Universal Plugin Manager documentation. For an end-user's view of requesting add-ons in Stash, see Requesting add-ons.

Add-ons for Stash Data Center

Installing, and managing, add-ons for Stash Data Center is done in the same way as for Stash Server, as described above. The only requirement is that the add-on is Data Center-compatible – see Stash Data Center Add-ons for compatibility information.

You can install an add-on from any cluster node. The add-on is stored on the shared file system for the Stash Data Center, and made available to all nodes in the cluster.

POST service webhook for Stash

Repository administrators can add a POST service to a repository. Stash POSTs to the service URL you specify.

You can use an URL with the following format:

```
https://server:port/path/
```

The service receives a POST whenever the user pushes to the repository.

The content type header of the POST has an `application/json` type. The content is a JSON payload that represents the repository push.

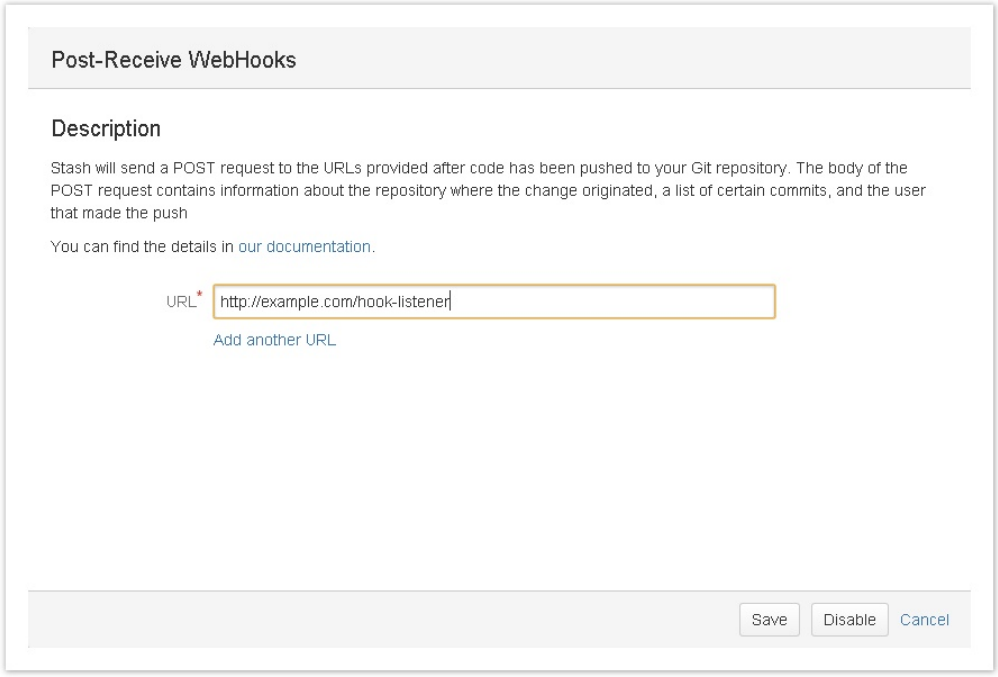
Setting up the POST service

You can either set up the POST service manually or you can write a service to automate this. You would write a service if you are integrating an application with Stash.

Set up in the repository settings

1. Go to the repository's settings.

2. Click **Hooks** in the left-hand navigation.
3. Click **Enable** for the 'Post-Receive Webhooks' item. You can add up to 5 URLs for where Stash should send its update messages:



4. Press **Save**.

POST data

When a user pushes to a repository, Stash POSTs to the URL you provided. The body of the POST request contains information about the repository where the change originated, a list of recent commits, and the name of the user that made the push.

Example of payload

This is an example of a push that contains one commit that changes 2 files (*pom.xml*) in folders *iridium-common* and *iridium-magma*.

JSON Payload


```
{
   "repository":{
      "slug":"iridium-parent",
      "id":11,
      "name":"iridium-parent",
      "scmId":"git",
      "state":"AVAILABLE",
      "statusMessage":"Available",
      "forkable":true,
      "project":{
         "key":"IR",
         "id":21,
         "name":"Iridium",
         "public":false,
         "type":"NORMAL",
         "isPersonal":false
      },
```

```
      "public":false
   },
   "refChanges":[
      {
         "refId":"refs/heads/master",
         "fromHash":"2c847c4e9c2421d038fff26ba82bc859ae6ebe20",
         "toHash":"f259e9032cdeb1e28d073e8a79a1fd6f9587f233",
         "type":"UPDATE"
      }
   ],
   "changesets":{
      "size":1,
      "limit":100,
      "isLastPage":true,
      "values":[
         {
            "fromCommit":{
               "id":"2c847c4e9c2421d038fff26ba82bc859ae6ebe20",
               "displayId":"2c847c4"
            },
            "toCommit":{
               "id":"f259e9032cdeb1e28d073e8a79a1fd6f9587f233",
               "displayId":"f259e90",
               "author":{
                  "name":"jhocman",
                  "emailAddress":"jhocman@atlassian.com"
               },
               "authorTimestamp":1374663446000,
               "message":"Updating poms ...",
               "parents":[
                  {
                     "id":"2c847c4e9c2421d038fff26ba82bc859ae6ebe20",
                     "displayId":"2c847c4"
                  }
               ]
            },
            "changes":{
               "size":2,
               "limit":500,
               "isLastPage":true,
               "values":[
                  {
                     "contentId":"2f259b79aa7e263f5829bb6e98096e7ec976d998",
                     "path":{
                        "components":[
                           "iridium-common",
                           "pom.xml"
                        ],
                        "parent":"iridium-common",
                        "name":"pom.xml",
                        "extension":"xml",
                        "toString":"iridium-common/pom.xml"
                     },
                     "executable":false,
                     "percentUnchanged":-1,
                     "type":"MODIFY",
                     "nodeType":"FILE",
                     "srcExecutable":false,
                     "link":{

"url":"/projects/IR/repos/iridium-parent/commits/f259e9032cdeb1e28d073e8a79a1fd6f95
87f233#iridium-common/pom.xml",
                        "rel":"self"
```

```
                    }
                },
                {
                    "contentId":"2f259b79aa7e263f5829bb6e98096e7ec976d998",
                    "path":{
                        "components":[
                            "iridium-magma",
                            "pom.xml"
                        ],
                        "parent":"iridium-magma",
                        "name":"pom.xml",
                        "extension":"xml",
                        "toString":"iridium-magma/pom.xml"
                    },
                    "executable":false,
                    "percentUnchanged":-1,
                    "type":"MODIFY",
                    "nodeType":"FILE",
                    "srcExecutable":false,
                    "link":{

"url":"/projects/IR/repos/iridium-parent/commits/f259e9032cdeb1e28d073e8a79a1fd6f95
87f233#iridium-magma/pom.xml",
                        "rel":"self"
                    }
                }
            ],
            "start":0,
            "filter":null
        },
        "link":{

"url":"/projects/IR/repos/iridium-parent/commits/f259e9032cdeb1e28d073e8a79a1fd6f95
87f233#iridium-magma/pom.xml",
            "rel":"self"
        }
    }
  ],
  "start":0,
```

```
        "filter":null
    }
}
```

Properties

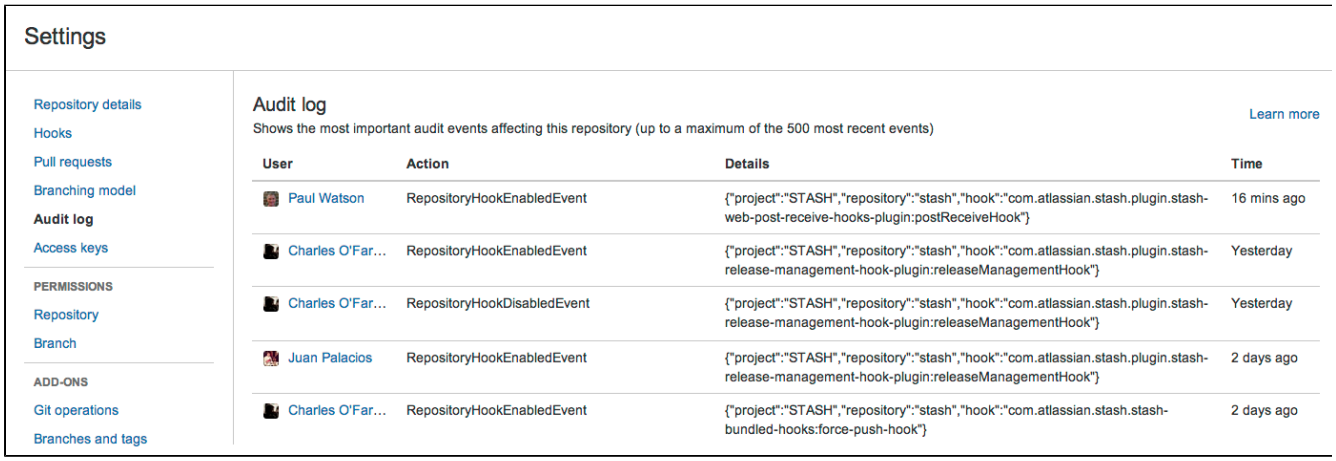
Some of the system-wide properties for the Webhook Plugin can be overridden in the Stash configuration file. The available properties are listed in Stash config properties.

Audit logging in Stash

Stash comes with an internal audit system enabled by default at installation. The audit system is intended to give administrators an insight into the way Stash is being used. The audit system could be used to identify authorized and unauthorized changes, or suspicious activity over a period of time.

Viewing recent events

Stash administrators and system administrators can see a list of recent events for each project and repository in the 'Audit log' view. This is found in the 'Settings' for a project or repository, and shows only the most important audit events.



The audit log displays a subset of the events recorded in the log file and is kept to a configurable maximum size (the default is 500 events). See Audit events in Stash for more details.

Accessing the audit log file

The full audit log file records a wide range of events in Stash. See Audit events in Stash for a list of these.
The volume of events that are logged is coarsely configurable by changing a Stash server setting. See Stash config properties for more details.
You can find the log file in the `<Stash home directory>/log/audit` directory.
The log file will roll daily and also when it grows past a maximum size of 25 MB. There is a limit (currently 100) to the number of rolled files that Stash will keep. When the limit is reached, the oldest file is deleted each day.
⚠ Note that you will need to back up the log files before they are removed, if your organisation needs to keep copies of those.

Configuring audit logging

There are various system properties that can be used to configure audit logging in Stash.

Audit events in Stash

The auditing component of Stash will log many different events that occur when Stash is being used. The events have been assigned priorities based on how important they are – these priorities can be used to

control how much information is added to the audit log file. For example, if you have a server under high load and no need for auditing, you may wish to turn audit logging off by setting it to `NONE` - see the audit log config properties.

On this page:

- Server level events
- User management events
- Permission events
- Project events
- Repository events
- Pull request events
- Plugin events
- SSH key events

Server level events

Event	Description	Priority
ApplicationConfigurationChangedEvent	The server configuration has changed e.g. the display name or the base url.	HIGH
BackupEvent	Audited at the beginning and the end of a system backup.	HIGH
LicenseChangedEvent	The server license has changed.	HIGH
MailHostConfigurationChangedEvent	The servers mail host has changed (used to send email notifications).	HIGH
MigrationEvent	Audited at the beginning and the end of a database migration.	HIGH
ServerEmailAddressChangedEvent	The server email address has changed (used in email notifications).	HIGH
TicketRejectedEvent	Certain resources (e.g. the Git processes) are throttled, when tickets are rejected (e.g. too many Git processes are in use) this event is fired.	LOW

User management events

Event	Description	Priority
DirectoryCreatedEvent	Occurs when a new directory is created.	HIGH
DirectoryDeletedEvent	Occurs when a new directory is deleted.	HIGH
GroupCreatedEvent	Occurs when a new groupis created in the internal directory.	HIGH
GroupUpdatedEvent	Occurs when a new group is updated (not when membership changes) in the internal directory.	HIGH
GroupDeletedEvent	Occurs when a new group is deleted from the internal directory.	HIGH

GroupMembershipCreatedEvent	Occurs when a user is added to a group in the internal directory.	HIGH
GroupMembershipDeletedEvent	Occurs when a user is removed from a group in the internal directory.	HIGH
UserAuthenticatedEvent	Occurs when a user is successfully authenticated (logged in).	LOW
UserAuthenticationFailedInvalidAuthenticationEvent	Occurs whenever a user fails to authenticate. Note that this can occur frequently in Stash whenever a command line CLI is used as the initial URL provided to Stash contains a username but no password, which is rejected by Crowd.	MEDIUM
UserCreatedEvent	Occurs when a user is created in the internal directory.	HIGH
UserCredentialUpdatedEvent	Occurs when a user changes password in the internal directory.	HIGH
UserDeletedEvent	Occurs when a user is deleted from the internal directory.	HIGH
UserRenamedEvent	Occurs when the username of a user is changed in the internal directory.	HIGH

Permission events

in the table below indicates that the event is visible in the recent audit log screen for the project or repository.

Event	Description	Priority
GlobalPermissionGrantedEvent	Occurs when a user or group is granted a global permission (e.g. create project).	HIGH
GlobalPermissionRevokedEvent	Occurs when a user or group has a global permission revoked.	HIGH
ProjectPermissionGrantedEvent	Occurs when a user or group is granted a permission for a specific project.	HIGH
ProjectPermissionRevokedEvent	Occurs when a user or group has a permission for a specific project revoked.	HIGH
RepositoryPermissionEvent	Occurs when a user or group has a permission for a specific repository altered.	HIGH
RestrictedRefEvent	Children of this event are fired when a restricted ref is altered.	HIGH

Project events

in the table below indicates that the event is visible in the recent audit log screen for the project.

Event	Description	Priority
ProjectAvatarUpdatedEvent	Raised when a project avatar has been successfully updated.	LOW
ProjectCreatedEvent	Raised when a project is created.	HIGH
ProjectCreationRequestedEvent	Raised just before a project is created; can be cancelled.	LOW
ProjectModifiedEvent	Raised when a project has been successfully updated (e.g. the project name).	HIGH
ProjectModificationRequestedEvent	Raised just before a project is updated; can be cancelled.	LOW
ProjectDeletedEvent	Raised when a project is deleted.	HIGH
ProjectDeletionRequestedEvent	Raised just before a project is deleted; can be cancelled.	LOW

Repository events

in the table below indicates that the event is visible in the recent audit log screen for the project or repository.

Event	Description	Priority
RepositoryAccessedEvent	Raised when a repository is accessed by a user. Stash currently only fires this event selectively - when users hit a repository page.	LOW
RepositoryCreatedEvent	Raised when a repository is created.	MEDIUM
RepositoryCreationFailedEvent	Raised when an attempt to create a repository fails.	LOW
RepositoryCreationRequestedEvent	Raised just before a repository is created; can be cancelled.	LOW
RepositoryForkedEvent	Raised when a repository is forked successfully.	MEDIUM
RepositoryForkFailedEvent	Raised when an attempt to fork a repository fails.	LOW
RepositoryForkRequestedEvent	Raised just before a repository is forked; can be cancelled.	LOW
RepositoryDefaultBranchModifiedEvent	Raised when the default branch of a repository is reconfigured (typically through repository settings).	LOW
RepositoryDeletedEvent	Raised when a repository is deleted.	HIGH
RepositoryDeletionRequestedEvent	Raised just before a repository is deleted; can be cancelled.	LOW
RepositoryOtherReadEvent	Raised when the server uploads a pack file to the client via HTTP.	LOW

RepositoryOtherWriteEvent	Raised when the server receives a pack file from the client via HTTP.	LOW
RepositoryPullEvent	Raised when a Git client pulls from a repository (only when new content is sent to the client).	LOW
RepositoryPushEvent	Raised when a Git client pushed to a repository.	LOW

Pull request events

Event	Description	Priority
PullRequestEvent	Fired at different points in the pull request lifecycle (declined, merged, opened, reopened, rescoped [code updated], updated, approved, unapproved, participants updated).	LOW

Plugin events

See this plugin documentation for details of when these events below are triggered.

Event	Description	Priority
PluginDisabledEvent	Occurs when a plugin has been disabled, either by the system or a user.	MEDIUM
PluginEnabledEvent	Occurs when a plugin has been enabled, either by the system or a user.	MEDIUM
PluginModuleDisabledEvent	Occurs when a plugin module has been disabled, either by the system or a user.	MEDIUM
PluginModuleEnabledEvent	Occurs when a plugin module has been enabled, either by the system or a user.	MEDIUM
PluginModuleUnavailableEvent	Signifies a plugin module is now unavailable outside the usual installation process.	MEDIUM
PluginUninstalledEvent	Occurs when a plugin is explicitly uninstalled (as opposed to as part of an upgrade).	MEDIUM
PluginUpgradedEvent	Signifies that a plugin has been upgraded at runtime.	MEDIUM
PluginContainerUnavailableEvent	Occurs when the container of a plugin is being shutdown, usually as a result of the server being stopped.	LOW
PluginModuleAvailableEvent	Signifies that a plugin module is now available outside the usual installation process.	LOW
PluginFrameworkStartedEvent	Signifies that the plugin framework has been started and initialized.	LOW

SSH key events

in the table below indicates that the event is visible in the recent audit log screen for the project or repository.

Event	Description	Priority
SshKeyCreatedEvent	Occurs when: • an SSH key is added for a user, or • an access key is added to a project or repository and the key has not yet been used on any other projects or repositories.	HIGH
SshKeyDeletedEvent	Occurs when: • an SSH key is removed from a user, or • an access key is removed from a project or repository and it is no longer being used by any other projects or repositories.	HIGH
SshKeyAccessGrantedEvent	Occurs when an access key is given access to a project or repository.	HIGH
SshKeyAccessRevokedEvent	Occurs when an access key is removed from a project or repository.	HIGH

Advanced actions

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

In this section:

- Running the Stash installer
- Automated setup for Stash
- Starting and stopping Stash
- Install Stash from an archive file
- Running Stash as a Linux service
- Running Stash as a Windows service
- Stash config properties
- Proxying and securing Stash
- Enabling SSH access to Git repositories in Stash
- Using diff transcoding in Stash
- Changing the port that Stash listens on
- Moving Stash to a different context path
- Running Stash with a dedicated user
- Stash debug logging
- Data recovery and backups
- Lockout recovery process
- Scaling Stash
- High availability for Stash
- Clustering with Stash Data Center
- Enabling JMX counters for performance monitoring
- Getting started with Stash and AWS
- Disabling HTTP(S) access to Git repositories in Stash

Related pages:

- Administering Stash
- Supported platforms
- Stash FAQ

Running the Stash installer

This page provides information about running the Stash installer. For high-level information about installing and using Stash see Getting started.

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

The installer will:

- Install Stash into a fresh directory, even if you have an earlier version of Stash installed.
- Install a supported version of the Java JRE, which is only available to Stash, if necessary.
- Launch Stash when it finishes.

Additional services provided by the installer, and described on this page, are:

- Installing Stash as a service
- Running the installer in console and unattended modes

Note that you can also automate the Stash Setup Wizard so that a Stash instance can be completely provisioned automatically – see Automated setup for Stash.

Running the installer

Download the Stash installer from the Atlassian download site.

On Linux, you need to set the executable flag on the installer file before running it:

```
chmod +x atlassian-stash-x.x.x-x64.bin
```

Run the installer, and follow the installation wizard:

Install Stash as a service

On Linux and Windows systems, the installer can install Stash as a service (although not when upgrading an existing instance of Stash).

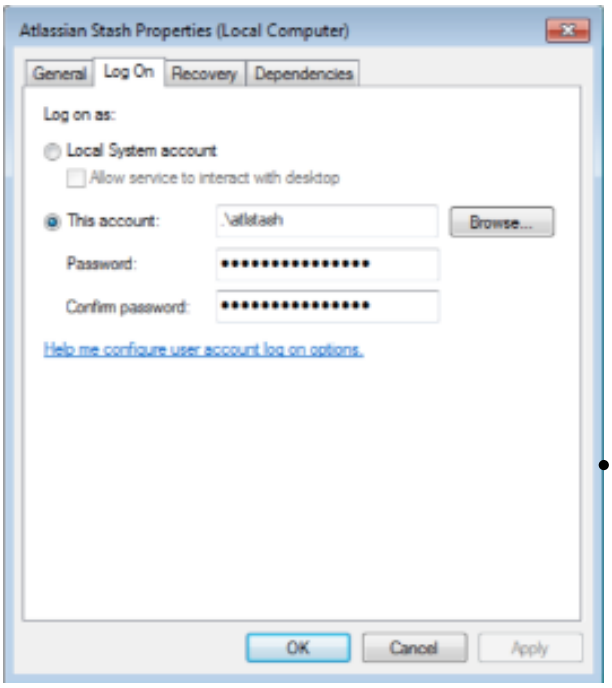
A service account named 'atlstash' will be created.

On Linux

- The 'atlstash' account will be a locked account (it cannot be used to log in to the system).
- The `init.d` script will be linked to run levels 2, 3, 4 and 5. If you wish to change this, you will need to configure it manually.

On Windows

- The installer generates a password for the service account. As a Windows administrator, you can update the account password if you wish to own the account. You'll also need to update the log on credentials for the service:



- The 'atlstash' account will be configured with SeServiceLogonRight so that it can be used by the service. It will also be configured with `SeDenyBatchLogonRight`, `SeDenyInteractiveLogonRight`, `SeDenyNetworkLogonRight`, and `SeDenyRemoteInteractiveLogonRight` so that it cannot be used to log into the machine.
- For Windows services created using the Stash installer, the Stash home directory location (defined by the `STASH_HOME` variable) is configured as a Tomcat Service JVM option. To change it see Change STASH_HOME when installed as a Windows service.

Console and unattended mode

The Stash installer has three modes:

- GUI mode: the default mode for the installer is to display a GUI installer.
- Console mode: if the installer is invoked with the `-c` argument, the interaction with the user is performed in the terminal from which the installer was invoked.
- Unattended mode: if the installer is invoked with the `-q` argument, there is no interaction with the user and the installation is performed automatically with the default values.

Unattended mode also allows you to supply a response file with a `-varfile` option, to supply answers for all questions that are used instead of the defaults. An example response file is:

Example response file

```
 // Should Stash be installed as a Service? Must be ADMIN (default: true if the
process is running with administrator rights, false otherwise). If false, the home
and installation directories must be specified to point to directories owned by the
user
app.install.service$Boolean=true

// The ports Stash should bind to (defaults: portChoice=default, httpPort=7990,
serverPort=8006)
portChoice=custom
httpPort=7990
serverPort=8006


// Path to the Stash HOME directory (default: /var/atlassian/application-data/stash
if the process is running with administrator rights,
~/atlassian/application-data/stash otherwise)
app.stashHome=/var/atlassian/application-data/stash


// The target installation directory (default: /opt/atlassian/stash/<VERSION> if
the process is running with administrator rights, ~/atlassian/stash/<VERSION>
otherwise)
app.defaultInstallDir=/opt/atlassian/stash/<VERSION>
```

On Windows, you must tell CMD/PowerShell to wait for the install4j process to use console/unattended mode:

```
start /wait installer.exe -c
```

On Mac OS X, mount the disk image, then run the Java stub in the installer using this command:

```
/Volumes/Stash/Stash\ X.X.X\Installer.app/Contents/MacOS/JavaApplicationStub
-options
```

where `X.X.X` is the version of Stash, and `-options` can include `-c` or `-q`, and `-varfile` followed by the path to the response file.

For more information see the install4j documentation.

Further reading

- Using Stash in the enterprise

Automated setup for Stash

This page describes how the Stash Setup Wizard can be completed automatically, that is without the need for manual interaction in the browser. See Getting started for an outline of the tasks that the Setup Wizard assists with when setting up Stash manually.

You might want to configure this when automating the provisioning of Stash Data Center, for example. Of

course, you'll need to configure provisioning tools such as Puppet or Vagrant yourself.

Note that you can also automate the 'install and launch' phase of provisioning Stash – see Running the Stash installer for information about how to run the installer in console and unattended modes.

1. Get a license for Stash

You can get a new Stash license by doing one of:

- Logging in to your My.Atlassian.com account.
- Contacting Atlassian, if you need a Stash Data Center license.

If you already have a licensed instance of Stash, you can find the license in the Stash admin area.

2. Set the configuration properties

After installing Stash, but before you start Stash for the first time, edit the `stash-config.properties` file to add the properties in the table below. Use the standard format for Java properties files.

Note that the `stash-config.properties` file is created automatically in the `shared` folder of your Stash home directory when you perform a database migration. Create the file yourself if it does not yet exist. See Stash config properties for information about the properties file.

Add these properties to the `stash-config.properties` file:

Property	Description
`setup.displayName=displayName`	The display name for the Stash application.
`setup.baseUrl= https://stash.yourcompany.com`	The base URL.
`setup.license=AAAB...`	The Stash license. Use the \ character at the end of each line if you wish to break the license string over multiple lines.
`setup.sysadmin.username=username`	Credentials for the system admin account.
`setup.sysadmin.password=password`	
`setup.sysadmin.displayName=John Doe`	The display name for the system admin account. An empty property is ignored.
`setup.sysadmin.emailAddress=sysadmin@yourcompany.com`	The email address for the system admin account.
`jdbc.driver=org.postgresql.Driver`	JDBC connection parameters. Use the appropriate values for your own JDBC connection.
`jdbc.url=jdbc:postgresql://localhost:5432/stash`	
`jdbc.user=stash`	
`jdbc.password=stash`	

You should specify the JDBC properties so that Stash can connect to the external database.

If any of the following required properties are not provided in the properties file, when you start Stash the Setup Wizard will launch at the appropriate screen so that you can enter values for those properties.

3. Start Stash

Start Stash as usual. See Starting and stopping Stash.

Stash reads the `stash-config.properties` file and applies the setup properties automatically.

When you now visit Stash in the browser, you see the welcome page.

Troubleshooting

The Setup Wizard launches in the browser

The Setup Wizard will run if there are missing configuration properties, such as the license string, in the `stash-config.properties` file. Check the properties file and compare with the table in Step 2 above. Alternatively, the set up can be completed using the web UI.

Write access for the `config.properties` file

Once the automated setup process completes, the relevant properties in the `stash-config.properties` file are commented out. This requires that the system user has write permission on the properties file.

Stash fails to start with a 'Could not acquire change log lock.' error

If Stash is forced to quit while modifying the config.properties file, you may not be able to restart Stash, and `atlassian-stash.log` contains the above error.

See this KB article for information about how to resolve this: Stash Does Not Start - Could not acquire change log lock

Starting and stopping Stash

There are a few ways that you can start and stop Stash:

- At install time
- When Stash runs as a service
- Manually

At install time

The Stash installer automatically starts Stash.

On Windows and Linux systems you can choose to have Stash installed as a service.

When Stash runs as a service

If Stash is installed as a service on Windows or Linux systems, it will be started automatically when the system boots.

Windows

Start and stop the Stash service from the services console, on Windows.

Linux

Manage the Stash service using the following commands:

```
# service atlstash status
# service atlstash stop
# service atlstash start
```

Mac OS X

On Mac OS X, you will need to restart Stash manually, as described below.

Manually

You can start and stop Stash manually as follows:

Windows

Start and stop Stash using the **Stash** items in the Windows Start menu. Use the URL item there to visit Stash in your default browser.

Alternatively, start Stash from a command prompt, by changing directory to the `<Stash installation directory>` and running the following command:

```
bin\start-stash.bat
```

Stop Stash manually by changing directory to the `<Stash installation directory>` and running the following command:

```
bin\stop-stash.bat
```

Linux

Start and stop Stash manually using the scripts provided.

Start Stash by changing directory in a terminal to the `<Stash installation directory>` and running:

```
bin/start-stash.sh
```

Stop Stash by changing directory in a terminal to the `<Stash installation directory>` and running:

```
bin/stop-stash.sh
```

Mac

Start and stop Stash manually using the app icons (shown above) in the `<Stash installation directory>`. These simply link to the `start-stash.sh` and `stop-stash.sh` scripts in `<Stash installation directory>/bin`.

Use the URL icon to visit Stash in your default browser.

Install Stash from an archive file

To install Stash yourself...	Use the installer instead...	For production installs...
... from an archive file, on Linux, Mac OS X or Windows, read on!	... for a quick and trouble-free install. See Getting started.	... we highly recommend that you first read Using Stash in the enterprise.

This page describes how to manually install Stash from an archive file. However, we strongly recommend that you use the Stash installer instead, for a quick and trouble-free install experience.

You may be interested in these alternative provisioning approaches:

- Docker container image for Stash (currently only supported for evaluations)
- Chef recipe for Stash
- Puppet/Vagrant How-To

1. Check supported platforms

Check the Supported platforms page for details of the application servers, databases, operating systems, web browsers and Java and Git versions that we have tested Stash with and recommend.

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

> Cygwin Git is *not supported*. No internal testing is done on that platform, and many aspects of Stash's functionality (pull requests and forks among them) have known issues. When running Stash on Windows, *always* use msysGit.

2. Check your version of Java

In a terminal or command prompt, run this:

```
java -version
```

The version of Java should be **1.7.x** or **1.8.x**. You'll need a 64-bit version of Java if you have a 64-bit operating system.

On Linux, if you don't see a supported version, then get Java...

Install Java

Download Java Server JRE from Oracle's website, and install it.

Now try running '`java -version`' again to check the installation. The version of Java should be **1.7.x** or **1.8.x.**

Check that the system can find Java

In a terminal, run this:

```
echo $JAVA_HOME
```

You should see a path like `/usr/jdk/jdk1.7.0`.

If you don't see a path, then set JAVA_HOME

Do one of the following:

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

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

  where `path/to/JAVA_HOME` may be like: `/usr/jdk/jdk1.7.0`
- If `JAVA_HOME` needs to be changed, open the `/etc/environment` file in a text editor and modify the value for `JAVA_HOME` to:

  `JAVA_HOME="path/to/JAVA_HOME"`

  It should look like: `/usr/jdk/jdk1.7.0`

On Mac OS X, if you don't see a supported version, then get Java...

Install Java

Download Java Server JRE from Oracle's website, and install it.

Now try running '`java -version`' again to check the installation. The version of Java should be **1.7.x** or **1.8.x**.

Check that the system can find Java

In a terminal, run this:

```
echo $JAVA_HOME
```

You should see a path like `/System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/`.

If you don't see a path, then set JAVA_HOME

Open your `~/.profile` file in a text editor and insert:

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

where `path/to/JAVA_HOME` may be like: `/System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/`

Refresh your ~/.profile in the terminal and confirm that `JAVA_HOME` is set:

```
source ~/.profile
$JAVA_HOME/bin/java -version
```

You should see a version of Java that is **1.7.x** or **1.8.x**, like this:

```
java version "1.7.0_1"
```

On Windows, if you don't see a supported version, then get Java...

Install Java

Download Java Server JRE from Oracle's website, and install it.

Now try running '`java -version`' again to check the installation. The version of Java should be **1.7.x** or **1.8.x.**

Check that the system can find Java

Stash uses the JAVA_HOME environment variable to find Java. To check that, in a command prompt, run:

```
echo %JAVA_HOME%
```

You should see a path to the root directory of the Java installation. When running Stash on Windows, unlike Linux or Unix, `JAVA_HOME` paths with spaces are just fine.

If you don't see a path, then set JAVA_HOME

Windows 7 ...

Stage 1. Locate the JRE installation directory

If you already know the installation path for the Java Runtime Environment, go to *Stage 2* b elow. Otherwise, find the installation path by following these instructions:

1. If you didn't change the installation path for the Java Runtime Environment during installation, it will be in a directory under `C:\Program Files\Java`. Using Explorer, open the directory `C:\Program Files\Java`.
2. Inside that path will be one or more subdirectories such as `C:\Program Files\Java\jre7`.

Stage 2. Set the JAVA_HOME variable

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "JAVA_HOME" as the **Variable name**, and the absolute path to where you installed Java as the **Variable value**. Don't use a trailing backslash, and don't wrap the value in quotes.

Now, in a *new command prompt*, try running '`%JAVA_HOME%\bin\java -version`'. You should see the same version of Java as you saw in 2. above.

Windows Server 2003 R2 ...

Stage 1. Locate the JRE installation directory

If you already know the installation path for the Java Runtime Environment, go to *Stage 2* b elow. Otherwise, find the installation path by following these instructions:

1. If you didn't change the installation path for the Java Runtime Environment during installation, it will be in a directory under `C:\Program Files\Java`. Using Explorer, open the directory `C:\Program Files\Java`.
2. Inside that path will be one or more subdirectories such as `C:\Program Files\Java\jre7`.

Stage 2. Set the JAVA_HOME variable

Once you have identified the JRE installation path:

1. Right-click the **My Computer** icon on your desktop and select **Properties**.
2. Click the **Advanced** tab.
3. Click the **Environment Variables** button.
4. Under **System Variables**, click **New**.
5. Enter the **variable name** as JAVA_HOME.
6. Enter the **variable value** as the installation path for the Java Development Kit. Don't use a trailing backslash, and don't wrap the value in quotes.
   - If your Java installation directory has a space in its path name, you should use the shortened path name (e.g. `C:\Progra~1\Java\jre7`) in the environment variable instead.

   Note for Windows users on 64-bit systems

   Progra~1 = 'Program Files'
   Progra~2 = 'Program Files(x86)'
7. Click **OK**.
8. Click **Apply Changes**.
9. Close any command window which was open before you made these changes, and open a new command window. There is no way to reload environment variables from an active command prompt. If the changes do not take effect even after reopening the command window, restart Windows.

Now, in a *new command prompt*, try running '`%JAVA_HOME%\bin\java -version`'. You should see the same version of Java as you saw in 2. above.

3. Check your versions of Git and Perl

In a terminal or command prompt, run:

```
git --version
perl --version
```

The version of Git should be **1.7.6** or higher. The version of Perl should be **5.8.8** or higher.

If you don't see supported versions of Git and Perl, either install or upgrade them – see Installing and upgrading Git.

4. Now it's time to get Stash

Download Stash from the Atlassian download site. Looking for

the Stash WAR file?

Extract the downloaded file to an install location (without spaces in the path).

The path to the extracted directory is referred to as the `<Stash installation directory>` in these instructions.

> Never unzip the Stash archive file over the top of an existing Stash installation – each version of Stash includes versioned jar files, such as `stash-model-2.4.1.jar`. If you copy these, you end up with multiple versions of Stash's jar files in the classpath, which leads to runtime corruption.

Note that you should use the same user account to both extract Stash and to run Stash (in Step 6.) to avoid possible permission issues at startup. For production installations, we recommend that you create a new dedicated user that will run Stash on your system. See Running Stash with a dedicated user.

5. Tell Stash where to store your data

The Stash home directory is where your Stash data is stored.

If you are upgrading Stash, simply update the value of `STASH_HOME` in the `<Stash installation directory>/bin/setenv` file so the *new* Stash installation points to your *existing* Stash home directory (if you use a `STASH_HOME` environment variable to specify the home directory location, no change is required).

Otherwise, for a new install, create your Stash home directory (without spaces in the name), and then tell Stash where you created it by editing the `<Stash installation directory>/bin/setenv` file – uncomment the `STASH_HOME` line and add the absolute path to your home directory. Here's an example of what that could look like when you're done:

```
#
if [ "x${STASH_HOME}" = "x" ]; then
    export STASH_HOME="/home/username/stash-home"
fi
```

⚠ **You *should not* locate your Stash home directory inside the `<Stash installation directory>`** — they should be entirely separate locations. If you do put the home directory in the `<Stash installation directory>` it may be overwritten, and lost, when Stash gets upgraded. And by the way, you'll need separate Stash home directories if you want to run multiple instances of Stash.

Click here for Windows notes...

Tell Stash where you created it by setting a `STASH_HOME` environment variable, for Windows 7, as follows:

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "`STASH_HOME`" as the **Variable name**, and the absolute path to your Stash home directory as the **Variable value**. Don't use a trailing backslash.

There are a few things to know about setting up the Stash home directory on Windows that will make life easier:

- Keep the path length to the Stash home directory as short as possible. See Stash always shows incorrect Merge Conflict in PRs for an explanation.
- Don't use spaces in the path to the Stash home directory.

6. Move server.xml to your Stash home `shared` directory

Starting with Stash 3.8, the server.xml file (which, among other things, configures which port Stash serves web requests on) should be located in the `shared` directory of your Stash home. When you unzip/untar the Stash installation archive server.xml will be located in `<Stash installation directory>/conf`. You should move this to the `shared` directory of your Stash home ensuring it is readable by the user account which will run Stash.

If you leave it in `<Stash installation directory>/conf` Stash will still read it but when you upgrade

to new versions you will have to remember to copy it across to the new Stash installation directory. Much easier is to keep it in the `shared` directory of your Stash home as it will not have to be moved for each upgrade.

7. Start Stash!

There are a couple of ways in which you can start Stash – see Starting and stopping Stash.

Now, in your browser, go to http://localhost:7990 and run through the Setup Wizard. In the Setup Wizard:

- Select **Internal** at the 'Database' step, if you are evaluating Stash. Stash will happily use its internal database, and you can easily migrate to external database later. See Connecting Stash to an external database.
- Enter your Stash license key.
- Set the base URL for Stash.
- Set up an administrator account.
- You can set up JIRA integration, but you can do this later if you wish. See Configuring JIRA integration in the Setup Wizard .

8. Set up your mail server

Configure your email server so users can receive a link from Stash that lets them generate their own passwords. See Setting up your mail server.

9. Add users and repositories

Now is the time to set up your users in Stash, and to tell Stash about any existing repositories you have. Please the following pages for the details:

- Getting started with Git and Stash
- Importing code from an existing project

Additional steps for production environments

For production or enterprise environments we recommend that you configure the additional aspects described on Using Stash in the enterprise. The aspects described there are not necessary when you are installing for evaluation purposes only.

If you wish to install Stash as a service on Linux or Windows, see either of:

- Running Stash as a Linux service
- Running Stash as a Windows service

Stopping Stash

See Starting and stopping Stash.

Uninstalling Stash

To uninstall Stash, stop Stash as described above and then delete the `<Stash installation directory>` and Stash home directory.

Running Stash as a Linux service

- The Stash installer for Linux installs Stash as a service – see Getting started. The information on this page only applies if you are manually installing or upgrading Stash.
- System administration tasks are not supported by Atlassian. These instructions are only provided

as a guide and may not be up to date with the latest version of your operating system.

For production use on a Linux server, Stash should be configured to run as a Linux service, that is, as a daemon process**.** This has the following advantages:

- Stash can be automatically restarted when the operating system restarts.
- Stash can be automatically restarted if it stops for some reason.
- Stash is less likely to be accidentally shut down, as can happen if the terminal Stash was manually started in is closed.
- Logs from the Stash JVM can be properly managed by the service.

This page describes the following approaches to running Stash as a service on Linux:

- Use the Java Service Wrapper, which allows a Java application to be run as a UNIX daemon.
- Use an `init.d` script to start Stash at boot time - this doesn't restart Stash if it stops for some reason.
- Use a systemd unit file to start Stash at boot time - this doesn't restart Stash if it stops for some reason.

Note that Stash assumes that the external database is available when it starts; these approaches do not support service dependencies, and the startup scripts will not wait for the external database to become available.

On this page:

- Using the Java Service Wrapper
- Using an init.d script
  - Running on system boot
- Using a systemd unit file

Related pages:

- Install Stash from an archive file

Using the Java Service Wrapper

Stash can be run as a service on Linux using the Java Service Wrapper. The Service Wrapper is known to work with Debian, Ubuntu, and Red Hat.

The Service Wrapper provides the following benefits:

- Allows Stash, which is a Java application, to be run as a service.
- No need for a user to be logged on to the system at all times, or for a command prompt to be open and running on the desktop to be able to run Stash.
- The ability to run Stash in the background as a service, for improved convenience, system performance and security.
- Stash is launched automatically on system startup and does not require that a user be logged in.
- Users are not able to stop, start, or otherwise tamper with Stash unless they are an administrator.
- Can provide advanced failover, error recovery, and analysis features to make sure that Stash has the maximum possible uptime.

Please see http://wrapper.tanukisoftware.com/doc/english/launch-nix.html for wrapper installation and configuration instructions.

The service wrapper supports the standard commands for SysV init scripts, so it should work if you just create a symlink to it from `/etc/init.d`.

Using an init.d script

The usual way on Linux to ensure that a process restarts at system restart is to use an init.d script. This approach does not restart Stash if it stops by itself.

1. Stop Stash.
2. Create a stash user, set the permissions to that user, create a home directory for Stash and create a symlink to make upgrades easier:

```
$> curl -OL
http://downloads.atlassian.com/software/stash/downloads/atlassian-stash-X.Y.Z.
tar.gz
$> tar xz -C /opt -f atlassian-stash-X.Y.Z.tar.gz
$> ln -s /opt/atlassian-stash-X.Y.Z /opt/atlassian-stash-latest

# Create a home directory
$> mkdir /opt/stash-home

# ! Update permissions and ownership accordingly
```

(Be sure to replace X.Y.Z in the above commands with the version number of Stash.)

3. Create the startup script in `/etc/init.d/stash` with the following contents (Ensure the script is executable by running `chmod 755 stash`):

```
#! /bin/sh

### BEGIN INIT INFO
# Provides:          stash
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Initscript for Atlassian Stash
# Description:  Automatically start Atlassian Stash when the system starts up.
#               Provide commands for manually starting and stopping Stash.
### END INIT INFO

# Adapt the following lines to your configuration
# RUNUSER: The user to run Stash as.
RUNUSER=vagrant

# STASH_INSTALLDIR: The path to the Stash installation directory
STASH_INSTALLDIR="/opt/atlassian-stash-X.Y.Z"

# STASH_HOME: Path to the Stash home directory
STASH_HOME="/opt/stash-home"

#
================================================================================
====
#
================================================================================
====
#
================================================================================
====

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="Atlassian Stash"
NAME=stash
PIDFILE=$STASH_INSTALLDIR/work/catalina.pid
SCRIPTNAME=/etc/init.d/$NAME

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
```

```
. /lib/lsb/init-functions


run_with_home() {
    if [ "$RUNUSER" != "$USER" ]; then
        su - "$RUNUSER" -c "export
STASH_HOME=${STASH_HOME};${STASH_INSTALLDIR}/bin/$1"
    else
        export STASH_HOME=${STASH_HOME};${STASH_INSTALLDIR}/bin/$1
    fi
}

#
# Function that starts the daemon/service
#
do_start()
{
    run_with_home start-stash.sh
}

#
# Function that stops the daemon/service
#
do_stop()
{
    if [ -e $PIDFILE ]; then
      run_with_home stop-stash.sh
    else
      log_failure_msg "$NAME is not running."
    fi
}


case "$1" in
  start)
    [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
    do_start
    case "$?" in
        0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
        2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
    esac
    ;;
  stop)
    [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
    do_stop
    case "$?" in
        0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
        2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
    esac
    ;;
  status)
       if [ ! -e $PIDFILE ]; then
         log_failure_msg "$NAME is not running."
         return 1
       fi
       status_of_proc -p $PIDFILE "" $NAME && exit 0 || exit $?
       ;;
  restart|force-reload)
    #
    # If the "reload" option is implemented then remove the
    # 'force-reload' alias
    #
    log_daemon_msg "Restarting $DESC" "$NAME"
```

```
    do_stop
    case "$?" in
      0|1)
        do_start
        case "$?" in
            0) log_end_msg 0 ;;
            1) log_end_msg 1 ;; # Old process is still running
            *) log_end_msg 1 ;; # Failed to start
        esac
        ;;
      *)
        # Failed to stop
        log_end_msg 1
        ;;
    esac
    ;;
  *)
    echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-reload}" >&2
```

```
    exit 3
    ;;
esac
```

Running on system boot

1. To start on system boot, add the script to the start up process.
   For Ubuntu (and other Debian derivatives) use:

   ```
   update-rc.d stash defaults
   ```

   For RHEL (and derivates) use:

   ```
   chkconfig --add stash --level 0356
   ```

   Note: You may have to install the `redhat-lsb` package on RHEL (or derivatives) to provide the LSB functions used in the script.
2. Verify that the Stash service comes back up after restarting the machine.

Using a systemd unit file

Thanks to Patrick Nelson for calling out this approach, which he set up for a Fedora system. It also works on other distributions that use systemd as the init system. This approach does not restart Stash if it stops by itself.

1. Create a `stash.service` file in your `/etc/systemd/system/` directory with the following lines:

   ```
   [Unit]
   Description=Atlassian Stash Service
   After=syslog.target network.target

   [Service]
   Type=forking
   User=atlstash
   ExecStart=/opt/atlassian-stash-X.Y.Z/bin/start-stash.sh
   ExecStop=/opt/atlassian-stash-X.Y.Z/bin/stop-stash.sh

   [Install]
   WantedBy=multi-user.target
   ```

   The value for `User` should be adjusted to match the user that Stash runs as. `ExecStart` and `ExecStop` should be adjusted to match the path to your `<Stash installation directory>`.
2. Enable the service to start at boot time by running the following in a terminal:

   ```
   systemctl enable stash.service
   ```

3. Stop Stash, then restart the system, to check that Stash starts as expected.
4. Use the following commands to manage the service:

   Disable the service:

```
systemctl disable stash.service
```

Check that the service is set to start at boot time:

```
if [ -f /etc/systemd/system/*.wants/stash.service ]; then echo "On"; else echo
"Off"; fi
```

Manually start and stop the service:

```
systemctl start stash
systemctl stop stash
```

Check the status of Stash:

```
systemctl status stash
```

Running Stash as a Windows service

This page only applies...

... if you are manually installing or upgrading Stash from an archive file.

If you're using the installer...

... you should read the Stash Getting started page instead.

Related pages...

- Running Stash as a Linux service
- Using Stash in the enterprise.

We recommend that you use the Stash installer to install Stash as a service on Windows. It installs Stash as a service and creates items in the Windows 'Start' menu for starting and stopping Stash – see Getting started .

The information on this page only applies if you are manually installing or upgrading Stash from an archive file. See Install Stash from an archive file.

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.

System administration tasks are not supported by Atlassian. These instructions are only provided as a guide.

Prerequisites

- If you are using a 64-bit version of Windows, first ensure that Stash uses a 64-bit JVM (check by running `java -version` in a Command Prompt, and ensure that the `JAVA_HOME` system environment variable points to the 64-bit JVM), and then replace the 32-bit Tomcat binaries with their

64-bit counterparts in the <Stash installation directory>/bin directory:

```
cd <STASH-INST/bin>
rename tomcat8.exe tomcat8.exe.x86
rename tcnative-1.dll tcnative-1.dll.x86
rename tomcat8.exe.x64 tomcat8.exe
rename tcnative-1.dll.x64 tcnative-1.dll
```

- On any Windows operating system with User Account Control (UAC) such as Windows Vista or Windows 7, simply logging in to Windows with an Administrator account will not be sufficient to execute the script in the procedure below. You must either disable UAC or run 'cmd.exe' as an administrator (e.g. by right-clicking on 'cmd.exe' 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 Java installation directory of C:\Program Files\Java won't work.*
- Stash should be run from a local dedicated user account that does not have admin privileges and that has read, write and execute access to the Stash home directory and the <Stash installation directory>. See Git Push Operations Extremely Slow on Windows.
- When you run Stash as a Windows service, all settings in setenv.bat are ignored. Ensure that you have set STASH_HOME as a *system* environment variable, before running the service.bat script.
- If you upgraded Stash from version 1.x to 2.x and Stash stopped running as a service you will need to reinstall the service according to instructions in the Stash upgrade guide.

Set up Stash as a Windows service

The information in this section only applies if you are manually installing Stash as a Windows service. Alternatively, you can use the Stash installer for Windows to install Stash as a service – see Running the Stash installer.

To run Stash as a Windows service:

1. Stop Stash.
2. Create a *system* environment variable with STASH_HOME as the **Variable name** and the absolute path to your Stash home directory as the **Variable value**. Don't use a trailing backslash. Note that the Stash home directory *should not* be located inside the <Stash installation directory>. You must do this step *before* running the service.bat script in Step 5 below.
3. Open a Command Prompt (as an Administrator – see the 'Prerequisites' section above).
4. Change directory 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:\Progra~1\..).
5. Run the following commands:

```
> service.bat install AtlassianStash
> tomcat8 //US//AtlassianStash --Startup auto
```

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

```
> service.bat install MyName
> tomcat8 //US//MyName --Startup auto
```

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

6. Run the following command to increase the amount of memory that Stash can use (the default is 768 Mb):

```
> tomcat8 //US//service_name --JvmMx 1024
```

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

Here is an example:

```
C:\Program Files (x86)\atlassian-stash-2.0.0\bin>service.bat install
Installing the service 'AtlassianStash' ...
Using CATALINA_HOME:    "C:\Program Files (x86)\atlassian-stash-2.0.0"
Using CATALINA_BASE:    "C:\Program Files (x86)\atlassian-stash-2.0.0"
Using JAVA_HOME:        "C:\Java\jre6"
Using JVM:              "auto"
The service 'AtlassianStash' has been installed.
C:\Program Files (x86)\atlassian-stash-2.0.0\bin>tomcat8.exe //US//AtlassianStash
--Startup auto
C:\Program Files (x86)\atlassian-stash-2.0.0\bin>tomcat8.exe //US//AtlassianStash
--JvmMx 1024

C:\Program Files (x86)\atlassian-stash-2.0.0\bin>net start AtlassianStash
The Atlassian Stash service is starting.
The Atlassian Stash service was started successfully.
```

Troubleshooting

- If your service fails to start with "code 4", make sure you ran `service.bat install` in a Command Prompt running as an Administrator.

Stash config properties

This page describes the Stash system properties that can be used to control aspects of the behaviour in Stash. Create the `stash-config.properties` file, in the `shared` folder of your Stash home directory, and add the system properties you need, use the standard format for Java properties files.

Note that the `stash-config.properties` file is created automatically when you perform a database migration.

Stash must be restarted for changes to become effective.

Default values for system properties, where applicable, are specified in the tables below.

On this page:

- Audit
- Authentication
- Avatars
- Backup
- Changesets
- Changeset indexing
- Commit graph cache
- Database
- Database pool
- Display
- Downloads
- Events
- Executor
- Features
- Hibernate
- JIRA
- JMX

- Liquibase
- Logging
- Notifications
- Paging
- Password reset
- Process execution
- Pull requests
- Readme parsing
- Ref metadata
- Resource throttling
- SCM – Cache
- SCM – Git
- Server busy banners
- Setup automation
- SMTP
- SSH command execution
- SSH security
- Syntax highlighting
- Webhooks

Audit

Property	Description
`audit.highest.priority.to.log=HIGH`	Defines the lowest priority audit events that will be logged. Accepted values are: HIGH, MEDIUM, LOW and NONE. Setting the value to HIGH will result in only HIGH level events being logged. NONE will cause no events to be logged. MEDIUM will only allow events with a priority of MEDIUM and HIGH to be logged. Refer to the levels for the various events. This does not affect events displayed in the Audit log screens for projects and repositories.
`audit.details.max.length=1024`	Defines the number of characters that can be can stored as details for a single audit entry.
`plugin.stash-audit.max.entity.rows=500`	The maximum number of entries a project or repository can have in the audit tables. This does not affect the data stored in the logs.
`plugin.stash-audit.cleanup.batch.size=1000`	When trimming the audit entries table this is the maximum number of rows that will be trimmed in one transaction. Reduce this size if you are having issues with long running transactions. This does not affect the data stored in the logs.
`plugin.stash-audit.cleanup.run.interval=24`	How often the audit tables will be checked to see if they need to be trimmed (in hours). This does not affect the data stored in the logs.

Authentication

See also Connecting Stash to Crowd.

Property	Descripti
`plugin.auth-crowd.sso.enabled=false`	Whether S support sh be enable not. Regardles this settin SSO authentica will only b activated v a Crowd directory i configured Stash that configured SSO.
`plugin.auth-crowd.sso.session.lastvalidation=atl.crowd.sso.lastvalidation`	The sessi key to use when stori Date value the user's authentica
`plugin.auth-crowd.sso.session.tokenkey=atl.crowd.sso.tokenkey`	The sessi key to use when stori String valu the user's authentica token.
`plugin.auth-crowd.sso.session.validationinterval=3`	The numb minutes to cache authentica validation the sessio this value set to 0, th SSO sess will be validated v the Crowd server for every HTT request.
`plugin.auth-crowd.sso.http.max.connections=20`	The maxir number of HTTP connectio the conne pool for communic with the C server.

`plugin.auth-crowd.sso.http.proxy.host`	The name the proxy server use transport SOAP traf the Crowd server.
`plugin.auth-crowd.sso.http.proxy.port`	The connectio port of the proxy serv (must be specified i proxy host specified).
`plugin.auth-crowd.sso.http.proxy.username`	The usern used to authentica with the pr server (if t proxy serv requires authentica
`plugin.auth-crowd.sso.http.proxy.password`	The passv used to authentica with the pr server (if t proxy serv requires authentica
`plugin.auth-crowd.sso.http.timeout=5000`	The HTTP connectio timeout in millisecon used for communic with the C server. A v of zero indicates t there is nc connectio timeout.
`plugin.auth-crowd.sso.socket.timeout=20000`	The socke timeout in millisecon You may v to overrid default val the latenc the Crowd server is h

Avatars

Property	Description
`avatar.gravatar.default=mm`	The fallback URL for Gravatar avatars when a user does not have an acceptable avatar configured. This ma be a URL resource, or a Gravatar provided default set. This configuration setting is DEPRECATED. It will be removed in Stash 3.0. Use `avatar.url.defaul` nstead.
`avatar.max.dimension=1024`	Controls the max height *and* width for a avatar image. Even if the avatar is with the acceptable file size, if its dimensior exceed this value for height *or* width, it will be rejected. When an avatar is loaded by the serve for processing, images with large dimensions may expand from as small a few kilobytes on disk to consume a substantially larger amount of memory, depending on how well the image data was compressed. Increasing this limit c *substantially* increase the amount of he used while processing avatars and ma result in OutOfMemoryErrors. Value is in PIXELS.
`avatar.max.size=1048576`	Controls how large an avatar is allowed to be. Avatars larger than this are rejected and cannot be uploaded to the server, to prevent excessive disk usag Value is in BYTES.
`avatar.temporary.cleanup.interval=1800000`	Controls how frequently temporary avatars are cleaned up. Any temporary avatars that have been uploaded are checked against their configured max age and removed from the file system they are "too old". Value is in MILLISECONDS.
`avatar.temporary.max.age=30`	Controls how long a temporary avatar that has been uploaded is retained bef it is automatically deleted. Value is in MINUTES.

`avatar.url.default=${avatar.gravatar.default}`	Defines the fallback URL to be formatte into the `avatar.url.format.http` `avatar.url.format.https` URL format for use when a user does not ha an acceptable avatar configured. This value may be a URL or, if using Gravatar, it may be the identifier for on of Gravatar's default avatars. The default here falls back on the now-deprecated `avatar.gravatar.` `fault` setting, which should ensure th value, if set, continues to work until it is removed in Stash 3.0. At that time, this default will become "mm".
`avatar.url.format.http=http://www.gravatar.com/` `avatar/%1$s.jpg?s=%2$d&d=%3$s`	Defines the default URL format for retrieving user avatars over HTTP. This default uses any G-rated avatar provided by the Gravatar service [http:/ ww.gravatar.com] The following format parameters are available: %1$s – the user's e-mail address, MD5 hashed, or "00000000000000000000000000000000 if the user has no e-mail. %2$d – the requested avatar size. %3$s – the fallback URL, URL-encode which may be defined using "avatar.url.default". %4$s – the user's e-mail address, not hashed, or an empty string if the user h no e-mail.
`avatar.url.format.https=https://secure.gravatar.com/` `avatar/%1$s.jpg?s=%2$d&d=%3$s`	Defines the default URL format for retrieving user avatars over HTTPS. Th default uses any G-rated avatar provided by the Gravatar service [http:/ ww.gravatar.com] The following format parameters are available: %1$s – the user's e-mail address, MD5 hashed, or "00000000000000000000000000000000 if the user has no e-mail. %2$d – the requested avatar size. %3$s – the fallback URL, URL-encode which may be defined using "avatar.url.default". %4$s – the user's e-mail address, not hashed, or an empty string if the user h no e-mail.

Backup

Property	Description
`backup.drain.database.timeout=60`	Defines the number of seconds Stash will wait for connections to the database to drain and latch in preparation for a backup. Value is in SECONDS.

Changesets

Property	Description
`changeset.diff.context=10`	Defines the number of context lines to include around diff segments in changeset diffs.

Changeset indexing

These properties control how changesets are indexed when new commits are pushed to Stash.

Property	Description
indexing.max.threads=2	Controls the maximum number of threads which are used to perform indexing. The resource limits configured below are not applied to these threads, so using a high number may negatively impact server performance.
indexing.job.batch.size=250	Defines the number of changesets which will be indexed in a single database transaction.
indexing.job.queue.size=150	Defines the maximum number of pending indexing requests. When this limit is reached, attempts to queue another indexing operation will be rejected.
indexing.process.timeout.execution=3600	Controls how long indexing processes are allowed to execute before they are interrupted, even if they are producing output or consuming input. Value is in SECONDS.

Commit graph cache

Property	Description
`commit.graph.cache.min.free.space=1073741824`	Controls how much space needs to be available on disk (specifically under `<Stash home directory>/caches`) for caching to be enabled. This setting ensures that the cache plugin does not fill up the disk. Value is in BYTES.

`commit.graph.cache.max.threads=2`	Defines the number of threads that will be used to create commit graph cache entries.
`commit.graph.cache.max.job.queue=1000`	Defines the maximum number of pending cache creation jobs.

Database

Database properties allow very specific configuration for your database connection parameters, which are set by Stash during database setup and migration, and allow you to configure a database of your own. We don't expect that you will edit these, except in collaboration with Atlassian Support.

If none of the properties below are specified in `stash-config.properties`, then the internal HSQL database will be used.

If the `jdbc.driver`, `jdbc.url`, `jdbc.password` and `jdbc.user` properties are specified in `stash-config.properties` when the Setup Wizard runs after installing Stash, then those values will be used, and the Setup Wizard will not display the database configuration screen.

Any other driver must be placed in `WEB-INF/lib` in order to use the associated database.

⚠ **Warning**: `jdbc.driver` and `jdbc.url` are available to plugins via the ApplicationPropertiesService. Some JDBC drivers allow the username and password to be defined in the URL. Because that property is available throughout the system (and will be included in STP support requests), that approach should not be used. The `jdbc.username` and `jdbc.password` properties should be used for these values instead.

Property	Description
`jdbc.driver=org.hsqldb.jdbcDriver`	The JDBC driver class that shoul to the database. The internal Stash database is H `jdbcDriver`. It stores its data in Stash bundles these other JDBC • `org.postgresql.Driver` • `com.microsoft.sqlserve` (more info) • `oracle.jdbc.driver.Ora` The JDBC drivers for MySQL are to licensing restrictions) so you w install the driver yourself. See Co instructions.
`jdbc.url=jdbc:hsqldb:${stash.home}/data/db;shutdown=true`	This is the JDBC url that Stash w database. This should include the `tgresql:`), the hostname, port a connect to. This string may vary d you are connecting to. Please see databases from your database pr
`jdbc.user=stash`	This is the user that Stash will co The user will need to be able to c indexes, as well as read and write database schema defined in `jdb`

`jdbc.password=stash`	The password that the user defin connect with.
`jdbc.ignoreunsupported=false`	Allows using a given database, e UNSUPPORTED. This is not inte documented, nor to be used gene mechanism to override the suppo event that it incorrectly blocks acc

Database pool

These properties control the database pool. The pool implementation used is BoneCP. Documentation for these settings can be found at: http://jolbox.com/configuration.html

To get a feel for how these settings really work in practice, the most relevant classes in BoneCP are:

- com.jolbox.bonecp.BoneCP - creates the partitions, opens initial connections, starts threads.
- com.jolbox.bonecp.BoneCPDataSource - manages the pool, bridge to the DataSource interface.
- com.jolbox.bonecp.PoolWatchThread - handles the connection threshold.

Property	Description
`db.pool.acquireIncrement=2`	Defines the number of connections to open in a batch when open connections are almost exhausted for a given partition.
`db.pool.cache.statements=100`	Defines the number of statements to cache. If you configure your JDBC driver to use loadbalancing or failover with the Stash server, you may need to set `db.pool.cache.statements=0`. Please note that Stash does not yet (as of version 2.8) support DB redundancy, and that this configuration change is not supported by Atlassian.
`db.pool.connection.timeout=15`	Defines the amount of time the system will wait when attempting to open a new connection before throwing an exception. The system may hang, during startup, for the configured number of seconds if the database is unavailable. As a result, the timeout configured here should **not** be generous. Value is in SECONDS.
`db.pool.idle.maxAge=30`	Defines the maximum period of time a connection may be idle before it is closed. Generous values should be used here to prevent creating and destroying many short-lived database connections (which defeats the purpose of pooling). Value is in MINUTES.
`db.pool.idle.testInterval=10`	Defines the amount of time a connection may be idle before a test query is executed by the pool. This helps prevent connections from being closed by the database server due to inactivity. Value is in MINUTES.

`db.pool.partition.connection.maximum=20`	Defines the maximum number of connections that may be open in a given partition.
`db.pool.partition.connection.minimum=4`	Defines the minimum number of connections open for a given partition. Each partition will open this many connections on startup. That means `db.pool.partition.connection.minimum` x `db.pool.partition.count` = initial connection count.
`db.pool.partition.connection.threshold=10`	Defines the threshold *as a percentage of the maximum connections* that each partition will attempt to keep available at all times. If the number of available connections drops **to or below** the threshold, `acquireIncrement` connections will be opened until the partition is above it again. ⚠️ **Warning**: Be careful to take this number into account when setting the minimum and maximum counts. For example, if the maximum is 30 and the minimum is 5 and the threshold is 20 (20%), 5 is not 20% of 30, so immediately after it is created the partition will open additional connections to get above the threshold. Effectively, that would mean that the "minimum" per partition is 10 (5 + 5 `acquireIncrement`), or 40 connections at all times.
`db.pool.partition.count=4`	Defines the number of different connection partitions to use. This value is used to decrease lock contention, because each partition locks individually. The recommended setting is 3 or 4, but in servers with heavy load and many short-lived requests, performance may be improved by using a higher value.
`db.pool.threads=4`	Defines the number of helper threads which will be used by the pool to cleanup and release connections back into the pool. Setting a value of 0 disables this feature, which means the executing thread will perform cleanup and release itself. A non-zero value results in a pool of helpers which process connections out of a holding queue. When a thread "closes" a connection, that thread is allowed to continue executing and the connection is placed in the queue. One of the helper threads then performs final cleanup to prepare the connection to be returned to the pool.

Display

Property	Description
`display.max.source.lines=20000`	Controls how many lines of a source file will be retrieved before a warning banner is shown that encourages the user is to download the raw file for further inspection. This property relates to `page.max.source.lines` (see Paging below) in that up to (`display.max.source.lines` / `page.max.source.lines`) requests will be made to view the page.

Downloads

Property	Description
`http.download.raw.policy=Smart`	Controls the download policy for raw content. Possible values are: `Insecure` – allows all file types to be viewed in the browser. `Secure` – requires all file types to be downloaded rather than viewed in the browser. `Smart` – forces "dangerous" file types to be downloaded, rather than allowing them to be viewed in the browser. These options are case-sensitive and defined in `com.atlassian.http.mime.DownloadPolicy`.

Events

These properties control the number of threads that are used for dispatching asynchronous events. Setting this number too high can decrease overall throughput when the system is under high load because of the additional overhead of context switching. Configuring too few threads for event dispatching can lead to events being queued up, thereby reducing throughput. These defaults scale the number of dispatcher threads with the number of available cpu cores.

Property	Description
`event.dispatcher.core.threads=0.8*cpu`	The minimum number of threads that is available to the event dispatcher. The `cpu` variable is resolved to the number of cpus that are available.
`event.dispatcher.max.threads=cpu`	The maximum number of event dispatcher threads. The number of dispatcher threads will only be increased when the event queue is full and max.threads has not been reached yet.
`event.dispatcher.queue.size=4096`	The number of events that can be queued. When the queue is full and no more threads can be created to handle the events, events will be discarded.
`event.dispatcher.keepAlive=60`	The time a dispatcher thread will be kept alive when the queue is empty and more than core.threads threads are running. Value is in SECONDS.

Executor

Controls the thread pool that is made available to plugins for asynchronous processing.

Property	Description
`executor.max.threads=100`	Specifies the maximum number of threads in the thread pool. When more threads are required than the configured maximum, the thread attempting to schedule an asynchronous task to be executed will block until a thread in the pool becomes available.
`executor.keepAliveTime=60`	Controls how long idle threads are kept alive. Threads idle for more than this time will be terminated. Value is in SECONDS and must be >= 1. If 0 or a negative value is used, a default value of 1 will be configured.

Features

Feature properties control high-level system features, allowing them to be disabled for the entire instance. Features that are disabled at this level are disabled *completely*. This means that instance-level configuration for a feature is overridden. It also means that a user's permissions are irrelevant; a feature is still disabled even if the user has the SYS_ADMIN permission.

Property	Description
`attachment.upload.max.size=10`	Controls the file size limit for individual attachments to pull request comments and descriptions. Value is in MB.
`feature.attachments=true`	Controls whether attachments can be added to pull request comments and descriptions.
`feature.auth.captcha=true`	Controls whether to require CAPTCHA verification when the number of failed logins is exceeded. If enabled, any client who has exceeded the number of failed logins allowed using either the Stash web interface or the Git hosting interface will be required to authenticate in the Stash web interface and successfully submit a CAPTCHA before continuing. Setting this to `false` will remove this restriction and allow users to incorrectly authenticate as many times as they like without penalty. ⚠️ **Warning**: It is STRONGLY recommended you keep this setting enabled. Disabling it will have the following ramifications: • Your users may lock themselves out of any underlying user directory service (LDAP, Active Directory etc) because Stash will pass through all authentication requests (regardless of the number of previous failures) to the underlying directory service. • For Stash installations where you use Stash for user management or where you use a directory service with no limit on the number of failed logins before locking out users, you will open Stash or the directory service up to brute-force password attacks.
`feature.forks=true`	Controls whether repositories can be forked. This setting *supersedes and overrides* instance-level configuration. If this is set to `false`, even repositories which are marked as forkable cannot be forked.

`feature.personal.repos=true`	Controls whether personal repositories can be ceated. When set to `false`, personal repository creation is disabled globally in Stash.
`feature.public.access=true`	Public access to Stash allows unauthenticated users to be granted access to projects and repositories for specific read operations including cloning and browsing repositories. This is normally controlled by project and repository administrators but can be switched off system wide by setting this property to `false`. This can be useful in highly sensitive environments.

Hibernate

Property	Description
`hibernate.format_sql=false`	When `hibernate.show_sql` is enabled, this flag controls whether Hibernate will format the output SQL to make it easier to read.
`hibernate.jdbc.batch_size=20`	Controls Hibernate's JDBC batching limit, which is used to make bulk processing more efficient (both for processing and for memory usage).
`hibernate.show_sql=false`	Used to enable Hibernate SQL logging, which may be useful in debugging database issues. This value should generally only be set by developers.

JIRA

Property	Description
`plugin.jira-integration.pullrequest.attribute.changesets.max=100`	Controls the maximum number of changesets to retrieve when retrieving attributes associated with changesets of a pull-request. This value should be between 50 and 1000 as Stash will enforce an lower bound of 50 issues and an upper bound of 1000 issues.

`plugin.jira-integration.remote.page.max.issues=20`	Controls the maximum number of issues to request from JIRA. This value should be between 5 and 50 as Stash will enforce a lower bound of 5 issues and an upper bound of 50 issues.
`plugin.jira-integration.remote.timeout.connection=5000`	The connection timeout duration in milliseconds for requests to JIRA. This timeout occurs if the JIRA server does not answer. e.g. the server has been shut down. This value should be between 2000 and 60000 as Stash will enforce a lower bound of 2000ms and an upper bound of 60000ms. Value is in MILLISECONDS.
`plugin.jira-integration.remote.timeout.socket=10000`	The socket timeout duration in milliseconds for requests to JIRA. This timeout occurs if the connection to JIRA has been stalled or broken. This value should be between 2000 and 60000 as Stash will enforce a lower bound of 2000ms and an upper bound of 60000ms. Value is in MILLISECONDS.

JMX

Property	Description
`jmx.enabled=true`	Controls the publishing of stash specific statistics via JMX. See Enabling JMX counters for performance monitoring.

Liquibase

Property	Description
`liquibase.commit.block.size=10000`	The maximum number of changes executed against a particular Liquibase database before a commit operation is performed. Very large values may cause DBMS to use excessive amounts of memory when operating within transaction boundaries. If the value of this property is less than one, then changes will not be committed until the end of the change set.

Logging

Logging levels for any number of loggers can be set in the `stash-config.properties` file using the following format:

```
logging.logger.<name>=<level>
```

For example, to configure all classes in the com.atlassian.stash package to DEBUG level:

```
logging.logger.com.atlassian.stash=DEBUG
```

To adjust the ROOT logger, you use the special name ROOT (case-sensitive):

```
logging.logger.ROOT=INFO
```

Notifications

Property	Description
`plugin.stash-notification.batch.min.wait.minutes=10`	Controls the minimum time to wait for new notifications before sending the batch. This is the inactivity timeout. Value is in MINUTES.
`plugin.stash-notification.batch.max.wait.minutes=30`	Controls the maximum time to wait for new notifications before sending the batch. This is the staleness timeout. Value is in MINUTES.

`plugin.stash-notification.mail.max.comment.size=2048`	Controls the maximum allowed size of a single comment in characters (not bytes). Extra characters will be truncated.
`plugin.stash-notification.mail.max.description.size=2048`	Controls the maximum allowed size of a single description in characters (not bytes). Extra characters will be truncated.
`plugin.stash-notification.mentions.enabled=true`	Controls whether notifications for mentions are enabled.
`plugin.stash-notification.max.mentions=200`	Controls the maximum number of allowed mentions in a single comment.
`plugin.stash-notification.sendmode.default=BATCHED`	Controls the system default for notifications batching for users who have not set an explicit preference. Value is either BATCHED or IMMEDIATE.

Paging

These properties control the maximum number of objects which may be returned on a page, regardless of how many were actually requested by the user. For example, if a user requests Integer.MAX_INT branches on a page, their request will be limited to the value set for `page.max.branches`.

This is intended as a safeguard to prevent enormous requests from tying up the server for extended periods of time and then generating responses whose payload is prohibitively large. The defaults configured here represent a sane baseline, but may be overridden if necessary.

Property	Description
`page.max.branches=1000`	Maximum number of branches per page.
`page.max.changes=1000`	Maximum number of changes per page. Unlike other page limits, this is a hard limit; subsequent pages cannot be requested when the number of changes in a changeset exceeds this size.
`page.max.changesets=100`	Maximum number of changesets (commits) per page.
`page.max.diff.lines=10000`	Maximum number of segment lines (of any type, total) which may be returned for a single diff. Unlike other page limits, this is a hard limit; subsequent pages cannot be requested when a diff exceeds this size.
`page.max.directory.children=500`	Maximum number of directory entries which may be returned for a given directory.

`page.max.directory.recursive.children=100000`	Maximum number of file entries which may be returned for a recursive listing of a directory. A relatively high number as this is used by the file finder which needs to load the tree of files upfront.
`page.max.groups=1000`	Maximum number of groups per page.
`page.max.index.results=50`	Maximum number of changesets which may be returned from the index when querying by an indexed attribute. For example, this limits the number of changesets which may be returned when looking up commits against a JIRA issue.
`page.max.projects=1000`	Maximum number of projects per page.
`page.max.repositories=1000`	Maximum number of repositories per page.
`page.max.source.length=5000`	Maximum length for any line returned from a given file when viewing source. This value truncates long lines. There is no mechanism for retrieving the truncated part short of downloading the entire file.
`page.max.source.lines=5000`	Maximum number of lines which may be returned from a given file when viewing source. This value breaks large files into multiple pages. See also Display above.
`page.max.tags=1000`	Maximum number of tags per page.
`page.max.users=1000`	Maximum number of users per page.
`page.max.pullrequests=100`	Maximum number of pull requests per page.
`page.scan.pullrequest.activity.size=500`	The size of the page Stash should use when scanning activities.
`page.scan.pullrequest.activity.count=4`	The number of pages of activities Stash should scan before giving up.

Password reset

Property	Description
`password.reset.validity.period=4320`	Controls how long a password reset token remains valid for. Default period is 72 hours. Value is in MINUTES.

Process execution

Property	Description

`process.timeout.execution=120`
 	
`process.timeout.idle=60`
	Controls timeouts for external processes, such as Git and Hg. The idle timeout configures how long the command is allowed to run without producing any output. The execution timeout configures a hard upper limit on how long the command is allowed to run even if it is producing output.
	
	Values are in SECONDS. Using 0, or a negative value, disables the timeout completely.
	
	USE AT YOUR OWN RISK!

Pull requests

Property	Description
`plugin.stash-scm-git.pullrequest.merge.strategy.KEY.slug=no-ff`	Control the merge strategy for a repository (where KEY and `slug` is the repository slug). Note that the URL for of a repository is of the following form: http://<stashdomain>/projects/<PROJECTKEY>/repos/ Overrides project and global settings. Possible values are: • `no-ff` – no fast-forward; the default setting. • `ff` – allow fast-forward; will merge when necessary • `ff-only` – require fast-forward; will never create m if a merge is required. • `squash` – collapse all incoming commits into a sing to the target branch; never create a merge. • `squash-ff-only` – collapse all the incoming com commit directly to the target branch, never creating *only* if the source branch is fast-forward.
`plugin.stash-scm-git.pullrequest.merge.strategy.KEY=no-ff`	Control the merge strategy for a project (where KEY is Overrides global settings. Is overridden by repository s Possible values are listed above.
`plugin.stash-scm-git.pullrequest.merge.strategy=no-ff`	Control the merge strategy globally. Is overridden by re project settings. Possible values are listed above.
`pullrequest.diff.context=10`	Defines the number of context lines to include around c request diffs. By default, Git only includes 3 lines. The c and include a bit more useful context around changes, "expand" the context is implemented.
`pullrequest.rescope.changesets.display=5`	Defines the maximum number of changesets per type (removed) to display in a rescope activity.
`pullrequest.rescope.changesets.max=1000`	Defines the absolute maximum number of changesets evaluated when attempting to determine, for a given re which changesets were added to or removed from a pu request. Adjusting this setting can have significant mem impact on the system. It is not recommended to be cha option is provided here to support unique use cases.

`pullrequest.rescope.detail.threads=2`	Defines the maximum number of threads to use for pre rescope details. These threads perform the requisite pr determine the commits added and removed when a pul rescoped, where most rescopes do not add or remove "dead" rescopes are deleted during processing. The pri ensure all details have already been calculated when u pull request's overview.
`pullrequest.rescope.drift.threads=4`	Defines the maximum number of threads to use when p comment drift for a pull request during rescope. Higher *not* necessarily mean higher throughput! Performing co usually force a new merge to be created, which can be Having a substantial number of merges running at the s significantly *reduce* the speed of performing comment d

Readme parsing

Property	Description
`plugin.stash-readme.max.size=65536`	Controls the maximum allowed size of a readme file to parse. Value is in BYTES.

Ref metadata

Property	Description
`ref.metadata.timeout=2`	Controls timeouts for retrieving metadata associated with a collection of refs from all metadata providers collectively. This values is in SECONDS.
`ref.metadata.max.request.count=100`	Controls the maximum number of refs that can be used in a metadata query.

Resource throttling

These properties define concurrent task limits for the ThrottleService, limiting the number of concurrent Git operations of a given type
that may be run at once. This is intended to help prevent Stash from overwhelming a server machine with running processes. 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).

When the limit is reached for the given resource, the request will wait until a currently running request has completed. If no request completes within a configurable timeout, the request will be rejected.

When requests while accessing the Stash UI are rejected, users will see either a 501 error page indicating the server is under load, or a popup indicating part of the current page failed.

When Git client 'hosting' commands (pull/push/clone) are rejected, Stash does a number of things:

- Stash will return an error message to the client which the user will see on the command line: "Stash is currently under heavy load and is not able to service your request. Please wait briefly and try your request

again"
- A warning message will be logged for every time a request is rejected due to the resource limits, using the following format:
  `"A [scm-hosting] ticket could not be acquired (12/12)"`
- For five minutes after a request is rejected, Stash will display a red banner in the UI to warn that the server is under load.

The hard, machine-level limits these are intended to prevent hitting are very OS- and hardware-dependent, so you may
need to tune them for your instance of Stash. When hyperthreading is enabled for the server CPU, for example, it is likely that the server will allow sufficient concurrent Git operations to completely bury the I/O on the machine. In such cases, we recommend starting off with a less aggressive default on multi-cored machines – the value can be increased later if hosting operations begin to back up. These defaults are finger-in-the-wind guesstimates (which so far have worked well).

Additional resource types may be configured by defining a key with the format `throttle.resource.<resource-name>`.
When adding new types, it is strongly recommended to configure their ticket counts explicitly using this approach.

Property	Description
`throttle.resource.scm-command=25`	Limits the number of operations that support the UI , such as `git diff` , `git blame` , or `git rev-list` , that can run concurrently. This is intended to prevent these SCM commands from competing with the running of push and pull operations.
`throttle.resource.scm-command.timeout=2`	Controls how long threads will wait for SCM commands to complete when the system is already running the maximum number of SCM commands. Value is in SECONDS.
`throttle.resource.scm-hosting=1.5*cpu`	Limits the number of SCM 'hosting' operations, such as `git clone`, `git push` and `git pull` over HTTP or SSH that may be running concurrently. This is intended primarily to prevent pulls, which can be very memory-intensive, from pinning a server's resources. There is limited support for mathematical expressions; +,-,*,\ and () are supported. You can also use the `cpu` variable which is resolved to the number of cpus that are available.
`throttle.resource.scm-hosting.timeout=300`	Controls how long threads will wait for SCM hosting operations to complete when the system is already running the maximum number of SCM hosting operations. Value is in SECONDS.
`throttle.resource.busy.message.timeout=5`	Controls how long a warning banner is displayed in the UI after a request is rejected due to excessive load. Value is in MINUTES. Using 0, or a negative value, disables displaying the banner. This is deprecated and replaced by `server.busy.on.ticket.rejected.within`, It is due to be removed in Stash 3.0.

SCM – Cache

See Scaling Stash for Continuous Integration performance for more information about using the SCM Cache Plugin for Stash.

Property	Description
`plugin.stash-scm-cache.expiry.check.interval=300`	Controls how frequently expired caches are checked and deleted from disk. Value is in SECONDS.
`plugin.stash-scm-cache.minimum.free.space=1073741824`	Controls how much space needs to be available on disk (specifically under `<Stash home directory>/caches`) for caching to be enabled. This setting ensures that the cache plugin does not fill up the disk. Value is in BYTES.
`plugin.stash-scm-cache.protocols=HTTP,SSH`	Controls which protocols caching is applied to. The `HTTP` value encapsulates both `http` and `https`.
`plugin.stash-scm-cache.refs.enabled=false`	Controls whether ref advertisement operations are cached.
`plugin.stash-scm-cache.refs.ttl=60`	Controls how long the caches for ref advertisements are kept around when there no changes to the repository. Caches are automatically invalidated when someone pushes to a repository or when a pull request is merged. Time is in SECONDS.
`plugin.stash-scm-cache.upload-pack.enabled=true`	Controls whether clone operations are cached.
`plugin.stash-scm-cache.upload-pack.ttl=14400`	Controls how long the caches for clone operations are kept around when there no changes to the repository. Caches are automatically invalidated when someone pushes to a repository or when a pull request is merged. Time is in SECONDS.

SCM – Git

Property	Description
`plugin.stash-scm-git.path.executable=git`	Defines the default path to the Windows machines, the .exe s configured value automatically general, "git" should be an acc platform, here, assuming that i runtime user's PATH. With the new path searching p DefaultGitBinaryHelper, setting unnecessary, as the plugin will This is left here purely for docu explicit path.
`plugin.stash-scm-git.path.libexec=`	Defines the path to the Git libe the git-core directory). This pat executable and is used for fork git-http-backend. If this value is out those processes. This elim (git -> git-http-backend) and m
`plugin.stash-scm-git.backend.http.buffer.size=32768`	Defines the buffer size in bytes marshalling data between the socket.
`plugin.stash-scm-git.backend.ssh.buffer.size=32768`	Defines the buffer size in bytes marshalling data between the socket.
`plugin.stash-scm-git.backend.timeout.idle=1800`	Defines the idle timeout for pus a limit to how long the operatio execute without either producir input. The default value is 30 m This value is in SECONDS.
`plugin.stash-scm-git.backend.timeout.execution=86400`	Defines the execution timeout applying a hard limit to how lor to run even if it is producing ou default value is 1 day. This value is in SECONDS.

`plugin.stash-scm-git.diff.renames=copies`	Defines whether copy and/or r performed. By default, both ren are performed. Only files modi considered as rename or copy overhead. The possible settings are: • `copy` or `copies` – applies • `rename` or `renames` – app • `off` – disables rename *and* When using `copy` or `copies`, suffixed with a "+" to use `--fi` This setting should be used wi expensive. It considers every f even files not modified in the s origins for copies. When copy and/or rename det `stash-scm-git.diff.rena` used control the similarity index required for a c copy or rename.
`plugin.stash-scm-git.diff.renames.threshold=50`	Defines the threshold, as a per detected as a rename or a cop applied if copy and/or rename default threshold applied is 50 itself). Git diff and Git diff-tree *do not* only) for the threshold. They ig and apply the default 50% thre threshold of 100 will be applied configured threshold that is 0, as 1.
`plugin.stash-scm-git.environment.variablesize=2000`	Defines the maximum number added to a single environment operating systems (and even d same operating system) have they apply to environment var intended to be low enough to w platforms out of the box, but st usable. It is configurable in cas on some platform.
`plugin.stash-scm-git.pullrequest.merge.auto.forceadd=false`	Defines whether conflicted files index using Git `add --force` By default, this behaviour is *off* However, when merging acros `.Gitignore` settings, enablin system to create a conflicted d the common ancestor will be s Note: This value has *no effect* merges. It is *only* applied durin for producing a pull request's d

`plugin.stash-scm-git.pullrequest.merge.auto.timeout=120`	Defines the maximum amount to perform a merge to support allowed to execute *or* idle. Bec generally do not produce outpu timeout. This value is in SECONDS.
`plugin.stash-scm-git.pullrequest.merge.real.timeout=300`	Defines the maximum amount to merge a pull request is allov Because the commands used output, there is no separate idl This value is in SECONDS.
`plugin.stash-scm-git.repository.size.timeout=75`	Defines the maximum amount the size of a single repository. repositories and/or remote stor value. This value is in MILLISECOND

Server busy banners

Property	Description
`server.busy.on.ticket.rejected.within=5`	Controls how long a warning banner is displayed in the UI after a request is rejected due to excessive load. Value is in MINUTES. Using 0, or a negative value, disables displaying the banner.
`server.busy.on.queue.time=60`	Controls how long requests need to be queued before they cause a warning banner to appear. Value is in SECONDS. Using 0, or a negative value, disables displaying the banner.

Setup automation

If these properties are specified in `stash-config.properties` when the Setup Wizard runs after installing Stash, then those values will be used, and the Setup Wizard will not display the corresponding configuration screens.

You can use these properties to automate Stash setup and remove the need to interact with the Stash Setup Wizard when provisioning Stash. See Automated setup for Stash.

Property	Description
`setup.displayName=displayName`	The display name for the Stash instance.
`setup.baseUrl= https://stash.yourcompany.com`	The base URL to use for the Stash instance.

`setup.license=AAAB...\u000a1ev...\u000aA4N...`	The Stash license. Use the the `\u000` character to *not* b reak the license over multiple lines.
`setup.sysadmin.username=username`	Credentials for the system admin account.
`setup.sysadmin.password=password`	
`setup.sysadmin.displayName=John Doe`	The display name for the system admin account.
`setup.sysadmin.emailAddress=sysadmin@yourcompany.com`	The email address for the system admin account.

SMTP

Property	Description

`mail.timeout.connect=60`
`mail.timeout.send=60`
`mail.test.timeout.connect=30`
`mail.test.timeout.send=30`
	Controls timeouts for establishing an SMTP connection and sending an e-mail. Shorter timeouts should be applied for when sending test e-mails, as the test occurs in user time.
	Values are in SECONDS.

`mail.error.pause.log=300`	Controls how frequently logs will go to the standard log file about mail sending errors. All errors are logged to the `atlassian-stash-mail.log` file, but Stash will periodically log a warning to the standard log file if there are errors sending messages. Value is in SECONDS.
`mail.error.pause.retry=5`	Controls how long Stash will wait before retrying to send a message if an error occurs. Value is in SECONDS.
`mail.threads=1`	Controls the number of threads to use for sending emails. Setting this to a higher value will put greater load on your mail server when Stash generates a lot of emails, but will make Stash clear its internal queue faster.
`mail.max.message.size=1048576`	Controls the maximum allowed size of a single mail message, which is the sum of the subject and body sizes. Value is in BYTES.
`mail.max.queue.size=157286400`	Controls the maximum allowed size for the mail queue (any new message will be rejected if the mail queue reaches that size). Value is in BYTES.

SSH command execution

Property	Description

`plugin.ssh.command.timeout.idle=86400`	Controls timeouts for all SSH commands, such as those that service Git and hg operations over SSH. The idle timeout configures how long the command is allowed to run without writing any output to the client. For SCM commands, the `plugin.*.backend.timeout.idle` properties defined above will be applied to the underlying process. The default value is 1 day. Value is in SECONDS.

SSH security

Property	Description
`plugin.ssh.disabled.ciphers`	Controls which default ciphers are disabled when executing all SSH commands. Non existent ciphers are ignored. Names are case sensitive. Example value: `arcfour128,3des-cbc` To enable additional ciphers see the KB article Disable default SSH algorithms.
`plugin.ssh.disabled.key.exchanges`	Controls which default key exchange algorithms are disabled when executing all SSH commands. Non existent key exchange algorithms are ignored. Names are case sensitive. Example value: `ecdh-sha2-nistp256,ecdh-sha2-nistp384` To enable additional key exchange algorithms see the KB article Disable default SSH algorithms.
`plugin.ssh.disabled.macs`	Controls which default macs are disabled when executing all SSH commands. Non existent macs are ignored. Names are case sensitive. Example value: `hmac-sha1-96,hmac-md5-96,hmac-md5` To enable additional macs see the KB article Disable default SSH algorithms.

Syntax highlighting

See Configuring syntax highlighting for file extensions for more information.

Stash applies syntax highlighting to diffs as well as source files.

Property	Description
`syntax.highlighter.<MIME type>.executables=exe1,exe2`	Controls the language highlighter used for a given set of hashbang executables. The `<MIME type>` refers to the MIME type CodeMirror uses.
`syntax.highlighter.<MIME type>.extensions=ext1,ext2`	Controls the language highlighter used for a given set of file extensions. The `<MIME type>` refers to the MIME types CodeMirror uses.

Webhooks

See POST service webhook for Stash for more information.

Property	Description
`plugin.com.atlassian.stash.plugin.hook.threadPoolCoreSize=2`	Core size of thread pool – the default number of concurrent hooks notifications.
`plugin.com.atlassian.stash.plugin.hook.threadPoolMaxSize=3`	Maximal size of thread pool – the maximum number of concurrent hooks notifications.
`plugin.com.atlassian.stash.plugin.hook.queueSize=1024`	The maximum size of the queue which holds queued requests that are yet to be sent. When this size is exceeded the oldest unsent message will be dropped and a warning message logged.
`plugin.com.atlassian.stash.plugin.hook.connectionTimeout=10000`	Connection timeout for hook request in MILLISECONDS. When the connection times out a warning message will be logged.
`plugin.com.atlassian.stash.plugin.hook.changesetsLimit=500`	Limit of maximum count of changesets that will be sent in the POST data for a single ref change.
`plugin.com.atlassian.stash.plugin.hook.changesLimit=100`	Limit of maximum count of changes for a single changeset in the POST data.

Proxying and securing Stash

This page provides an overview of some common network topology options for running Stash, including running Stash behind a reverse proxy and securing access to Stash by using HTTPS (HTTP over SSL).

Note that Stash does not need to run behind a web server – it is capable of serving web requests directly using the bundled Tomcat application server. On this page, 'connecting to Stash' really means connecting to Tomcat, which is used to serve Stash content.

On this page:

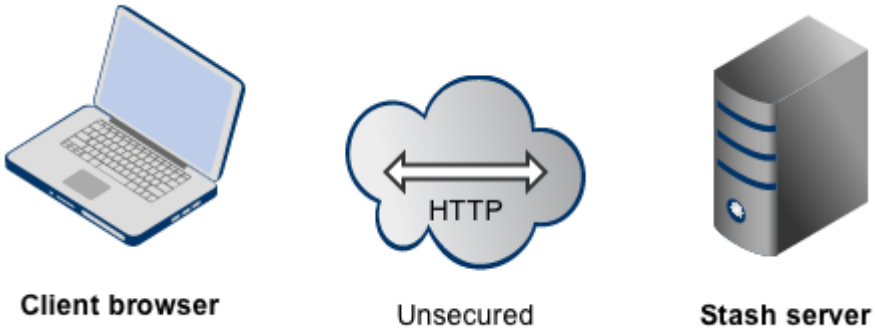
- Connecting to Stash directly over HTTP
- Securing access to Stash using HTTPS
- Using a reverse proxy for Stash
- Securing a reverse proxy using HTTPS

Connecting to Stash directly over HTTP

Connecting directly to Stash (that is, Tomcat) is the default install configuration, as described in the Stash install documentation:

- Getting started

When set up this way, the user accesses Stash directly over HTTP, without using SSL – all communication between the user's browser and Stash will be unsecured.



You may also wish to consider the following:

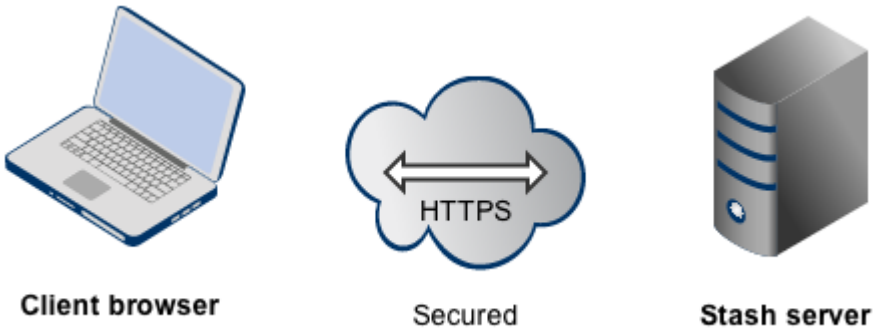
- Stash, by default, will listen for requests on port 7990 – this port can be changed if required.
- The address with which to access Stash, by default, will be http://<computer name>:7990. Change the base URL for Stash if required.
- You can set the context path for Stash if you are running another Atlassian application, or Java web application, at the same hostname and context path as Stash.
- Securing Git operations between the user's computer and Stash is a separate consideration - see Enabling SSH access to Git.

Securing access to Stash using HTTPS

Access to Stash can be secured by enabling HTTPS (HTTP over SSL) for the Tomcat application server that is bundled with Stash. You should consider doing this, and making secure access mandatory, if Stash will be internet-facing and usernames, passwords and other proprietary data may be at risk.

When set up in this way, access to Stash is direct, and all communication between the user's browser and Stash will be secured using SSL.

See Securing Stash with Tomcat using SSL for configuration details.



Note that:

- Stash will listen for requests on port 8443. This port can be changed if required.
- The address with which to access Stash, by default, will be https://<computer name>:8443. Change the base URL for Stash if required.

- Any existing links with other applications will need to be reconfigured using this new URL for Stash.
- You can set the context path for Stash if you are running another Atlassian application, or Java web application, at the same hostname and context path as Stash.
- Securing Git operations between the user's computer and Stash is a separate consideration - see Enabling SSH access to Git.

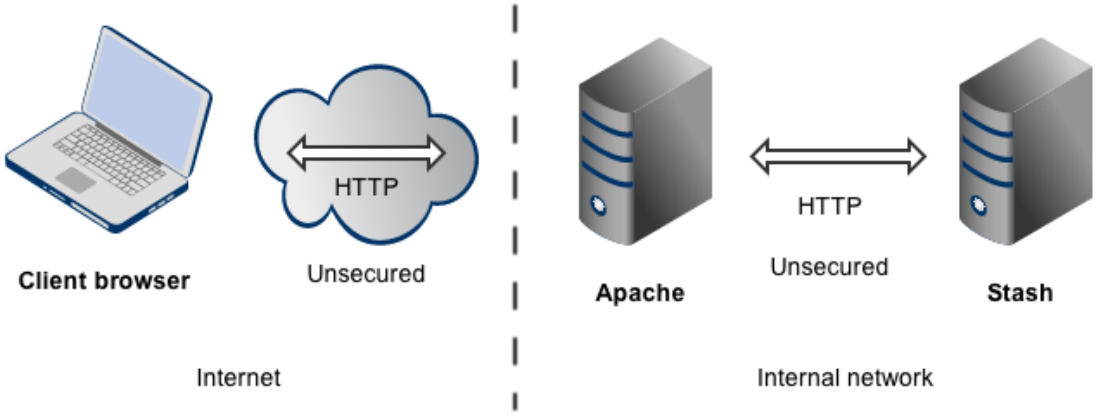
Using a reverse proxy for Stash

You can run Stash behind a reverse proxy, for example Apache HTTP Server. You may wish to do this if you want to:

- use a different port number to access Stash, particularly if you are Integrating JIRA Cloud with Stash.
- use a different context path to access Stash

When set up this way, external access to Stash is via a reverse proxy, without using SSL. All communication between the user's browser and Apache, and so Stash, will be unsecured, but users do not have direct access to Stash. An example scenario is where Apache provides a gateway through which users outside the firewall can access Stash.

See Integrating Stash with Apache HTTP Server for configuration details.



Note that:

- Stash, by default, will listen for requests on port 7990 – this port can be changed if required.
- Stash (Tomcat) needs to know the URL (proxy name) that Apache serves.
- The address with which to access Stash will be http://<proxy name>:7990. Change the base URL for Stash if required.
- Any existing links with other applications will need to be reconfigured using this new URL for Stash.
- You can set the context path for Stash if you are running another Atlassian application, or Java web application, at the same hostname and context path as Stash.
- Securing Git operations between the user's computer and Stash is a separate consideration - see Enabling SSH access to Git.

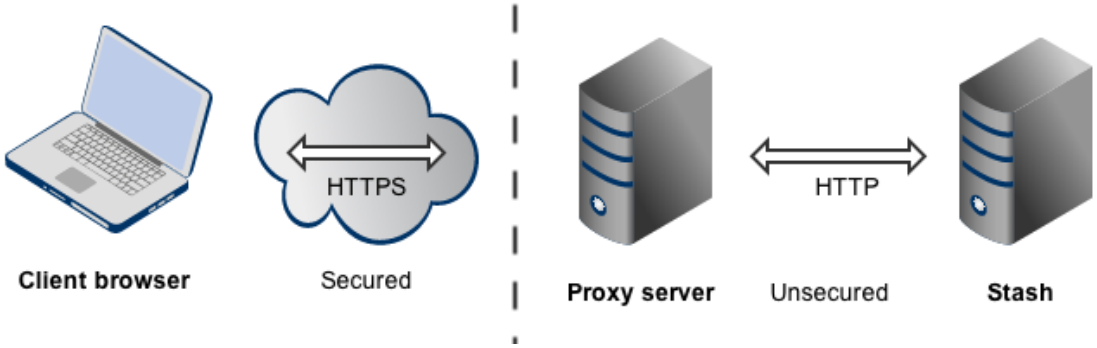
Securing a reverse proxy using HTTPS

You can run Stash behind a reverse proxy, such as Apache HTTP Server or nginx, that is secured using HTTPS (HTTP over SSL). You should consider doing this, and making secure access mandatory, if usernames, passwords and other proprietary data may be at risk. An example scenario is where Apache HTTP Server provides a gateway through which users outside the firewall can access Stash.

When set up in this way, external access to Stash is via a reverse proxy, where external communication with the proxy uses HTTPS. All communication between the user's browser and the reverse proxy will be secured, whereas communication between the proxy and Stash will not be secured (it doesn't use SSL).

See the following pages for configuration details:

- Securing Stash with Apache using SSL
- Securing Stash behind nginx using SSL



Note that:

- The reverse proxy (for example, Apache) will listen for requests on port 443.
- Stash, by default, will listen for requests on port 7990. Stash (Tomcat) needs to know the URL (proxy name) that the proxy serves.
- The address with which to access Stash will be https://<proxyName>:<proxyPort>/<context path>, for example https://mycompany.com:443/stash
- Any existing links with other applications will need to be reconfigured using this new URL for Stash.
- Stash (Tomcat) should be configured to refuse requests on port 7990 and to redirect those to the proxy on port 443.
- Securing Git operations between the user's computer and Stash is a separate consideration - see Enabling SSH access to Git.
- It would be possible to set up an SSL connection between the proxy server and Tomcat (Stash), but that configuration is very unusual, and not recommended in most circumstances.
- Incidentally, note that Stash 2.10 and later versions do not support `mod_auth_basic`.

Securing Stash with Tomcat using SSL

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

If you are setting up a production instance of Stash you should consider using a CA certificate, briefly described below.

There are other network topology options for running Stash, including running Stash behind a reverse proxy. For an overview of some common options, see Proxying and securing Stash.

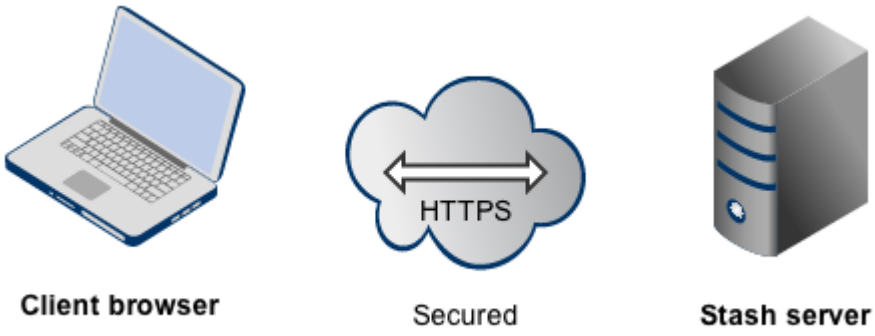
When Stash is set up following the instructions on this page, access to Stash is direct, and all communication between the user's browser and Stash will be secured using SSL.

On this page:

- 1. Generate a self-signed certificate
- 2. Configure HTTPS in Tomcat
- Exporting the self-signed certificate
- Requesting a CA certificate
- Troubleshooting

Related pages:

- Integrating Stash with Apache HTTP Server
- Securing Stash with Apache using SSL



Note that:

- Stash will listen for requests on port 8443. This port can be changed if required.
- The address with which to access Stash, by default, will be https://<computer name>:8443. Change the base URL for Stash if required.
- Any existing links with other applications will need to be reconfigured using this new URL for Stash.
- You can set the context path for Stash if you are running another Atlassian application, or Java web application, at the same hostname and context path as Stash.
- Securing Git operations between the user's computer and Stash is a separate consideration - see Enabling SSH access to Git.

> Please note that Atlassian Support will refer SSL-related support to the issuing authority for the certificate. The documentation on this page is for reference only.

1. Generate a self-signed certificate

Self-signed certificates are useful where you require encryption but do not need to verify the website identity. They are commonly used for testing and on internal corporate networks (intranets). If you are setting up a production instance of Stash you should consider using a CA certificate, briefly described below.

Users may receive a warning that the site is untrusted and have to "accept" the certificate before they can access the site. This usually will only occur the first time they access the site.

The following approach to creating a certificate uses Java's keytool. Other tools for generating certificates are available.

To generate a self-signed certificate:

Log in with the user account that Stash will run under, and run the following command:

Windows	`"%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA`
Linux, Mac OS X	`$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA`

This will create (if it doesn't already exist) a new `.keystore` file located in the home directory of the user you used to run the keytool command.

If you used the Stash installer to install Stash as a service on your system, the installer will have created a user account called `atlstash`. This account is locked (it cannot be used to log in to the system) and doesn't have a home directory. In this case you need to specify a location for the `.keystore` file using the `keystore` parameter like this:

Windows	`"%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA -keystore /path/to/keystore/stash.jks`
Linux, Mac OS X	`$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA -keystore /path/to/keystore/stash.jks`

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 `<Stash home directory>/shared/server.xml` by adding the following attribute to the `<Connector/>` tag: `keystorePass="<password value>"`

2. Configure HTTPS in Tomcat

To configure HTTPS in Tomcat:

1. Edit `<Stash home directory>/shared/server.xml` and, at the bottom, before the `</Service>` tag, add this section (or uncomment this if it already exists):

```
<Connector port="8443"
  maxHttpHeaderSize="8192"
  SSLEnabled="true"
 maxThreads="150"
  minSpareThreads="25"
  maxSpareThreads="75"
 enableLookups="false"
  disableUploadTimeout="true"
  useBodyEncodingForURI="true"
 acceptCount="100"
  scheme="https"
  secure="true"
 clientAuth="false"
  sslProtocol="TLS" />
```

   This enables SSL access on port 8443 (the default for HTTPS is 443, but 8443 is used here instead of 443 to avoid conflicts).

   If you created the keystore somewhere else on the filesystem, add the `keystoreFile` attribute to the connector tag as well:

```
keystoreFile="/path/to/keystore/stash.jks"
```

2. Comment out the existing Connector directive for port 7990 in `<Stash home directory>/shared/server.xml`, so as to disable HTTP access, if you want all access to Stash to make use of HTTPS. That is, comment out this directive:

```
<Connector port="7990"
    protocol="HTTP/1.1"
    connectionTimeout="20000"
    useBodyEncodingForURI="true"
    redirectPort="8443"
    compression="on"

compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,
application/javascript,application/x-javascript" />
```

3. Start, or re-start, Stash. You will be able to access Stash at https://localhost:8443/ in your browser.

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:

Windows	`"%JAVA_HOME%\bin\keytool" -export -alias tomcat -file file.cer`
Linux, Mac OS X	`$JAVA_HOME/bin/keytool -export -alias tomcat -file file.cer`

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, DigiCert or Thawte. The instructions below are adapted from the Tomcat documentation.

First, you will generate a local certificate and create a 'certificate signing request' (CSR) based on that certificate. You then submit the CSR to your chosen certificate authority. The CA will use that CSR to generate a certificate for you.

1. Use Java's `keytool` utility to generate a local certificate, as described in the section above.
2. Use the `keytool` utility to generate a CSR, replacing the text `<MY_KEYSTORE_FILENAME>` with the path to and file name of the `.keystore` file generated for your local certificate:

Windows	`"%JAVA_HOME%\bin\keytool" -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <MY_KEYSTORE_FILENAME>`
Linux, Mac OS X	`$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, Mac OS X	`$JAVA_HOME/bin/keytool -import -alias tomcat -file file.cer`

Troubleshooting

Here are some troubleshooting tips if you are using a self-signed key created by keytool, or a CA certificate, as described above.

When you enter "https://localhost:8443/" in your browser, if you get a message such as "Cannot establish a connection to the server at localhost:8443", look for error messages in your `logs/catalina.out` log file. Here are some possible errors with explanations:

SSL + Apache + IE problems

Some people have reported errors when uploading attachments over SSL using Internet Explorer. This is due to an IE bug, and can be fixed in Apache by setting:

```
BrowserMatch ".MSIE." \
 nokeepalive ssl-unclean-shutdown \
 downgrade-1.0 force-response-1.0
```

Google has plenty more on this.

Can't find the keystore

```
java.io.FileNotFoundException: /home/user/.keystore (No such file or directory)
```

This indicates that Tomcat cannot find the keystore. The keytool utility creates the keystore as a file called `.keystore` in the current user's home directory. For Unix and 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 `<Stash home directory>/shared/server.xml`. Add the following attribute to the connector tag you uncommented:

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

Incorrect password

```
java.io.IOException: Keystore was tampered with, or password was incorrect
```

You used a different password than "changeit". You must either use "changeit" for both the keystore password and for the key password for Tomcat, or if you want to use a different password, you must specify it using the `keystorePass` attribute of the Connector tag, as described above.

Passwords don't match

```
java.io.IOException: Cannot recover key
```

You specified a different value for the keystore password and the key password for Tomcat. Both passwords

must be the same.

Wrong certificate

```
javax.net.ssl.SSLException: No available certificate corresponds to the SSL cipher
suites which are enabled.
```

If the Keystore has more than one certificate, Tomcat will use the first returned unless otherwise specified in the SSL Connector in `<Stash home directory>/shared/server.xml`.

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

```
<Connector port="8443"
  maxHttpHeaderSize="8192"
 SSLEnabled="true"
 maxThreads="150"
  minSpareThreads="25"
  maxSpareThreads="75"
 enableLookups="false"
  disableUploadTimeout="true"
  useBodyEncodingForURI="true"
 acceptCount="100"
  scheme="https"
  secure="true"
 clientAuth="false"
  sslProtocol="TLS"
 keystoreFile="/opt/local/.keystore"
 keystorePass="removed"
 keyAlias="tomcat"/>
```

Using Apache Portable Runtime

APR uses a different SSL engine, and you will see an exception like this in your logs:

```

```

The reason for this is that the APR Connector uses OpenSSL and cannot use the keystore in the same way. You can rectify this in one of two ways:

Use the Http11Protocol to handle SSL connections

Edit the server.xml so that the SSL Connector tag you just uncommented specifies the Http11Protocol instead of the APR protocol:

```
<Connector port="8443"
 protocol="org.apache.coyote.http11.Http11Protocol"
maxHttpHeaderSize="8192"
 SSLEnabled="true"
 keystoreFile="${user.home}/.keystore"
maxThreads="150"
 enableLookups="false"
 disableUploadTimeout="true"
acceptCount="100"
 scheme="https"
 secure="true"
clientAuth="false"
 sslProtocol="TLS"
 useBodyEncodingForURI="true" />
```

Configure the Connector to use the APR protocol

This is only possible if you have PEM encoded certificates and private keys. If you have used OpenSSL to generate your key, then you will have these PEM encoded files - in all other cases contact your certificate provider for assistance.

```
<Connector port="8443"
 maxThreads="200"
scheme="https"
 secure="true"
 SSLEnabled="true"
SSLCertificateFile="${user.home}/certificate.pem"
SSLCertificateKeyFile="${user.home}/key.pem"
clientAuth="optional"
 SSLProtocol="TLSv1"/>
```

Enabling client authentication

To enable client authentication in Tomcat, ensure that the value of the `clientAuth` attribute in your `Connecto r` element of your Tomcat's `server.xml` file is `true`.

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

For more information about `Connector` element parameters, please refer to the 'SSL Support' section of the To mcat 6.0 documentation.

Wrong certificate type

If the certificate from the CA is in PKSC12 format, add the `keystoreType` attribute to the SSL Connector in <S `tash home directory>/shared/server.xml`.

```
keystoreFile="/opt/local/wildcard_atlassian_com.p12"
keystorePass="removed"
keystoreType="PKCS12"/>
```

Certificate chain is incomplete

If the root certificate and intermediary certificate(s) aren't imported into the keystore before the entity/domain

certificate, you will see the following error:

```
[root@dev atlas]# /usr/java/jdk1.7.0_17/bin/keytool -import -alias tomcat -file
my_entity_cert.crt
Enter keystore password:
keytool error: java.lang.Exception: Failed to establish chain from reply
```

Most likely, the CA sent a compressed file containing several certificates. The import order matters so you must import the root certificate first, followed by one or many intermediate certificates, followed lastly by the entity/domain certificate. There are many resources online that provide guidance for certificate installation for Tomcat (Java-based) web servers using keytool.

Integrating Stash with Apache HTTP Server

This page explains how to establish a network topology in which Apache HTTP Server acts as a reverse proxy f or 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. You may wish to do this if you want to:

- use a different port number to access Stash
- use a different context path to 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 Tomcat using SSL. For an overview of other network topology options, see Proxying and securing Stash. Otherwise, if you want to install Stash in an environment that incorporates Apache HTTP Server, this document is for you.

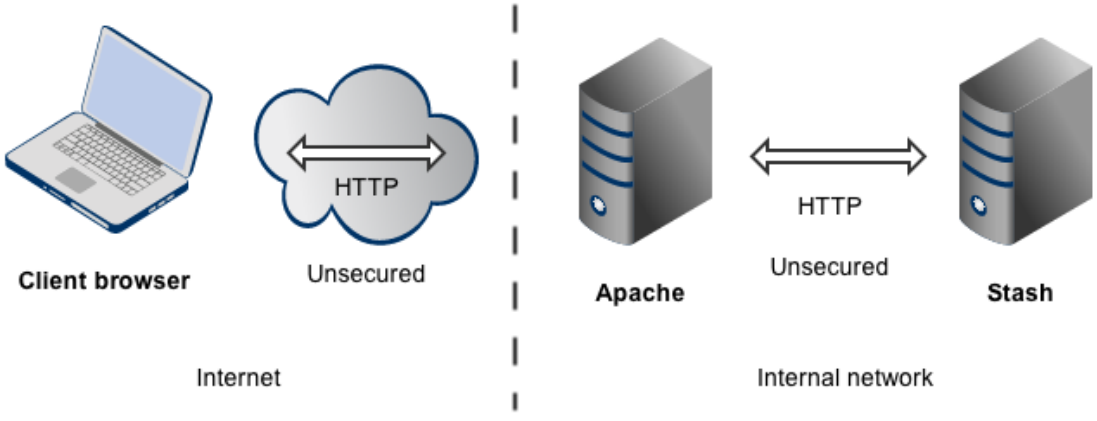
When Stash is set up following the instructions on this page, external access to Stash uses a reverse proxy, without using SSL. All communication between the user's browser and Apache, and so Stash, will be unsecured, but users do not have direct access to Stash.

On this page:

- About using Apache software
- Step 1: Configure the Tomcat Connector
- Step 2: Change Stash's base URL
- Step 3 (optional): Set a context path for Stash
- Step 4: Enable mod_proxy and mod_proxy_http in Apache HTTP Server
- Step 5: Configure mod_proxy to map requests to Stash
- Step 6: Configure mod_proxy to disable forward proxying
- Step 7: Allow proxying to Stash from everywhere
- Step 8 (optional): Configure Apache HTTP Server for SSL
- A note about application links
- Troubleshooting

Related pages:

- Securing Stash with Apache using SSL
- Securing Stash with Tomcat using SSL



Note that:

- Stash, by default, will listen for requests on port 7990 – this port can be changed if required.
- Stash (Tomcat) needs to know the URL (proxy name) that Apache serves.
- The address with which to access Stash will be http://<proxy name>:7990. Change the base URL for Stash if required.
- Any existing links with other applications will need to be reconfigured using this new URL for Stash.
- You can set the context path for Stash if you are running another Atlassian application, or Java web application, at the same hostname and context path as Stash.
- Securing Git operations between the user's computer and Stash is a separate consideration - see Enabling SSH access to Git.

About using Apache software

This section has general information pertaining to the use of Apache HTTP Server and Apache Tomcat. It is important that you read this section before proceeding to the steps that follow.

Configuring Tomcat 7

The Stash distribution includes an instance of Tomcat 7, the configuration of which is determined by the contents of the `<Stash home directory>/shared/server.xml` file. Note that any changes that you make to the `server.xml` file will only be effective upon starting or re-starting Stash.

You may find it helpful to refer to the Apache Tomcat 7.0 Proxy Support HowTo page.

Configuring Apache HTTP Server

> Since Apache HTTP Server is not an Atlassian product, Atlassian does not guarantee to provide support for its configuration. You should consider the material on this page to be for your information only; use it at your own risk. If you encounter problems with configuring Apache HTTP Server, we recommend that you refer to the Apache HTTP Server Support page.
>
> Note that Stash 2.10 and later versions do not support `mod_auth_basic`.

You may find it helpful to refer to the Apache HTTP Server Documentation, which describes how you can control Apache HTTP Server by changing the contents of the `httpd.conf` file. The section on Apache Module mod_proxy is particularly relevant. Note that any changes you make to the `httpd.conf` file will only be effective upon starting or re-starting Apache HTTP Server.
This document relates to Apache HTTP Server version 2.4.2; the configuration of other versions may differ.

Step 1: Configure the Tomcat Connector

Find the normal (non-SSL) `Connector` directive in Tomcat's `<Stash home directory>/shared/server.xml` file, and add the `scheme`, `proxyName`, and `proxyPort` attributes as shown below. Instead of `mycompany.com`, set the `proxyName` attribute to your domain name that Apache HTTP Server will be configured to serve. This informs 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 URLs.

```
<Connector port="7990"
    protocol="HTTP/1.1"
    connectionTimeout="20000"
    useBodyEncodingForURI="true"
    redirectPort="8443"
    compression="on"

compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,appli
cation/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 7.0 HTTP Connector Reference.

Step 2: Change Stash's base URL

After re-starting Stash, open a browser window and log into Stash using an administrator account. Go to the Stash administration area and click **Server settings** (under 'Settings'), and change **Base URL** to match the proxy URL (the URL that Apache HTTP Server will be serving).

Step 3 (optional): Set a context path for 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 `<Stash home directory>/shared/server.xml` file:

```
<Context path="/stash" docBase="${catalina.home}/atlassian-stash"
reloadable="false" useHttpOnly="true">
    ....
</Context>
```

If you do set a context path, it is important that the same path be used in Step 5, when setting up the `ProxyPass` and `ProxyPassReverse` directives. You should also append the context path to Stash's base URL (see Step 2).

Step 4: Enable `mod_proxy` and `mod_proxy_http` in Apache HTTP Server

In the `mod_proxy` documentation, you will read that `mod_proxy` can be used as a forward proxy, or as a reverse proxy (gateway); you want the latter. Where the `mod_proxy` documentation mentions '*origin server*', it refers to your 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 5: Configure `mod_proxy` to map requests to Stash

To configure `mod_proxy` for use with Stash, you need to use the `ProxyPass` and `ProxyPassReverse` directiv es 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 3, you'll need to use that context path in your `ProxyPass` and `Pro xyPassReverse` directives. Suppose your context path is set to "`/stash`", the directives would be as follows:

```
ProxyPass        /stash http://localhost:7990/stash connectiontimeout=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 connectiontimeout=5
timeout=300
ProxyPassReverse /stash http://private.mycompany.com:9900/stash
```

Step 6: Configure `mod_proxy` to disable forward proxying

If you are using Apache HTTP Server as a reverse proxy only, and not as a forward proxy server, you should turn forward proxying off by including a `ProxyRequests` directive in the `httpd.conf` file, as follows:

```
ProxyRequests Off
```

Step 7: Allow proxying to 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:

```
<Proxy *>
    Order Deny,Allow
    Allow from all
</Proxy>
```

The `Proxy` directive provides a context for the directives that are contained within its delimiting tags. In this case, we specify a wild-card url (the asterisk), which applies the two contained directives to all proxied requests.

The `Order` directive controls the order in which any `Allow` and `Deny` directives are applied. In the above configuration, we specify "`Deny,Allow`", which tells Apache HTTP Server to apply any `Deny` directives first, and if any match, the request is denied unless it also matches an `Allow` directive. In fact, "`Deny,Allow`" is the default; we include it merely for the sake of clarity. Note that we specify one `Allow` directive, which is described below, and don't specify any `Deny` directives.

The `Allow` directive, in this context, controls which hosts can access Stash via Apache HTTP Server. Here, we specify that all hosts are allowed access to Stash.

Step 8 (optional): Configure Apache HTTP Server for SSL

If you want to set up SSL access to Stash, follow the instructions on Securing Stash with Apache using SSL. When you are finished, users will be able to make secure connections to Apache HTTP Server; connections between Apache HTTP Server and 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.

A note about application links

When an application link is established between Stash and another Atlassian product (for example JIRA), and Stash is operating behind Apache HTTP Server, the link from the other product to Stash must be via the proxy URL; that is, the 'reciprocal URL' from, say JIRA, to Stash must match the proxy name and port that you set at Step 1.

Troubleshooting

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.

In particular:

- On **Fedora Core 4** people have reported 'permission denied' errors when trying to get mod_proxy (and mod_jk) working. Disabling SELinux (`/etc/selinux/config`) apparently fixes this.
- Some users have reported problems with user sessions being hijacked when the mod_cache module is enabled. If you have such problems, disable the mod_cache module. Note that this module is enabled by default in some Apache HTTP Server version 2 distributions.

Securing Stash with Apache using SSL

You can run Stash behind a reverse proxy, such as Apache HTTP Server or nginx, that is secured using HTTPS (HTTP over SSL). You should consider doing this, and making secure access mandatory, if usernames, passwords and other proprietary data may be at risk.

There are other network topology options for running Stash; for an overview of some common options, see Proxying and securing Stash.

When Stash is set up following the instructions on this page, external access to Stash is via Apache HTTP

Server as a reverse proxy, where external communication with the proxy uses HTTPS. All communication between the user's browser and Apache will be secured, whereas communication between Apache and Stash will not be secured (it doesn't use SSL).

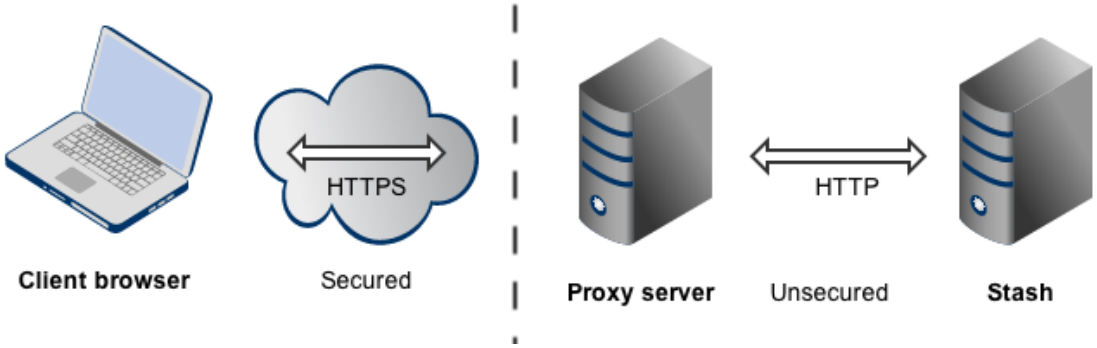
- The steps on this page would normally be performed after integrating Stash with Apache HTTP Server.

On this page:

- Step 1: Configure the Tomcat Connector for SSL
- Step 2: Set up a virtual host in Apache HTTP Server
- Step 3: Create SSL certificate and key files
- Step 4: Update the base URL to use HTTPS
- Using a self-signed certificate

Related pages:

- Integrating Stash with Apache HTTP Server
- Securing Stash with Tomcat using SSL
- Securing Stash behind nginx using SSL



Note that:

- The reverse proxy (for example, Apache) will listen for requests on port 443.
- Stash, by default, will listen for requests on port 7990. Stash (Tomcat) needs to know the URL (proxy name) that the proxy serves.
- The address with which to access Stash will be https://<proxyName>:<proxyPort>/<context path>, for example https://mycompany.com:443/stash
- Any existing links with other applications will need to be reconfigured using this new URL for Stash.
- Stash (Tomcat) should be configured to refuse requests on port 7990 and to redirect those to the proxy on port 443.
- Securing Git operations between the user's computer and Stash is a separate consideration - see Enabling SSH access to Git.
- It would be possible to set up an SSL connection between the proxy server and Tomcat (Stash), but that configuration is very unusual, and not recommended in most circumstances.
- Incidentally, note that Stash 2.10 and later versions do not support `mod_auth_basic`.

Step 1: Configure the Tomcat Connector for SSL

Find the normal (non-SSL) `Connector` directive in Tomcat's `<Stash home directory>/shared/server.xml` file, and change the `redirectPort`, `scheme`, `proxyName` and `proxyPort` attributes as follows:

```
<Connector port="7990"
     protocol="HTTP/1.1"
     connectionTimeout="20000"
     useBodyEncodingForURI="true"
     redirectPort="443"
     compression="on"

compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,appli
cation/javascript,application/x-javascript"
     secure="true"
     scheme="https"
     proxyName="mycompany.com"
     proxyPort="443" />
```

The `redirectPort` directive causes Tomcat-initiated redirections to secured resources to use the specified port. Right now, the 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.

Start, or restart, Stash.

Step 2: Set up a virtual host in Apache HTTP Server

Un-comment the following LoadModule directive in Apache HTTP Server's `httpd.conf` file:

```
LoadModule ssl_module modules/mod_ssl.so
```

Add the following directives to the `httpd.conf` file:

```
Listen 443
<VirtualHost *:443>
    SSLEngine On
    SSLCertificateFile    "/usr/local/apache2/conf/server.crt"
    SSLCertificateKeyFile "/usr/local/apache2/conf/server.key"
 SSLCertificateChainFile "/usr/local/apache2/conf/server.crt"
    ProxyPass        / http://localhost:7990/ connectiontimeout=5 timeout=300
    ProxyPassReverse / http://localhost:7990/
</VirtualHost>
```

The `Listen` directive instructs Apache HTTP Server to listen for incoming requests on port 443. Actually, we could omit that directive in this case, since Apache HTTP Server listens for `https` requests on port 443 by default. Nevertheless, it's good to make one's intentions explicit.

The `VirtualHost` directive encloses a number of child directives that apply only and always to requests that arrive at port 443. Since our `VirtualHost` block does not include a `ServerName` directive, it inherits the server name from the main server configuration.

The `SSLEngine` directive toggles the use of the SSL/TLS Protocol Engine. In this case, we're using it to turn SSL on for all requests that arrive at port 443.

The `SSLCertificateFile` directive tells Apache HTTP Server where to find the PEM-encoded certificate file for the server.

The `SSLCertificateKeyFile` directive tells Apache HTTP Server where to find the PEM-encoded private key file corresponding to the certificate file identified by the `SSLCertificateFile` directive. Depending on how the certificate file was generated, it may contain a RSA or DSA private key file, making the `SSLCertificateKeyFile` directive redundant; however, Apache strongly discourages that practice. The recommended

approach is to separate the certificate and the private key. If the private key is encrypted, Apache HTTP Server will require a pass phrase to be entered when it starts up.

The `SSLCertificateChainFile` is optional. Please consult with the CA vendor to verify if this is required. This directive sets the optional all-in-one file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate.

The `ProxyPass` and `ProxyPassReverse` directives should be set up in the manner described in Step 5 of the Integrating Stash with Apache HTTP Server page. In particular, if Stash is to run on a separate machine from Apache, you should use that domain (and perhaps the port number and context path) in the `ProxyPass` and ProxyPassReverse directives.

For more information about the support for SSL in Apache HTTP Server, refer to the Apache SSL/TLS Encryption manual. In addition, you will find lots of relevant information in the `<apache directory>/conf/extra/httpd-ssl.conf` file, which is included in the standard Apache distribution.

Start, or restart, Apache.

Step 3: Create SSL certificate and key files

In Step 2, you specified `server.crt` and `server.key` as the certificate file and private key file respectively. Those two files must be created before we can proceed. This step assumes that OpenSSL is installed on your server.

Generate a server key file:

```
openssl genrsa -des3 -out server.key 2048
```

You will be asked to provide a password. Make sure that the password is strong because it will form the one real entry point into the SSL encryption set-up. **Make a note of the password because you'll need it when starting Apache HTTP Server later**.

If you don't wish to specify a password, don't use the `-des3` option in the command above.

Generate a certificate request file (`server.csr`):

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

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

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

The above command generates a self-signed certificate that is valid for one year. You can use the certificate signing request to purchase a certificate from a certificate authority. For testing purposes though, the self-signed certificate will suffice. Copy the certificate file and private key file to the locations you specified in Step 2.

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

Step 4: Update the base URL to use HTTPS

Open a browser window and log into Stash using an administrator account. Go to the Stash administration area and click **Server settings** (under 'Settings'). Change **Base URL** to use HTTPS, for example, "https://stash.mycompany.com").

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.

Securing Stash behind nginx using SSL

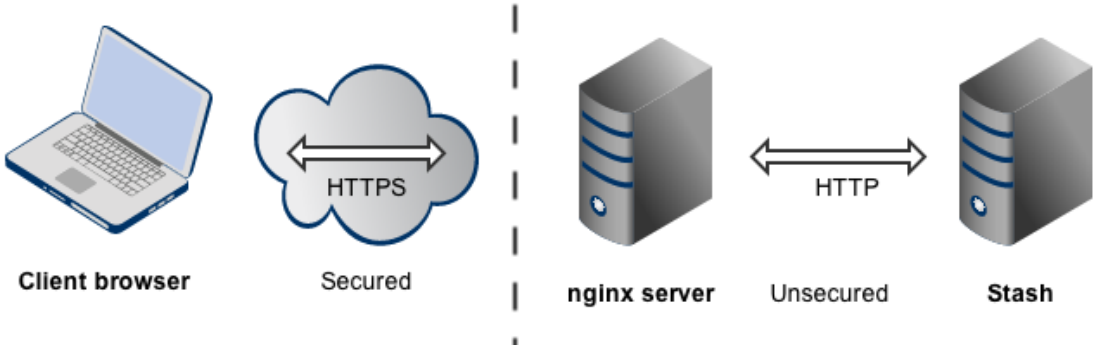
This page describes how to establish a network topology in which the nginx server acts as a reverse proxy for Stash. Typically, such a configuration would be used when Stash is installed in a protected zone 'behind the firewall', and nginx provides a gateway through which users outside the firewall can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

The configuration described on this page results in a scenario where:

- External client connections with nginx are secured using SSL. Connections between nginx and Stash are unsecured.
- Stash and nginx run on the same machine.
- Stash is available at https://mycompany.com:7990/stash.

On this page:

- Step 1: Configure the Tomcat Connector
- Step 2: Set a context path for Stash
- Step 3: Change Stash's base URL
- Step 4: Configure nginx
- Resources



Please note that:

- We assume that you already have a running instance of nginx. If not, refer to the nginx documentation for instructions on downloading and installing nginx.
- SSL certificates must be installed on the server machine.
- Any existing links with other applications will need to be reconfigured using the new URL for Stash.
- Securing Git operations between the user's computer and Stash is a separate consideration - see Enabling SSH access to Git.

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 Tomcat using SSL. Otherwise, if you want to install Stash in an environment that incorporates nginx, this document is for you. (You can of course run

Stash behind nginx without securing client connections to nginx using SSL – we don't describe this option on this page.)

Note that the Atlassian Support Offering does not cover nginx integration. Assistance with nginx may be obtained through the Atlassian community from answers.atlassian.com or from an Atlassian Expert.

Step 1: Configure the Tomcat Connector

Find the normal (non-SSL) `Connector` directive in Tomcat's `<Stash home directory>/shared/server.xml` file, and add the `scheme`, `proxyName`, and `proxyPort` attributes as shown below. Instead of `mycompany.com`, set the `proxyName` attribute to your domain name that the nginx server will be configured to serve. This informs Stash of the domain name and port of the requests that reach it via nginx, and is important to the correct operation of the Stash functions that construct URLs.

```
<Connector port="7990"
     protocol="HTTP/1.1"
     connectionTimeout="20000"
     useBodyEncodingForURI="true"
     redirectPort="443"
     compression="on"

compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,appli
cation/javascript,application/x-javascript"
     secure="true"
     scheme="https"
     proxyName="mycompany.com"
     proxyPort="443" />
```

For more information about configuring the Tomcat Connector, refer to the Apache Tomcat 7.0 HTTP Connector Reference.

Step 2: Set a context path for 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 would be accessed at:

```
http://mycompany.com:7990/
```

For the example configuration on this page, we want Stash to be accessed at:

```
https://mycompany.com:7990/stash
```

In Tomcat's `<Stash home directory>/shared/server.xml` file, set the context path to `/stash`:

```
<Context path="/stash" docBase="${catalina.home}/atlassian-stash"
reloadable="false" useHttpOnly="true">
    ....
</Context>
```

If you use a context path, it is important that the same path is:

- appended to the context path of Stash's base URL (Step 3).
- used when setting up the location for the `proxy_pass` directive (Step 4).

Step 3: Change Stash's base URL

After re-starting Stash, open a browser window and log into Stash using an administrator account. Go to the Stash administration area and click **Server settings** (under 'Settings'), and change **Base URL** to match the proxy URL (the URL that the nginx server will be serving).

For this example, use `http://mycompany.com:7990/stash` (Note the context path included with this.)

Step 4: Configure nginx

Edit `/etc/nginx/nginx.conf`, using the example server configuration below, to configure nginx as a proxy server.

Put the `proxy_pass` directive in the location block, and specify the protocol, name and port of the proxied server in the parameter (in our case, it is http://localhost:7990):

```
server {
 listen          443;
    server_name     mycompany.com;

 ssl                   on;
    ssl_certificate       <path/to/your/certificate>;
    ssl_certificate_key   <path/to/your/certificate/key>;
    ssl_session_timeout   5m;
    ssl_protocols     TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers     HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers   on;

 # Optional optimisation - please refer to
http://nginx.org/en/docs/http/configuring_https_servers.html
 # ssl_session_cache   shared:SSL:10m;
    location /stash {
        proxy_pass    http://localhost:7990/stash;
  proxy_set_header  X-Forwarded-Host $host;
        proxy_set_header  X-Forwarded-Server $host;
  proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_set_header    X-Real-IP $remote_addr;
  proxy_redirect   off;
    }
}
```

Refer to http://nginx.org/en/docs/http/ngx_http_proxy_module.html.

Changes made in the configuration file will not be applied until the command to reload configuration is sent to nginx or it is restarted. To reload the configuration, execute:

```
nginx -s reload
```

This command should be executed under the same user that started nginx.

Resources

You may find the following resources helpful in setting up Stash behind nginx:

- http://nginx.org/en/docs/http/configuring_https_servers.html
- http://www.cyberciti.biz/tips/using-nginx-as-reverse-proxy.html
- https://mywushublog.com/2012/08/atlassian-tools-and-nginx/

Enabling SSH access to Git repositories in Stash

A Stash administrator can enable SSH access to Git repositories in Stash. This allows your Stash users to:

- add their own SSH keys to Stash

- use those SSH keys to secure Git operations between their computer and the Stash server.

Stash users must each add their own SSH key pairs to their Stash account to be able to use SSH access to repositories.

Supported key types are DSA and RSA2. Note that RSA1 is not supported. We've tested key sizes of 768, 1024, 2048, 4096 and 8192 bytes.

On this page:

- Enabling SSH access
- SSH base URL
- When running Stash behind a proxy

Related pages:

- Setting up SSH port forwarding
- Creating SSH keys

Performance
There are performance implications for Stash when using SSH. When users connect to Stash using SSH the encryption of data adds to overall CPU usage. See Scaling Stash for more information.

Security
To implement SSH authentication support, Stash bundles a version of the Apache Mina SSHD server. Stash's SSH server is not integrated with the SSH server on the host Stash is running on nor does it consider the users on the host when authenticating Stash users. To prevent security issues, the embedded SSH server has been locked down to allow execution of a small set of commands for Git hosting. The only commands that are supported are `git upload-pack`, `git receive-pack`, `git archive-pack` and `whoami` (a custom `whoami` implemented in Stash not the `whoami` command that exists on Linux). It is not possible to open an SSH shell using the embedded server to execute arbitrary commands on the server.

Enabling SSH access

To enable SSH access:

1. Go to the Stash administration area and click **Server settings** (under 'Settings').
2. Under 'SSH access', check **SSH enabled**.
3. Enter values for **SSH port** and **SSH base URL**, according the information in the sections below.
4. Click **Save**.

These options will only be available if the "SSH support for Stash" add-on is enabled. For instructions on how to enable this add-on on your instance, please refer to Disabling and enabling add-ons.

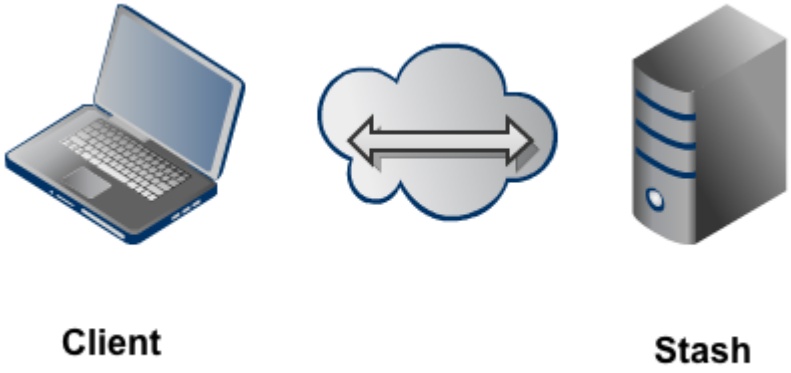
SSH base URL

The **SSH base URL** is the base URL with which users can access the SSH push/pull/clone functionality of Stash.

This is the base URL that Stash will use when displaying SSH URLs to users. If you do not set this, it will default to the host that is set in **Stash base URL**, with the port that SSH is listening on. See Specifying the base URL for Stash.

For example, if the **SSH base URL** is not set and the **Stash base URL** is `https://stash.atlassian.com` and the SSH port is `7999`, the SSH URL for the repository `Jira` in the project `Atlassian` will be `ssh://git@stash.atlassian.com:7999/ATLASSIAN/jira.git`

If you set up port forwarding, you will need to set the **SSH base URL** to the machine and port that is being forwarded to Stash. However, you do not need to specify the port portion of the URL if the default SSH port (port 22) is being forwarded to Stash.



Port forwarding	SSH base URL	Stash base URL	SSH port	Resulting S
✗	Not set	`https://stash.atlassian.com`	`7999`	`ssh://git`
✓ Port 22 -> 7999	`https://stash.atlassian.com`	`https://stash.atlassian.com`	`7999`	`ssh://git`

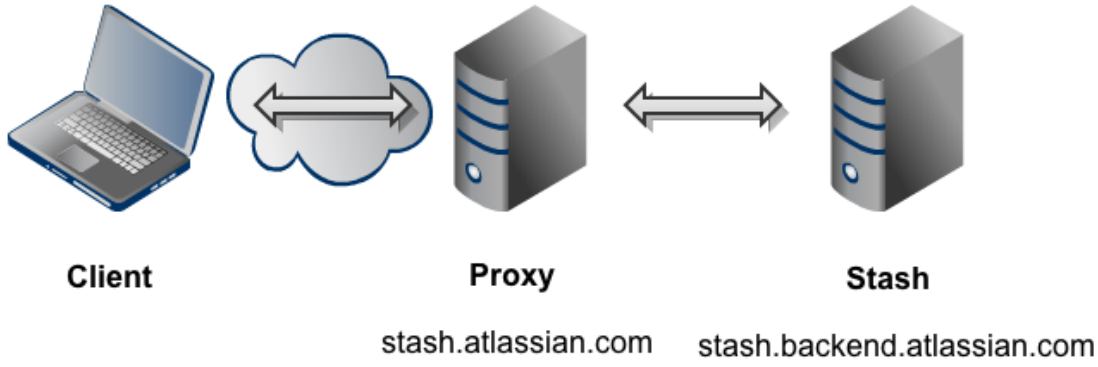
When running Stash behind a proxy

If you run Stash behind a http proxy such as Apache (e.g. as per our instructions), and if Apache runs on a different host, SSH will not be available on that host. Instead, you will need to set the SSH base URL to the machine Stash is actually running on (and the URL should include the SSH port Stash is serving from).

> For example, if the **SSH base URL** is set to `ssh://stash.backend.atlassian.com:7999`, the SSH URL for the repository `Jira` in the project `Atlassian` will be `ssh://git@stash.backend.atlassian.com:7999/ATLASSIAN/jira.git`

If you set up port forwarding, you will need to set the **SSH base URL** to the proxy machine and port that is being forwarded to Stash. However, you do not need to specify the port portion of the URL if the default SSH port (port 22) is being forwarded to Stash.

> For example, if you set up port forwarding from your http proxy host, `stash.atlassian.com`, port 22, to `stash.backend.atlassian.com` port 7999, set the **SSH base URL** to `ssh://stash.atlassian.com`. Then, the SSH URL for the repository `Jira` in the project `Atlassian` will be `ssh://git@stash.atlassian.com/ATLASSIAN/jira.git`



Port forwarding	SSH base URL	SSH port	Stash base URL
✗	`ssh://stash.backend.atlassian.com:7999`	`7999`	`https://stash.backend.atlas`
✓ `Port 22 -> 7999`	`ssh://stash.atlassian.com`	`7999`	`https://stash.backend.atlas`
✓ `Port 44 -> 7999`	`ssh://stash.atlassian.com:44`	`7999`	`https://stash.backend.atlas`

Setting up SSH port forwarding

Why set up port forwarding?

There are two scenarios where you might want to set up port forwarding.

Remove port numbers from your SSH URLs

Stash listens for SSH connections on port 7999 by default.

Your users will need to include the port in the URL they use to clone from Stash, for example:

```
git clone ssh://git@stash.mycompany.com:7999/PROJECT/repo.git
```

Rather than have the port number in the URL, you may wish to set up port forwarding so that connections to the default SSH port are forwarded to the port Stash is listening on (e.g. you could forward port 22 to port 7999).

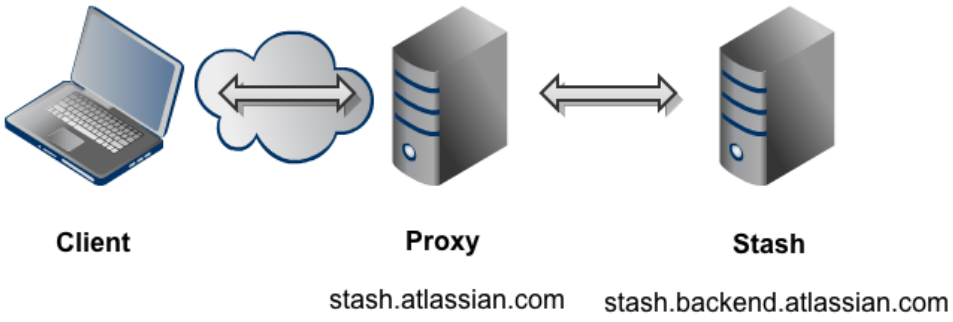
This would allow your users to use a URL without a port number in it, like this:

```
git clone ssh://git@stash.mycompany.com/PROJECT/repo.git
```

Stash is running behind a reverse proxy on a separate machine

You may be following our instructions for setting up Stash behind an Apache front-end.

In this case, your users may not be able to access Stash directly for SSH connections, or if they can, you may wish to make the SSH and HTTPS URLs consistent.



For example, if you have the above topology, without port forwarding (and assuming the default port of 7999), your users will need to clone Stash directly from the backend, like this:

```
git clone ssh://git@stash.backend.atlassian.com:7999/PROJECT/repo.git
```

In your network, the `stash.backend.atlassian.com` machine may not be accessible directly, or you may want the URL to be consistent with the HTTPS URL of `https://stash.atlassian.com/scm/PROJECT/repo.git`.

In this case, you need to set up port forwarding on the `stash.atlassian.com` machine to accept connections and forward them to port 7999 on the `stash.backend.atlassian.com` machine.

How to set up port forwarding

HAProxy

Atlassian recommends the use of HAProxy for forwarding SSH connections through to Stash.

HAProxy is supported on Linux, Solaris and FreeBSD.

> HAProxy is not an Atlassian product, so Atlassian does not guarantee to provide support for its configuration. This section is provided for your information only – use it at your own risk. We recommend that you refer to the HAProxy documentation.

Installing HAProxy

Your Operating System may support installing HAProxy via it's system package manager, such as `apt-get`, `yum` or `rpm`. This will be the easiest way.

Alternatively, you may build HAProxy yourself and install it.

1. Download the latest version of HAProxy from http://haproxy.1wt.eu/#down.
2. Extract the archive and cd into the directory:

```
tar xzvf haproxy-1.4.21.tar.gz
cd haproxy-1.4.21
```

3. Read the instructions in the README for how to build on your system. This is generally quite simple - on a Linux 64 bit 2.6 Kernel, the command is:

```
make TARGET=linux26 ARCH=x86_64
```

4. If it completes successfully, install it following the instructions in the README:

```
sudo make install
```

Configuring HAProxy

HAProxy is extremely powerful - it is designed as a HTTPS load balancer, but also can serve as a port forwarder for ssh.

The full documentation for version 1.4 is here. More documentation is available on the HAProxy web site.

An example simple configuration is as follows:

```
global
       daemon
       maxconn 10000

defaults
       timeout connect 500s
       timeout client 5000s
       timeout server 1h

frontend sshd
       bind *:7999
       default_backend ssh
       timeout client 1h

backend ssh
       mode tcp
       server localhost-stash-ssh 127.0.0.1:7999 check port 7999
```

The above configuration will listen on port 7999 (indicated by the `bind` directive) on all network interfaces. As indicated by the `server` directive, traffic is forwarded to 127.0.0.1, port 7999. You will need to replace `127.0.0.1` with the IP address of the machine running Stash.

You can check your configuration by running:

```
haproxy -f haproxyconf.txt -c
```

To run haproxy, simply start it using

```
haproxy -f haproxyconf.txt
```

If you use HAProxy to additionally proxy HTTP traffic, ensure that the running `mode` configuration is set to `http`:

```
backend http
       mode http
    bind *:80
       server localhost-stash-http 127.0.0.1:7990
```

Using the default SSH port

You can configure HAProxy to listen on the default SSH port instead, so that the port does not need to be

specified in the clone URL.

By default, the normal ssh daemon is running on port 22. You have several options:

- Configure HAProxy to listen on an alternate port as in the previous example.
- Configure multiple network interfaces on the physical machine and force the default ssh daemon to listen on all but the interface for accessing Stash. Configure HAProxy to only listen on that interface.
- Move the default ssh daemon to listen on another port and let HAProxy bind on port 22.

We do not provide instructions on the last two options, except for how to configure HAProxy.

Use the same configuration as the last example, but change the bind port to 22, e.g.

```
...
frontend sshd
       bind *:22
...
```

You will have to run this configuration as the `root` user, using `sudo`, because it specifies a port to listen on that is less than 1024.

```
sudo haproxy -f haproxyconf.txt
```

Configuring the SSH base URL

Once port forwarding is set up, you will need to configure the SSH base URL in Stash so that the clone urls presented in Stash indicate the correct host and port to clone from. See the SSH base URL section in Enabling SSH access to Git repositories in Stash.

Using diff transcoding in Stash

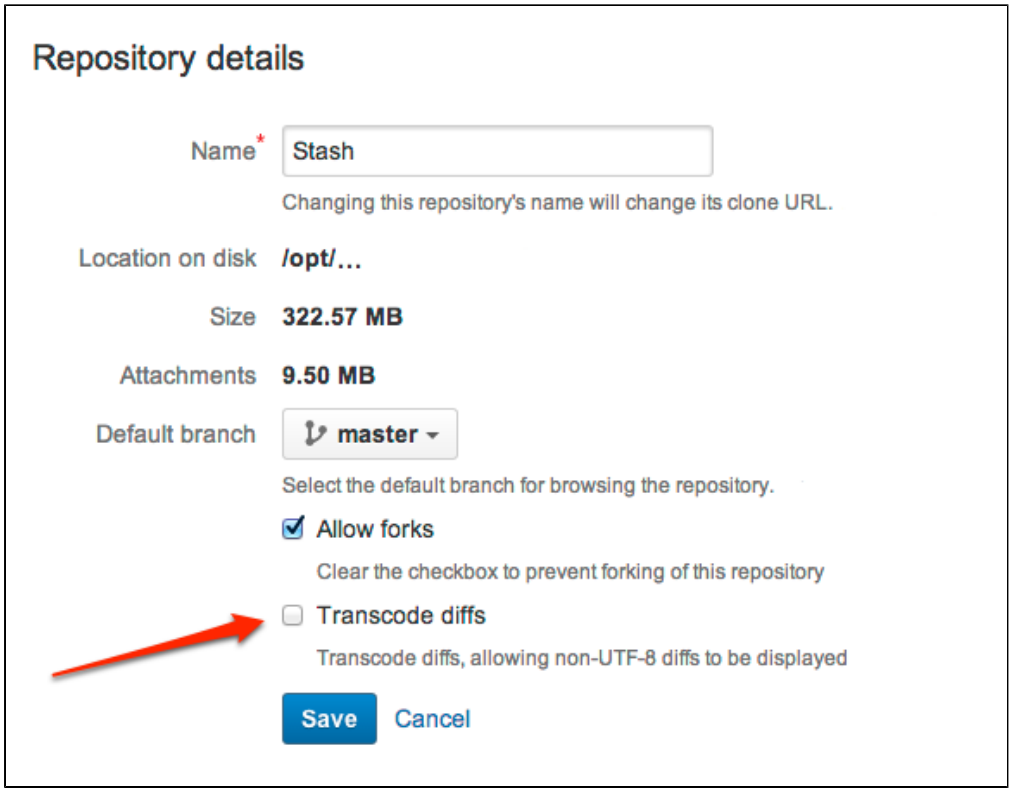
As of Stash 3.1, Stash supports transcoding for diffs. This allows Stash to convert files in encodings like EUC-JP, GB18030 and UTF-16 to UTF-8, so they are processed correctly by `git diff`, which only supports UTF-8. Similar transcoding has been applied to Stash's source view since it was released, so this change brings the diff view in line with the source view. Diff transcoding is applied to commit and pull request diffs, as well as the diff-to-previous view.

> Git for Windows, formerly known as msysgit, has known issues with Unicode paths. Diff transcoding works on all supported versions of Git for Windows, but 1.8.0 or higher is required to support Unicode paths.

Enabling diff transcoding

Diff transcoding must be explicitly enabled for each repository (unlike source view transcoding, which is always performed).

Repository administrators can enable diff transcoding on the repository settings page:



Performance and scaling

There's a performance consideration with transcoding. It is implemented using Git's `textconv` support, so using it adds overhead to displaying diffs. Where possible, the best approach, given `git` only supports UTF-8 content, is to use UTF-8 encoding so that transcoding is not necessary. In repositories without non-UTF-8 content, diff transcoding should be left disabled. Other encodings are often a necessity, however, and for repositories containing such content enabling diff transcoding allows using the full range of Stash features.

Click here to read more...

When transcoding is enabled, `git diff` writes the before and after blobs to temporary files and invokes the `textconv` script once for each file. The script Stash installs uses Perl to send a request back to Stash with the path to each temporary file. Stash then opens each file, detects the encoding using the same algorithm the source view uses, converts the file to UTF-8 and streams it out for `git diff` to use. After `git diff` ha s invoked the `textconv` script the temporary files it created are deleted.

Writing the blobs to disk, starting Perl and calling back into Stash are all overhead processing compared to performing a diff without transcoding. How much overhead that is varies by the size of the diff. When nominally-sized files containing two or three thousand lines or less are being compared the overhead is miniscule, under 50 milliseconds on an average server. However, when comparing larger files the overhead can result in a noticeable delay displaying the diff.

Changing the port that Stash listens on

You may wish to change the port that Stash listens on from the default '7990' to a different value if another

application is already running on that port.

To change the port, edit the `shared/server.xml` file in the `<Stash home directory>`.

Find the following lines in `server.xml`:

```
<Server port="8006" shutdown="SHUTDOWN">
...
<Connector port="7990" protocol="HTTP/1.1"
                connectionTimeout="20000"
                useBodyEncodingForURI="true"
                redirectPort="8443"
                compression="on"

compressableMimeType="text/html,text/xml,text/plain,text/css,application/json"/>
```

You need to modify both the server port (the default is 8006) and the connector port (the default is 7990) to ports that are free on your machine. The server port is required by Tomcat but is not user-facing in any way. The connector port is the port you use to access Stash. For example, in the snippet above, the URL would be http://example.com:7990.

Hint: You can use netstat to identify free ports on your machine. See more information on using netstat on Windows or on Linux.

If you are using a firewall, you should ensure that it is configured to allow HTTP or HTTPS traffic over the connector port you have chosen.

If you are running Stash on a Linux server and want to bind to privileged ports (those below 1024, for example port 80), you will **need to start Stash as root** in order to successfully bind to the port. Alternatively, you can bind Stash to a port over 1024 and then configure iptables to redirect traffic from port 80 to the higher port.

Related pages:

- Specifying the base URL for Stash
- Proxying and securing Stash

Moving Stash to a different context path

There are various reasons why you may wish to change the context path for Stash. Two of those are:

- You are running Stash behind a proxy.
- You have another Atlassian application, or Java web application, available at the same hostname and context path as Stash, and are experiencing login problems (see Login and session conflicts with multiple Atlassian applications).

Related pages:

- Integrating Stash with Apache HTTP Server
- Login and session conflicts with multiple Atlassian applications

Upgrade Note
Note that the location of `server.xml` changed in Stash 3.8. See the Stash upgrade guide.

Changing the context path for Stash:

1. Navigate to your Stash home directory.
2. Stop Stash. SeeStarting and stopping Stash.
3. Edit `<Stash home directory>/shared/server.xml` and find the element below:

```
<Context path="" docBase="${catalina.home}/atlassian-stash" reloadable="false"
useHttpOnly="true"/>
```

Update the `path` attribute to reflect the context path that you want Stash to be accessible at, e.g. "/stash":

```
<Context path="/stash" docBase="${catalina.home}/atlassian-stash"
reloadable="false" useHttpOnly="true"/>
```

Then save the file.

4. Start Stash. See Starting and stopping Stash.

   Stash should now be available at the same host as before under the new context path. For example a server that was at http://localhost:7990 will now be reachable at http://localhost:7990/stash.

5. Once Stash has started, go to the administration area and click **Server settings** (under 'Settings'). Append the new context path to your base URL:

   ```
   https://my-stash-hostname:7990/stash
   ```

6. Click **Save**.

Stash + Apache
Note that if you are running Stash behind Apache:

- You will need to make sure that the host or context path that Stash is exposed on is not also being used by another web application that is listening on a different port.
- If you have updated the Stash context path using the steps outlined above, you will need to update your Apache configuration, as described in Integrating Stash with Apache HTTP Server.

Application Links
If you had Application Links set up before changing the context path in Stash, you will have to recreate those using the new Stash URL. See Linking Stash with JIRA.

SSH
The context path does not affect the URL at which SSH operations occur. After changing the context path so that Stash is accessible at `https://my-stash-hostname:7990/stash`, SSH operations occur without the context path at `ssh://my-stash-hostname:7999`.

Running Stash with a dedicated user

For production installations, we recommend that you create a new dedicated user that will run Stash on your system. This user:

- Should be local.
- Should *not* have admin privileges.
- Should be a non-privileged user with read, write and execute access (called "Full control" permission on Windows) on the Stash install directory and home directory.

Note that, on Windows, running Stash (whether as a service, or not) as a user that is part of the Administrator group can cause Windows to spend a lot of time running permission checks, with a consequent performance impairment for Git operations.

See also Running Stash as a Windows service and Running Stash as a Linux service.

For **Linux**, here is an example of how to create a dedicated user to run Stash:

```
$ sudo /usr/sbin/useradd --create-home --home-dir /opt/atlassian/stash --shell
/bin/bash atlstash
```

Stash debug logging

On this page:

- Debug logging for the Stash server
  - Enabling debug logging via the UI
  - Enabling debug logging on startup
  - Enabling debug logging at runtime
- Profiling logging for the Stash server
  - Enabling profiling logging via the UI
- Debug logging for Git operations on the client
  - On Linux
  - On Windows
- Debug logging for the Stash Backup Client

Debug logging for the Stash server

This section describes how to enable debug level logging in Stash. Stash logs can be found in `<Stash home directory>/log`.

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 via the UI

To enable debug logging, go to the Stash admin area, choose **Logging and Profiling** (under 'Support') and select **Enable debug logging**.

Enabling debug logging on startup

To enable debug logging whenever Stash is started, edit the `<Stash home directory>/shared/stash-config.properties` file (if this file doesn't exist then you should create it) and add the following two lines:

```
logging.logger.ROOT=DEBUG
logging.logger.com.atlassian.stash=DEBUG
```

If your Stash instance is earlier than version 3.2, the `stash-config.properties` file is at the top level of the Stash home directory.

Enabling debug logging at runtime

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

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

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

Profiling logging for the Stash server

This section describes how to enable profiling in Stash. This log is essential when troubleshooting performance issues. Stash logs can be found in `<Stash home directory>/log`.

When using the standard Stash distribution, logs for the Tomcat webserver that hosts Stash can be found in `<Stash installation directory>/log`.

Enabling profiling logging via the UI

To turn on detailed trace information, go to the Stash admin area, choose **Logging and Profiling** (under 'Support') and select **Enable profiling**.

Debug logging for Git operations on the client

Atlassian Support might request DEBUG logs for Git operations (on the client) when troubleshooting issues. You can enable DEBUG logging on the Git client by setting the following variables. If you are using HTTP/S please do remove the `Authorization` header from the output as it will contain your Basic-Auth information. Atlassian provides a set of scripts that simplify the collection of git client debug information.

On Linux

Execute the following in the command line before executing the Git command:

```
export GIT_TRACE_PACKET=1
export GIT_TRACE=1
export GIT_CURL_VERBOSE=1
```

On Windows

Execute the following in the command line before executing the Git command:

```
set GIT_TRACE_PACKET=1
set GIT_TRACE=1
set GIT_CURL_VERBOSE=1
```

Setting `GIT_CURL_VERBOSE` is only useful for connections over HTTP/S since SSH doesn't use the `libcurl` library.

Debug logging for the Stash Backup Client

Atlassian Support might request DEBUG logs for the Backup client when troubleshooting issues.

You can enable DEBUG logging on the Backup client by adding a file named `logback.xml` to your working directory (`pwd`) with the following content:

logback.xml

```
<included><logger name="com.atlassian.stash" level="DEBUG"/></included>
```

Data recovery and backups

This page provides an overview of the backup and restore strategies that Atlassian recommends for use with Stash:

- Stash backup essentials
- Two ways to back up Stash
- How Stash backup and restore works

Questions? Check out FAQ - Data recovery and backup.

Related pages:

- Connecting Stash to an external database
- Supported platforms

Stash backup essentials

An effective backup strategy is essential:

- for avoiding data loss in the event of any system breakdown
- for restoring Stash after any system breakdown
- as part of the Stash upgrade process.

We highly recommend that you establish a data recovery plan that is aligned with your company's policies. At the very least, you should consider these aspects:

- How frequently should Stash be backed up? We recommend that backups are made daily.
- How much downtime is acceptable?
- How long should backups be stored for? We recommend that backups be kept for at least one month.
- Where should the backups be stored? We recommend that backups are stored offsite.

With any strategy, you should schedule the backup window so as to minimise the impact on Stash availability. You might consider checking the access logs to determine patterns of lowest usage to help with this.

Two ways to back up Stash

When it comes to backing up, every organization has slightly different policies and requirements. Some organizations will want a backup solution that just works with minimal intervention, is independent of the underlying database and file system configuration, and don't mind a little downtime when the backup is run as part of a nightly maintenance schedule. Other organizations will have more specific policies and requirements surrounding the use of vendor-specific database and storage backup tools, the maximum acceptable downtime, the format of backups, and where the backups are ultimately stored.

To cater for these different policies and requirements, Stash provides two different backup strategies:

- the Stash Backup Client,
- Stash DIY Backup.

The features of each backup strategy are summarized in the following table:

	Stash Backup Client	Stash DIY Backup
Audience	Recommended for most people without specific backup policies and requirements.	Recommended for developers and system administrators who wish to minimize downtime and/or customize the Stash back up process for their specific database and file system configuration.
Usage	Ready to use out of the box.	Requires you to write some code (in your preferred language) to perform the backup steps.
Downtime	Locks Stash for the entire duration of the back up.	Only locks Stash for the minimum time necessary.
Backup	Backup files are in a vendor-independent format and do not depend on the underlying database configuration.	Backup files rely on vendor-specific database and/or storage tools; for example, `pg_dump` is used if your back end database is PostgreSQL.
Destination	Backups are saved on the local filesystem.	Backups can be saved anywhere.
Supports Stash Server	Yes	Yes
Supports Stash Data Center	Not supported with two or more cluster nodes running.	Yes
Documentation	Using the Stash Backup Client	Using Stash DIY Backup

> Note that the Stash Backup Client does not support Stash Data Center with two or more cluster nodes running. In order to back up an instance of Stash Data Center, you must either:
>
> - bring all nodes in the cluster down except one before running the Stash Backup Client, or
> - switch to Stash DIY Backup.

How Stash backup and restore works

Whether you choose Stash Backup Client or Stash DIY Backup, it is important to understand that there is a tight coupling between the Stash file system on disk and the database that Stash uses. The following types of data are stored in the backup process:

- The Stash **home directory** on the file system, containing your repository data, cache and log files (see Stash home directory for more detail).
- The Stash **database**, containing data about pull requests (pointers to branches in the repos, comments and pull request diffs) and user management.

The backup process must ensure that you keep the repository data and the database perfectly synchronised, by:

- Shutting down Stash before performing the backup.
- Restoring the database and file system at the same time.
- Using the same version or snapshot of the database and file system.

Any backup strategy that captures both the file system and database while Stash is still available to users would run the risk that the backed up Git repositories might be corrupted or that the data in the database doesn't reflect the repository state on disk. Therefore, strategies for backing up and restoring Stash data must keep the repository data and the database perfectly synchronised.

Backing up Stash when using the internal database

When Stash uses the built-in HSQL database, the database files are stored in the Stash file system. See Stash home directory for more detail.

Making a backup of Stash involves copying the Stash home directory.

Note that Atlassian does not recommend using the internal database for a production instance.

Backing up Stash when using an external database

When Stash uses an external database, both the Stash home directory and the external database must be backed up.

If you use the Stash Backup Client, the external database is automatically included as part of the backup in a vendor-independent format.

If you use Stash DIY Backup, you have full control over the database backup tools and procedures, and can use your database vendor's specific backup tooling that is optimized for your database back end and stores the dump in a vendor-specific format.

Restoring Stash from a cold backup

Whether you use the Stash Backup Client or Stash DIY Backup, recovering a Stash instance from backup requires restoring both:

1. the file system backup, and
2. the database backup.

If you use the Stash Backup Client, these components are both automatically included when you run the Stash Restore Client.

If you use Stash DIY Backup, the backup script restores the file system backup, and you need to use your database vendor's specific restore tooling to restore the database backup.

Using the Stash Backup Client

This page describes using the Stash Backup Client, which is the backup strategy that Atlassian recommends for most people running Stash Server instances. This tool can be used to backup data from Stash Server instances from release 2.7.0 and later.

> The Stash Backup Client is not compatible with Stash Data Center instances running two or more cluster nodes. To back up an instance of Stash Data Center, you must either:
>
> - bring down all the cluster nodes except one, or
> - switch to Stash DIY Backup instead of the Stash Backup Client.

For information about other backup strategies for Stash, see Data recovery and backups.

With any strategy, you should consider scheduling the backup window so as to minimise the impact on Stash availability. You might consider checking the access logs to determine patterns of lowest usage to help with this.

We highly recommend that you establish a data recovery plan that is aligned with your company's policies.

Questions? Check out FAQ - Data recovery and backup.

Download the Stash Backup Client from the Atlassian Marketplace, or from here:

Unzip the client into a directory on the Stash server.

On this page:

- How it works
- What is backed up
- Backing up Stash using the client
- Cancelling the client backup
- Restoring Stash to use the existing DB
- Restoring Stash to use a newly created DB
- Debug logging

Related pages:

- Data recovery and backups
- Using Stash DIY Backup
- Scheduling tasks on Linux
- Scheduling tasks on Windows
- Debug logging for the Stash Backup Client
- Stash - FAQ - Data recovery and backups

How it works

The Backup Client implements a common and universal way to back up a Stash instance, and does the following:

1. Locks access to the Stash application, the repositories managed by Stash and the Stash database for the entire duration of the back up. This state is called 'maintenance mode'.
2. Checks that all Git and database operations have completed.
3. Performs an application-specific backup of the Stash home directory and the Stash database. The backup is generic and does not depend on the server or database configuration.
4. Stores the backup as a single tar file on the local filesystem in the specified location.
5. Unlocks Stash from maintenance mode.

You will get an error message if you try to access the Stash web interface, or use the Stash hosting services, when Stash is in maintenance mode.

The client supports Windows and Linux platforms, and Stash versions 2.7 and higher, but does not provide ways to integrate with your organizations IT policies or processes.

As an indication of the unavailability time that can be expected when using the Stash Backup Client, in our testing and internal use we have seen downtimes for Stash of 7–8 minutes with repositories totalling 6 GB in size. For comparison, using Stash DIY Backup for the same repositories typically results in a downtime of less than a minute.

What is backed up

The Backup Client backs up all the following data:

- the database Stash is connected to (either the internal or external DB)
- managed Git repositories
- the Stash audit logs
- installed plugins and their data

The backup does NOT include the following files and directories:

- `export/*`
- `log/*` (except for the audit logs)
- `data/db*` (HSQL data in the DB is backed up, but the files on disk are not)
- `tmp`
- the `plugins` directory (except for the `installed-plugins` directory)

Backing up Stash using the client

The Backup Client must be run from somewhere with access to the Stash home directory. Usually, you will run the Backup Client directly on the Stash server. Run the client with the following commands:

```
cd <path/to/backup-config.properties file>
java -jar <path/to/stash-backup-client.jar>
```

Configuration options are kept in the `backup-config.properties` file, an example of which is included with the client. This file is automatically read from the directory you were in when the stash-backup-client is run. The properties are fully documented in the `backup-config.properties` file, but include:

`stash.home`	Defines the location of the home directory of the Stash instance you wish to back up or restore to. REQUIRED If omitted here it will be taken from the `STASH_HOME` environment variable or the Java system property of the same name if supplied to the Backup and Restore Client on the command line. As a required value, backup and restore will fail if it is not supplied through one of these mechanisms.
`stash.user`	Defines the username of the Stash user with administrative privileges you wish to perform the backup. REQUIRED If omitted here it will be taken from the Java system property of the same name if supplied to the Backup Client on the command line. As a required value, backup will fail if it is not supplied through one of these mechanisms.
`stash.password`	Defines the password of the Stash user with administrative privileges you wish to perform the backup. REQUIRED If omitted here it will be taken from the Java system property of the same name if supplied to the Backup Client on the command line. As a required value, backup will fail if it is not supplied through one of these mechanisms.
`stash.baseUrl`	Defines base URL of the Stash instance you wish to back up. REQUIRED E.g. `http://localhost:7990/stash` or `http://stashserver/`. If omitted here it will be taken from the Java system property of the same name if supplied on the command line to the Backup Client. As a required value, backup will fail if it is not supplied through one of these mechanisms.
`backup.home`	Defines where the Backup Client will store its own files, such as backup archives. If not specified, these files are stored beneath the working directory for the Backup Client. Backup files will be stored in a `backup` subdirectory and logs will be stored in a `logs` subdirectory. Note that on Windows, you must use two backslashes between paths. E.g. `C:\\path\\to\\folder` or instead use the forward slash e.g. `C:/path/to/folder`. The location defined by `backup.home` must not be located in the directory defined by `stash.home`. If that is the case, the Backup Client will fail.

Alternatively, these properties can be given on the command-line, when they need to be prefixed with "`-D`", and be placed before the "-jar" parameter. For example:

```
java -Dstash.password="admin" -Dstash.user="admin"
-Dstash.baseUrl="http://localhost:7990" -Dstash.home=path/to/stash/home
-Dbackup.home=path/to/backup-home -jar stash-backup-client.jar
```

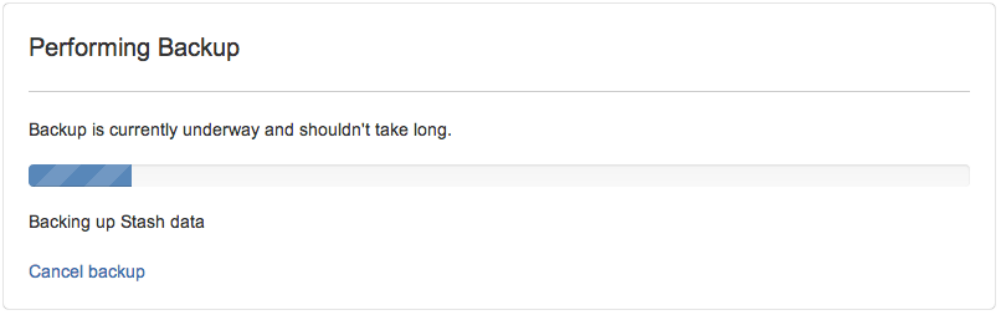
Cancelling the client backup

You can cancel the running client backup operation if necessary.

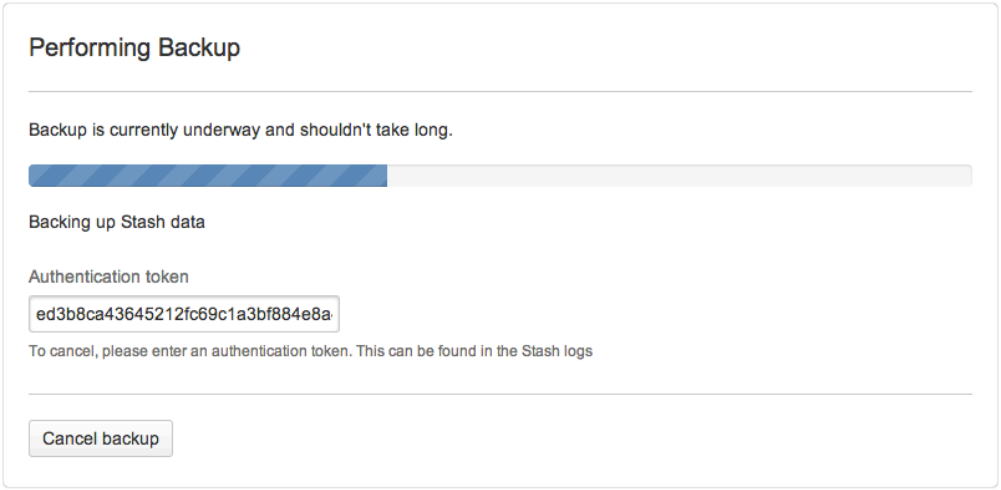
To cancel the backup:

1. Copy the cancel token echoed by the client in the terminal (or the Command Prompt on Windows):

2. Go to the Stash interface in your browser. Stash will display this screen:



3. Click **Cancel backup**, and enter the cancel token:



4. Click **Cancel backup**.

Restoring Stash to use the existing DB

This section applies if you are restoring Stash to fix a corrupted installation, but are able to use the existing DB that Stash was backed up from. This scenario assumes that Stash is to be restored to the same server from which Stash was originally backed up.

The Restore Client must be run on the machine that Stash should be restored to. To ensure restores do not accidentally delete existing data, the Restore Client will only restore into an empty home directory and an empty

database.

The Restore Client will use the JDBC connection configuration contained in the backup you are restoring from.

Follow this process:

1. Stop your Stash instance.
2. Delete the content of the current home directory, so that it is empty.
3. Drop the existing tables in your database so it is empty. The database still needs to exist with the same user/password, and it should have the configuration described in the 'Create the Stash database' section of the relevant page here:
    - MySQL
    - Oracle
    - PostgreSQL
    - SQL Server
4. Run the Restore Client using the following command (replacing '`path/to/stash/home`' and '`/path/to/original/backup/file`' with your own values):

    ```
    java -Dstash.home="path/to/stash/home" -jar stash-restore-client.jar
    /path/to/original/backup/file
    ```

5. If you are restoring Stash to fix a corrupted installation, now follow Steps 4 to 6 of the Stash upgrade guide. Note that you should use the same version of Stash that was used to back up Stash.

Restoring Stash to use a newly created DB

This section applies if you intend to perform a restore into a newly created DB. This scenario assumes the restore is done to a server different from the one from which Stash was originally backed up.

The restore process is database agnostic, meaning the database you are restoring your backup to could be of a different configuration or type from the originally backed up database.

When restoring Stash, the Restore Client must be run on the machine that Stash should be restored to. To ensure restores do not accidentally delete existing data, the Restore Client will only restore into an empty home directory and an empty database.

Follow this process:

1. Create a new empty home directory using the user account that will be used to run Stash.
2. Create the database. It should have the configuration described in the ' Create the Stash database' secti on of the relevant page here:
    - MySQL
    - Oracle
    - PostgreSQL
    - SQL Server
3. Run the Restore Client. See the following section.
4. Follow Steps 4 to 6 of the Stash upgrade guide. Note that you should use the same version of Stash that was used to back it up.

Running the Restore Client from the command line

You can run the Restore Client from the command line. In this scenario, as you will have created a new database, you need to specify the JDBC connection parameters that should be used.

The Restore Client uses the JDBC connection configuration specified in the `jdbc.driver`, `jdbc.url`, `jdbc.user` and `jdbc.password` parameters used in the command to run the Restore Client (see below). Once the database backup is successfully restored, the client will write the specified parameters to the `stash-config.properties` file in the newly restored Stash home directory, allowing the new instance to connect to the restored database once the steps outlined below are followed.

In this example, you can follow the restore into a newly created PostgreSQL database:

```
java -Djdbc.override=true -Djdbc.driver=org.postgresql.Driver
-Djdbc.url=jdbc:postgresql://HOSTNAME:PORT/DATABASE -Djdbc.user=stashuser
-Djdbc.password=password -Dstash.home="path/to/stash/home" -jar
/path/to/stash-restore-client.jar /path/to/original/backup/file
```

Alternatively, you can configure these parameters on the `backup-config.properties` file – make sure the file exists in the current working directory. A sample file is shipped with the client. The properties are fully documented in the `backup-config.properties` file and more details are described below:

`stash.home`	The full path to a directory that the Restore Client will populate with the Stash home data. This directory must be empty. On Windows, you must use two backslashes (`\\`) or a single forward slash (`/`) to separate paths.
`jdbc.override`	By default, the Restore Client will restore into the same database that was backed up. If `jdbc.override` is set to `true`, the Restore Client will restore into the database specified by the `jdbc` properties in the table below. The database must be empty.
`jdbc.driver`	The driver class that Stash should use to log in to the new database. See examples below.
`jdbc.url`	The connection details for the new database, formatted as a JDBC URL. See examples below.
`jdbc.user`	The username that Stash should use to log in to the new database.
`jdbc.password`	The username that Stash should use to log in to the new database.
`stash.home.restore.whitelist`	Defines a comma-separated list of files and directories that may be present in the target home *and* shared directories when restoring a backup. Files other than those matching these entries will result in a failure. By default files .snapshot, lost+found, .DS_Store are white listed.

Example use of JDBC properties

Example `jdbc.driver` and `jdbc.url` properties are shown below:

Database	`jdbc.driver`	`jdbc.url`
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://HOSTNAME:PORT/DATABASE?a
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@//HOSTNAME:PORT/SERVIC
PostgreSQL	org.postgresql.Driver	jdbc:postgresql://HOSTNAME:PORT/DATABAS
SQL Server	com.microsoft.sqlserver.jdbc.SQLServerDriver	jdbc:sqlserver://HOSTNAME:PORT;databaseNa

Debug logging

Debug logging can be turned on by adding the following to the `logback.xml` file in the working directory from where you're running the backup client. Create this file if it does not already exist.

logback.xml

```
<included><logger name="com.atlassian.stash" level="DEBUG"/></included>
```

Using Stash DIY Backup

Stash DIY Backup was introduced in Stash 2.12 as an alternative strategy to using the Stash Backup Client. It allows you to:

- significantly reduce the downtime needed to create a consistent backup,
- use the vendor-specific database backup tool appropriate to your back end database, for example:
  - `pg_dump` if your back end database is PostgreSQL, or
  - `sqlcmd` with an appropriate command for differential backup, if your back end database is MS SQL Server,
- use the optimal file system backup tool for your Stash home directory, for example:
  - an LVM snapshot logical volume if your Stash home directory uses LVM,
  - a SAN-based backup if your Stash home directory uses a Storage Area Network, or
  - `rsync`, if available.
- take backups of Stash Data Center instances without having to bring nodes down manually.

The key to reducing Stash downtime to a minimum is the use of optimal, vendor-specific database and file system backup tools. Such tools are generally able to take snapshots (though sometimes in a vendor-specific format) in much less time than the generic, vendor-neutral format used by the Stash Backup Client.

Stash DIY Backup does require you to write some code in a language of your choice to perform the required backup steps, using the REST API available for Stash 2.12.

DIY Backup supports Windows and Linux platforms, and Stash versions 2.12 and higher. DIY Backup supports both Stash Server and Stash Data Center instances equally - any DIY Backup solution that works on one should work on the other without modification.

For information about other backup strategies for Stash, see Data recovery and backups. That page also discusses the tight coupling between the Stash file system on disk and the database that Stash uses.

> Please note that the examples on this page are provided as guidance for developing a DIY Backup solution. As such, the third-party tools described are for example only – you will need to choose the tools that are appropriate to your own specific installation of Stash.
>
> Consult the vendor documentation for the third-party tools you choose – unfortunately, Atlassian can not provide support for those tools.

On this page:

- How it works
- What is backed up
- DIY Backups using Bash scripts
- Running the Bash script
- Restoring a DIY Backup
- Cancelling the backup
- Advanced - writing your own DIY Backup using the REST APIs

Related pages:

- Data recovery and backups
- Using the Stash Backup Client
- Scheduling tasks on Linux
- Scheduling tasks on Windows

Download the worked example scripts from Bitbucket:

This page:

- Describes a complete DIY Backup solution for a PostgreSQL database and local filesystem, using `bash` shell scripts.
- Provides background information about how the Stash REST API can be used for DIY Backups.

You can use this solution directly if your Stash instance has the same or similar configuration, or use this as a starting point to develop your own DIY Backup solution tailored to your hardware configuration.

How it works

When you use DIY Backup instead of the Stash Backup Client, you have complete control over the backup steps, and can implement any custom processes you like in the language of your choice. For example, you can use your database's incremental or fast snapshot tools and/or your file server's specific tools as part of a DIY Backup.

The DIY Backup works in a similar way to the Stash Backup Client and does the following:

1. Prepare the instance for backup. This happens before Stash is locked, so we want to do as much processing as possible here in order to minimize downtime later. For example, we can take an initial snapshot using incremental database and filesystem utilities. These do not have to be 100% consistent as Stash is still running and modifying the database and filesystem. But taking the initial snapshot now may reduce the amount of work done later (while Stash is locked), especially if the amount of data modified between backups is large. The steps include:
    - Taking an initial backup of the database (if it supports progressive/differential backups).
    - Doing an initial `rsync` of the home folder to the backup folder.
2. Tell Stash to initiate the backup. Stash will:
    - Lock the instance.
    - Drain and latch the connections to the database and the filesystem.
    - Wait for the drain/latch step to complete.
3. Once the Stash instance is ready for backup we can start with the actual DIY Backup. This will include steps to:
    - Make a fully consistent backup of the database, using `pg_dump`.
    - Make a fully consistent backup of the filesystem, using `rsync`.
4. Notify the Stash instance once the backup process finishes and unlock it.
5. Archive all files created during the backup into one big archive.

A user will get an error message if they try to access the Stash web interface, or use the Stash hosting services, when Stash is in maintenance mode.

As an indication of the unavailability time that can be expected, in Atlassian's internal use we have seen downtimes for Stash of 7–8 minutes with repositories totalling 6 GB in size when using the Stash Backup Client. For comparison, using Stash DIY Backup for the same repositories typically results in a downtime of less than a minute.

What is backed up

DIY Backup backs up all the same data as the Stash Backup Client:

- the database Stash is connected to (either the internal or external DB)
- managed Git repositories
- the Stash logs
- installed plugins and their data

DIY Backups using Bash scripts

This section presents a complete DIY Backup solution that uses the following tools:

- `bash` - for scripting
- `jq` - an open source command line JSON processor for parsing the REST responses from Stash

- `pg_dump` (or `sqlcmd`) - for backing up a PostgreSQL database
- `rsync` - for backing up the filesystem
- `tar` - for making a backup archive

This approach (with small modifications) can be used for running DIY Backups on:

- Linux and Unix
- OSX
- Windows with cygwin (note that cygwin Git is *not* supported by Stash).

Download the scripts from Bitbucket.

> The scripts below are for example only and are **not** kept in sync with the scripts found in the Bitbucket repository.

Bash scripts

These scripts implement one particular solution for performing a DIY Backup:

stash.diy-backup.sh

```
#!/bin/bash

#############################################################
# Configure the settings in this section to suit your Stash installation.

SCRIPT_DIR=$(dirname $0)
# Contains all variables used by the other scripts.
source ${SCRIPT_DIR}/stash.diy-backup.vars.sh
# Contains util functions (bail, info, print).
source ${SCRIPT_DIR}/stash.diy-backup.utils.sh
# Contains functions that perform lock/unlock and backup of a Stash instance.
source ${SCRIPT_DIR}/stash.diy-backup.common.sh

# The following scripts contain functions which are dependant on the
configuration of this Stash instance.
# Generally every each of them exports certain functions, which can be
implemented in different ways.

# Exports the following functions:
#     stash_prepare_db     - for making a backup of the DB if differential
backups a possible. Can be empty.
#     stash_backup_db      - for making a backup of the Stash DB.
source ${SCRIPT_DIR}/stash.diy-backup.postgresql.sh

# Exports the following functions:
#     stash_prepare_home   - for preparing the filesystem for the backup.
#     stash_backup_home    - for making the actual filesystem backup.
source ${SCRIPT_DIR}/stash.diy-backup.rsync.sh

# Exports the following functions:
#     stash_backup_archive - for archiving the backup folder and puting the
archive in archive folder.
source ${SCRIPT_DIR}/stash.diy-backup.tar.sh

#############################################################
# This section does NOT need any configuration.
# The actual back up process. It has the following steps:

# Prepare the database and the filesystem for taking a backup.
stash_prepare_db
stash_prepare_home

# Locking the Stash instance, starting an external backup and waiting for
instance readiness.
stash_lock
stash_backup_start
stash_backup_wait

# Backing up the database and reporting 50% progress.
stash_backup_db
stash_backup_progress 50

# Backing up the filesystem and reporting 100% progress.
stash_backup_home
stash_backup_progress 100

# Unlocking the Stash instance.
stash_unlock

# Making an archive for this backup.
stash_backup_archive
#############################################################
```

The `stash.diy-backup.sh` script above references the following scripts:

stash.diy-backup.vars.sh

stash.diy-backup.vars.sh

```
#!/bin/bash

# Used by the scripts for verbose logging. If not true only errors will be
shown.
STASH_VERBOSE_BACKUP=TRUE

# The base url used to access this Stash instance.
STASH_URL=

# The username and password for the user used to make backups (and have this
permission).
STASH_BACKUP_USER=
STASH_BACKUP_PASS=

# The name of the database used by this instance.
STASH_DB=stash

# The path to Stash home folder (with trailing /).
STASH_HOME=

# The path to working folder for the backup.
STASH_BACKUP_ROOT=
STASH_BACKUP_DB=${STASH_BACKUP_ROOT}/stash-db/
STASH_BACKUP_HOME=${STASH_BACKUP_ROOT}/stash-home/

# The path to where the backup archives are stored.
STASH_BACKUP_ARCHIVE_ROOT=
```

stash.diy-backup.utils.sh

stash.diy-backup.utils.sh

```
#!/bin/bash

function error {
    echo "[${STASH_URL}] ERROR:" $*
}

function bail {
    error $*
    exit 99
}

function info {
    if [ "${STASH_VERBOSE_BACKUP}" == "TRUE" ]; then
        echo "[${STASH_URL}]  INFO:" $*
    fi
}

function print {
    if [ "${STASH_VERBOSE_BACKUP}" == "TRUE" ]; then
        echo $*
    fi
}


function check_command {
    type -P $1 &>/dev/null || bail "Unable to find $1, please install it and
run this script again"
}
```

stash.diy-backup.common.sh

stash.diy-backup.common.sh

```
#!/bin/bash

check_command "curl"
check_command "jq"

STASH_HTTP_AUTH="-u ${STASH_BACKUP_USER}:${STASH_BACKUP_PASS}"

function stash_lock {
    STASH_LOCK_RESULT=`curl -s -f ${STASH_HTTP_AUTH} -X POST -H "Content-type:
application/json" "${STASH_URL}/mvc/maintenance/lock"`
    if [ -z "${STASH_LOCK_RESULT}" ]; then
        bail "Locking this Stash instance failed"
    fi

    STASH_LOCK_TOKEN=`echo ${STASH_LOCK_RESULT} | jq -r ".unlockToken"`
    if [ -z "${STASH_LOCK_TOKEN}" ]; then
        bail "Unable to find lock token. Result was '$STASH_LOCK_RESULT'"
    fi

    info "locked with '$STASH_LOCK_TOKEN'"
}

function stash_backup_start {
    STASH_BACKUP_RESULT=`curl -s -f ${STASH_HTTP_AUTH} -X POST -H
"X-Atlassian-Maintenance-Token: ${STASH_LOCK_TOKEN}" -H "Accept:
```

```
application/json" -H "Content-type: application/json"
"${STASH_URL}/mvc/admin/backups?external=true"`
    if [ -z "${STASH_BACKUP_RESULT}" ]; then
        bail "Entering backup mode failed"
    fi

    STASH_BACKUP_TOKEN=`echo ${STASH_BACKUP_RESULT} | jq -r ".cancelToken"`
    if [ -z "${STASH_BACKUP_TOKEN}" ]; then
        bail "Unable to find backup token. Result was
'${STASH_BACKUP_RESULT}'"
    fi

    info "backup started with '${STASH_BACKUP_TOKEN}'"
}

function stash_backup_wait {
    STASH_PROGRESS_DB_STATE="AVAILABLE"
    STASH_PROGRESS_SCM_STATE="AVAILABLE"

    print -n "[${STASH_URL}] .INFO: Waiting for DRAINED state "
    while [ ${STASH_PROGRESS_DB_STATE} != "DRAINED" -a
${STASH_PROGRESS_SCM_STATE} != "DRAINED" ]; do
        print -n "."

        STASH_PROGRESS_RESULT=`curl -s -f ${STASH_HTTP_AUTH} -X GET -H
"X-Atlassian-Maintenance-Token: ${STASH_LOCK_TOKEN}" -H "Accept:
application/json" -H "Content-type: application/json"
"${STASH_URL}/mvc/maintenance"`
        if [ -z "${STASH_PROGRESS_RESULT}" ]; then
            bail "[${STASH_URL}] ERROR: Unable to check for backup progress"
        fi

        STASH_PROGRESS_DB_STATE=`echo ${STASH_PROGRESS_RESULT} | jq -r
'.["db-state"]'`
        STASH_PROGRESS_SCM_STATE=`echo ${STASH_PROGRESS_RESULT} | jq -r
'.["scm-state"]'`
    done

    print "done"
    info "db state '${STASH_PROGRESS_DB_STATE}'"
    info "scm state '${STASH_PROGRESS_SCM_STATE}'"
}

function stash_backup_progress {
    STASH_REPORT_RESULT=`curl -s -f ${STASH_HTTP_AUTH} -X POST -H "Accept:
application/json" -H "Content-type: application/json"
"${STASH_URL}/mvc/admin/backups/progress/client?token=${STASH_LOCK_TOKEN}&perc
entage=$1"`
    if [ $? != 0 ]; then
        bail "Unable to update backup progress"
    fi

    info "Backup progress updated to $1"
}

function stash_unlock {
    STASH_UNLOCK_RESULT=`curl -s -f ${STASH_HTTP_AUTH} -X DELETE -H "Accept:
application/json" -H "Content-type: application/json"
"${STASH_URL}/mvc/maintenance/lock?token=${STASH_LOCK_TOKEN}"`
    if [ $? != 0 ]; then
        bail "Unable to unlock instance with lock ${STASH_LOCK_TOKEN}"
    fi
```

```
    info "Stash instance unlocked"
}
```

stash.diy-backup.postgresql.sh

stash.diy-backup.postgresql.sh

```
#!/bin/bash

check_command "pg_dump"
check_command "psql"
check_command "pg_restore"

function stash_prepare_db {
    info "Prepared backup of DB ${STASH_DB} in ${STASH_BACKUP_DB}"
}

function stash_backup_db {
    rm -r ${STASH_BACKUP_DB}
    pg_dump -Fd ${STASH_DB} -j 5 --no-synchronized-snapshots -f
${STASH_BACKUP_DB}
    if [ $? != 0 ]; then
        bail "Unable to backup ${STASH_DB} to ${STASH_BACKUP_DB}"
    fi
    info "Performed backup of DB ${STASH_DB} in ${STASH_BACKUP_DB}"
}

function stash_bail_if_db_exists {
    psql -d ${STASH_DB} -c '' >/dev/null 2>&1
    if [ $? = 0 ]; then
        bail "Cannot restore over existing database ${STASH_DB}. Try dropdb
${STASH_DB} first."
    fi
}

function stash_restore_db {
    pg_restore -C -Fd -j 5 ${STASH_RESTORE_DB} | psql -q
    if [ $? != 0 ]; then
        bail "Unable to restore ${STASH_RESTORE_DB} to ${STASH_DB}"
    fi
    info "Performed restore of ${STASH_RESTORE_DB} to DB ${STASH_DB}"
}
```

An alternative `stash.diy-backup.mssql.sh` script is included in the downloadable scripts bundle.

stash.diy-backup.rsync.sh

stash.diy-backup.rsync.sh

```
#!/bin/bash

check_command "rsync"
function stash_perform_rsync {
    mkdir -p ${STASH_BACKUP_HOME}
    rsync -avh --delete --delete-excluded --exclude=/caches/
--exclude=/data/db.* --exclude=/export/ --exclude=/log/ --exclude=/plugins/.*/
--exclude=/tmp --exclude=/.lock ${STASH_HOME} ${STASH_BACKUP_HOME}
    if [ $? != 0 ]; then
        bail "Unable to rsynch from ${STASH_HOME} to ${STASH_BACKUP_HOME}"
    fi
}

function stash_prepare_home {
    stash_perform_rsync
    info "Prepared backup of ${STASH_HOME} to ${STASH_BACKUP_HOME}"
}

function stash_backup_home {
    stash_perform_rsync
    info "Performed backup of ${STASH_HOME} to ${STASH_BACKUP_HOME}"
}

function stash_restore_home {
    cp -rf ${STASH_RESTORE_HOME}/`basename ${STASH_HOME}` `dirname
${STASH_HOME}`
    info "Performed restore of ${STASH_RESTORE_ROOT} to ${STASH_HOME}"
}
```

stash.diy-backup.tar.sh

stash.diy-backup.tar.sh

```
#!/bin/bash

check_command "tar"
function stash_backup_archive {
    mkdir -p ${STASH_BACKUP_ARCHIVE_ROOT}
    STASH_BACKUP_ARCHIVE_NAME=`perl -we 'use Time::Piece; my $sydTime =
localtime; print "stash-", $sydTime->strftime("%Y%m%d-%H%M%S-"),
substr($sydTime->epoch, -3), ".tar.gz"'`
    tar -czf ${STASH_BACKUP_ARCHIVE_ROOT}/${STASH_BACKUP_ARCHIVE_NAME} -C
${STASH_BACKUP_ROOT} .

 info "Archived ${STASH_BACKUP_ROOT} into
${STASH_BACKUP_ARCHIVE_ROOT}/${STASH_BACKUP_ARCHIVE_NAME}"
}

function stash_restore_archive {
    if [ -f ${STASH_BACKUP_ARCHIVE_NAME} ]; then
        STASH_BACKUP_ARCHIVE_NAME=${STASH_BACKUP_ARCHIVE_NAME}
    else

STASH_BACKUP_ARCHIVE_NAME=${STASH_BACKUP_ARCHIVE_ROOT}/${STASH_BACKUP_ARCHIVE_
NAME}
    fi
    tar -xzf ${STASH_BACKUP_ARCHIVE_NAME} -C ${STASH_RESTORE_ROOT}
    info "Extracted ${STASH_BACKUP_ARCHIVE_ROOT}/${STASH_BACKUP_ARCHIVE_NAME}
into ${STASH_RESTORE_ROOT}"
}
```

stash.diy-restore.sh

stash.diy-restore.sh

```
#!/bin/bash

SCRIPT_DIR=$(dirname $0)
# Contains all variables used by the other scripts.
source ${SCRIPT_DIR}/stash.diy-backup.vars.sh
# Contains util functions (bail, info, print).
source ${SCRIPT_DIR}/stash.diy-backup.utils.sh

# The following scripts contain functions which are dependant on the
configuration of this Stash instance.
# Generally every each of them exports certain functions, which can be
implemented in different ways.

# Exports the following functions:
# stash_restore_db - for restoring the Stash DB.
source ${SCRIPT_DIR}/stash.diy-backup.postgresql.sh

# Exports the following functions:
# stash_restore_home - for restoring the filesystem backup.
source ${SCRIPT_DIR}/stash.diy-backup.rsync.sh

# Exports the following functions:
# stash_restore_archive - for un-archiving the archive folder.
source ${SCRIPT_DIR}/stash.diy-backup.tar.sh

################################################################
```

```
# The actual restore process. It has the following steps:

function available_backups {
    echo "Available backups:"
    ls ${STASH_BACKUP_ARCHIVE_ROOT}
}

if [ $# -lt 1 ]; then
    echo "Usage: $0 <backup-file-name>.tar.gz"
    if [ ! -d ${STASH_BACKUP_ARCHIVE_ROOT} ]; then
        error "${STASH_BACKUP_ARCHIVE_ROOT} does not exist!"
    else
        available_backups
    fi
    exit 99
fi
STASH_BACKUP_ARCHIVE_NAME=$1
if [ ! -f ${STASH_BACKUP_ARCHIVE_ROOT}/${STASH_BACKUP_ARCHIVE_NAME} ]; then
        error "${STASH_BACKUP_ARCHIVE_ROOT}/${STASH_BACKUP_ARCHIVE_NAME} does
not exist!"
        available_backups
        exit 99
fi
stash_bail_if_db_exists
if [ -e ${STASH_HOME} ]; then
        bail "Cannot restore over existing contents of ${STASH_HOME}. Please
rename or delete this first."
fi
STASH_RESTORE_ROOT=`mktemp -d /tmp/stash.diy-restore.XXXXXX`
STASH_RESTORE_DB=${STASH_RESTORE_ROOT}/stash-db
STASH_RESTORE_HOME=${STASH_RESTORE_ROOT}/stash-home

# Extract the archive for this backup.
stash_restore_archive

# Restore the database.
stash_restore_db

# Restore the filesystem.
```

```
stash_restore_home

############################################################
```

Running the Bash script

Once you have downloaded the Bash scripts, you need to customize two files:

- `stash.diy-backup.vars.sh`
- `stash.diy-backup.sh`

for your setup.

For example, if your Stash server is called `stash.example.com`, uses port 7990, and has its home directory in `/stash-home`, here's how you might configure `stash.diy-backup.vars.sh`:

Example usage:

```
#!/bin/bash

# Used by the scripts for verbose logging. If not true only errors
will be shown.
STASH_VERBOSE_BACKUP=TRUE

# The base url used to access this Stash instance.
STASH_URL= http://stash.example.com:7990

# The username and password for the account used to make backups
(and has permission for this).
STASH_BACKUP_USER= admin
STASH_BACKUP_PASS= admin

# The name of the database used by this instance.
STASH_DB= stash

# The path to Stash home folder (with trailing /).
STASH_HOME= /stash-home

# The path to the working folder for the backup.
STASH_BACKUP_ROOT= /stash-backup
STASH_BACKUP_DB=${STASH_BACKUP_ROOT}/stash-db/
STASH_BACKUP_HOME=${STASH_BACKUP_ROOT}/stash-home/


# The path to where the backup archives are stored.
STASH_BACKUP_ARCHIVE_ROOT= /stash-backup-archives
```

The supplied `stash.diy-backup.vars.sh` is written to use PostgreSQL, rsync, and tar by default. But if you want to use different tools, you can also customize the top section of this file to use different tools as follows:

Example usage:

```
# The following scripts contain functions which are dependant on the
configuration of this Stash instance.
# Generally, each of them exports certain functions, which can be
implemented in different ways.


# Exports the following functions:
# stash_prepare_db - for making a backup of the DB if differential
backups are possible. Can be empty.
# stash_backup_db - for making a backup of the Stash DB.

source ${SCRIPT_DIR}/ stash.diy-backup.postgresql.sh

# Exports the following functions: # stash_prepare_home - for
preparing the filesystem for the backup. # stash_backup_home - for
making the actual filesystem backup.

source ${SCRIPT_DIR}/ stash.diy-backup.rsync.sh

# Exports the following functions:
# stash_backup_archive - for archiving the backup folder and moving
the archive to the archive folder.

source ${SCRIPT_DIR}/ stash.diy-backup.tar.sh
```

You also need to create two directories for DIY Backup to work:

1. `${STASH_BACKUP_ROOT}` is a working directory (`/stash-backup` in our example) where copies of Stash's home directory and database dump are built during the DIY Backup process.
2. `${STASH_BACKUP_ARCHIVE_ROOT}` is the directory (`/stash-backup-archives` in our example) where the final backup archives are saved.

The Bash scripts may be run on any host, provided it has:

- read/write access to the above `${STASH_BACKUP_ROOT}` and `${STASH_BACKUP_ARCHIVE_ROOT}` directories,
- read access to the `${STASH_HOME}` directory,
- read access to the database, and
- network access to run `curl` commands on the Stash server.

This is true regardless of whether you have an instance of Stash Server or Stash Data Center. It doesn't matter whether the filesystem access is direct or over NFS, or whether the network access is direct to a Stash node or to a load balancer / reverse proxy.

Once your `stash.diy-backup.vars.sh` is correctly configured, run the backup in a terminal window:

```
$ ./stash.diy-backup.sh
```

The first time you run the backup, `rsync` will do most of the work since the `/stash-backup` working directory is initially empty. This is normal. Fortunately, this script performs one `rsync` before locking Stash, followed by a second `rsync` while Stash is locked. This minimizes downtime.

On second and subsequent backup runs, `/stash-backup` is already populated so the backup process should be faster. The output you can expect to see looks something like this:

```
$ ./stash.diy-backup.sh
[http://stash.example.com:7990/stash]  INFO: Prepared backup of DB stash in /stash-backup/stash-db/
building file list ... done

sent 109 bytes  received 20 bytes  258.00 bytes/sec
total size is 52  speedup is 0.40
[http://stash.example.com:7990/stash]  INFO: Prepared backup of /stash-home to /stash-backup/stash-home/
[http://stash.example.com:7990/stash]  INFO: locked with '82b4fd41f1e015fa9d53ae99cdf328b1be35f0b5'
[http://stash.example.com:7990/stash]  INFO: backup started with '37ee4b61311a9ddc822ecc0ea1a74faf2bfacddf'
[http://stash.example.com:7990/stash] .INFO: Waiting for DRAINED state....done
[http://stash.example.com:7990/stash]  INFO: db state 'DRAINED'
[http://stash.example.com:7990/stash]  INFO: scm state 'DRAINED'
[http://stash.example.com:7990/stash]  INFO: Performed backup of DB stash in /stash-backup/stash-db/
[http://stash.example.com:7990/stash]  INFO: Backup progress updated to 50
building file list ... done

sent 109 bytes  received 20 bytes  258.00 bytes/sec
total size is 52  speedup is 0.40
[http://stash.example.com:7990/stash]  INFO: Performed backup of /stash-home to /stash-backup/stash-home/
[http://stash.example.com:7990/stash]  INFO: Backup progress updated to 100
[http://stash.example.com:7990/stash]  INFO: Stash instance unlocked
[http://stash.example.com:7990/stash]  INFO: Archived /stash-backup into /stash-backup-archives/stash-201403
$
```

Restoring a DIY Backup

When restoring Stash, you must run the `stash.diy-restore.sh` script on the machine that Stash should be restored to. In order to ensure accidental restores do not delete existing data, you should never restore into an existing home directory.

The new database should be configured following the instructions in Connecting Stash to an external database and its sub-page that corresponds to your database type.

To see the available backups in your `${STASH_BACKUP_ARCHIVE_ROOT}` directory, just type:

```
$ ./stash.diy-restore.sh
```

You should see output similar to this:

```
$ ./stash.diy-restore.sh
Usage: ./stash.diy-restore.sh <backup-file-name>.tar.gz
Available backups:
stash-20140320-092743-063.tar.gz stash-20140320-093043-243.tar.gz stash-20140320-135235-955.tar.gz
stash-20140320-092941-181.tar.gz stash-20140320-093333-413.tar.gz stash-20140320-142649-009.tar.gz
$
```

To restore a backup, run `stash.diy-restore.sh` with the file name as the argument:

```
$ ./stash.diy-restore.sh stash-20140320-092743-063.tar.gz
```

You should see output like this:

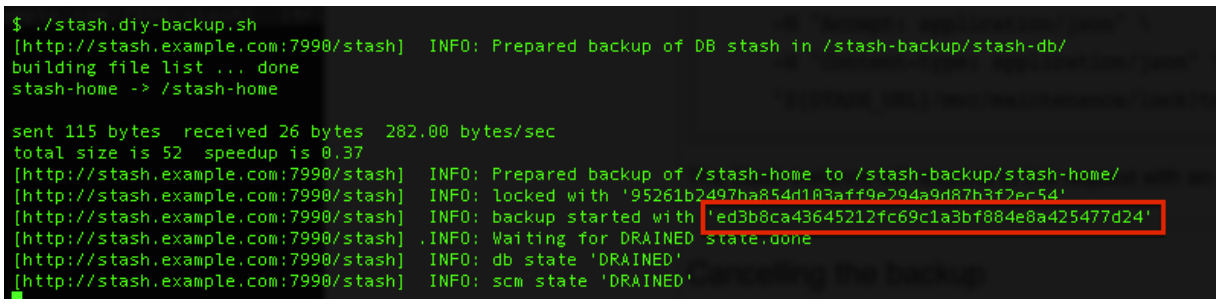
```
$ ./stash.diy-restore.sh stash-20140320-092743-063.tar.gz
[http://stash.example.com:7990/stash]  INFO: Extracted /stash-backup-archives/stash-20140320-092743-063.tar.gz into
[http://stash.example.com:7990/stash]  INFO: Performed restore of /stash-backup/stash-db/ to DB stash
[http://stash.example.com:7990/stash]  INFO: Performed restore of /stash-backup/stash-home/ to /stash-home
$
```

Cancelling the backup

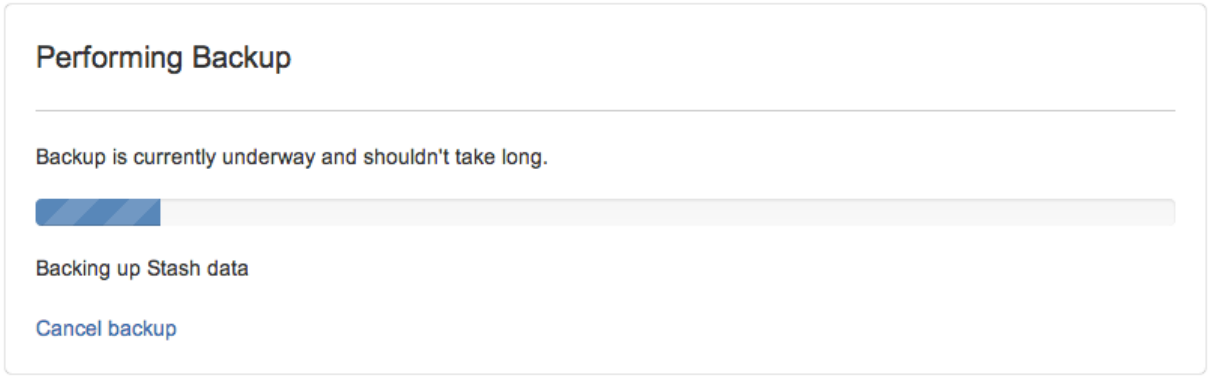
You can cancel the running backup operation if necessary.

To cancel the backup:

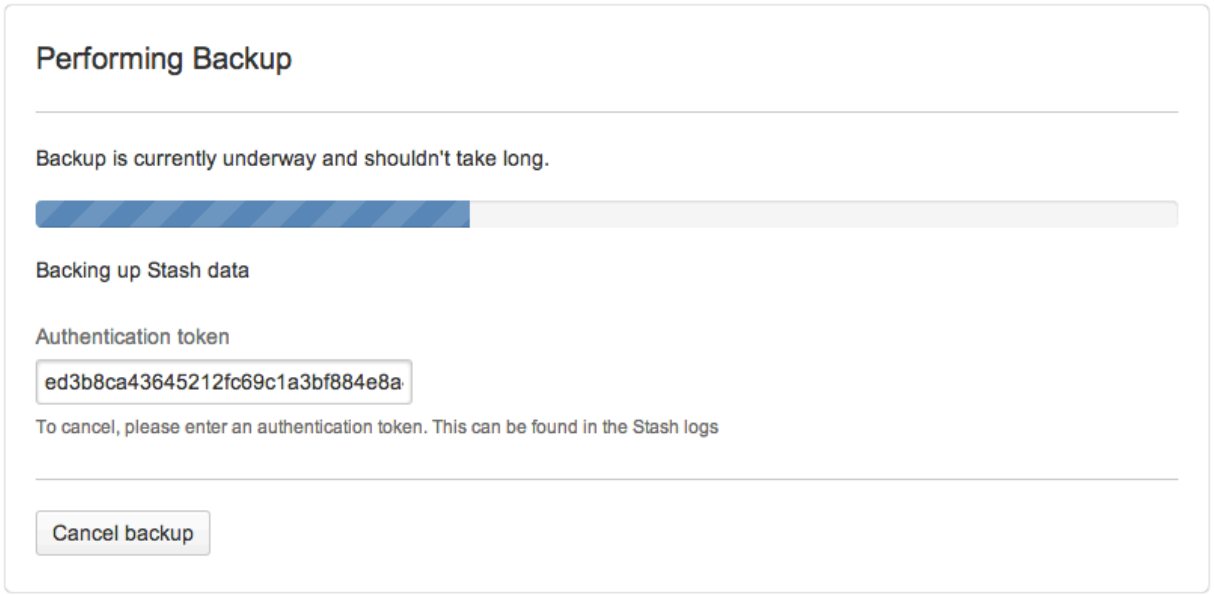
1. Copy the cancel token echoed in the terminal (or the Command Prompt on Windows):



2. Go to the Stash interface in your browser. Stash will display this screen:



3. Click **Cancel backup**, and enter the cancel token:



4. Click **Cancel backup**.

Advanced - writing your own DIY Backup using the REST APIs

> This section is optional and provides background information about how you might use the Stash REST APIs if you need to rewrite the DIY Backup scripts described above in your preferred language or to customize them heavily.

Note that this discussion shows `curl` commands in Bash, however you can use any language.

The following steps are involved:

Preparation

Before you lock Stash you can perform any preparation you like. It makes sense to perform as much processing as possible before you lock Stash, to minimize downtime later. For example, you could perform an `rsync`:

```
rsync -avh --delete --delete-excluded --exclude=/caches/ --exclude=/data/db.*
--exclude=/export/ --exclude=/log/ --exclude=/plugins/.*/ --exclude=/tmp
--exclude=/.lock ${STASH_HOME} ${STASH_BACKUP_HOME}
```

Lock the Stash instance

The next step in making a backup of a Stash instance is to lock the instance for maintenance. This can be done using a POST request to the `/mvc/maintenance/lock` REST point (where `STASH_URL` points to the Stash instance, `STASH_BACKUP_USER` is a Stash user with backup permissions, and `STASH_BACKUP_PASS` is this user's password).

REQUEST

```
curl -s \
  -u
${STASH_BACKUP_USER}:${STASH_BACKU
P_PASS} \
  -X POST \
  -H "Content-type:
application/json" \

"${STASH_URL}/mvc/maintenance/lock
"
```

RESPONSE

```
{
 "unlockToken":"0476adeb6cde3a41aa
0cc19fb394779191f5d306",
 "owner": {
  "displayName":"admin",
  "name":"admin"
 }
}
```

If successful, the Stash instance will respond with a 202 and will return a response JSON similar to the one above. The `unlockToken` should be used in all subsequent requests where `$STASH_LOCK_TOKEN` is required. This token can also be used to manually unlock this Stash instance.

Start the backup process

Next, all connections to both the database and the filesystem must be drained and latched. Your code must handle backing up of *both* the filesystem and the database.

At this point, you should make a `POST` request to `/mvc/admin/backups`. Notice that the `curl` call includes the `?external=true`
parameter (requires Stash 2.12+):

REQUEST

```
curl -s \
  -u
${STASH_BACKUP_USER}:${STASH_BACKU
P_PASS} \
  -X POST \
  -H
"X-Atlassian-Maintenance-Token:
${STASH_LOCK_TOKEN}" \
  -H "Accept: application/json" \
  -H "Content-type:
application/json" \

"${STASH_URL}/mvc/admin/backups?ex
ternal=true"
```

RESPONSE

```
{
 "id":"d2e15c3c2da282b0990e8efb30b
4bffbcbf09e04",
 "progress": {
  "message":"Closing connections
to the current database",
  "percentage":5
 },
 "state":"RUNNING",
 "type":"BACKUP",
 "cancelToken":"d2e15c3c2da282b099
0e8efb30b4bffbcbf09e04"
}
```

If successful the Stash instance will respond with 202 and a response JSON similar to the one above will be returned. The `cancelToken` can be used to manually cancel the back up process.

Wait for the Stash instance to complete preparation

Part of the back up process includes draining and latching the connections to the database and the filesystem. Before continuing with the back up we have to wait for the Stash instance to report that this has been done. To get details on the current status we make a `GET` request to the `/mvc/maintenance` REST point.

REQUEST

```
curl -s \
  -u
${STASH_BACKUP_USER}:${STASH_BACKU
P_PASS} \
  -X GET \
  -H
"X-Atlassian-Maintenance-Token:
${STASH_LOCK_TOKEN}" \
  -H "Accept: application/json" \
  -H "Content-type:
application/json" \
  "${STASH_URL}/mvc/maintenance"
```

RESPONSE

```
 {
 "task":{

"id":"0bb6b2ed52a6a12322e515e88c5d
515d6b6fa95e",
  "progress":{
   "message":"Backing up Stash
home",
   "percentage":10
  },
  "state":"RUNNING",
  "type":"BACKUP"
 },
 "db-state":"DRAINED",
 "scm-state":"DRAINED"
}
```

This causes the Stash instance to report its current state. We have to wait for both `db-state` and `scm-state` to have a status of `DRAINED` before continuing with the back up.

Perform the actual backup

At this point we are ready to create the actual backup of the Stash filesystem. For example, you could use rsync again:

```
rsync -avh --delete --delete-excluded --exclude=/caches/ --exclude=/data/db.*
--exclude=/export/ --exclude=/log/ --exclude=/plugins/.*/ --exclude=/tmp
--exclude=/.lock ${STASH_HOME} ${STASH_BACKUP_HOME}
```

The rsync options shown here are for example only, but indicate how you can include only the required files in the backup process and exclude others. Consult the documentation for `rsync`, or the tool of your choice, for a more detailed description.

When creating the database backup you could use your vendor-specific database backup tool, for example `pg_dump` if you use PostgreSQL:

```
pg_dump -Fd ${STASH_DB} -j 5 --no-synchronized-snapshots -f ${STASH_BACKUP_DB}
```

While performing these operations, good practice is to update the Stash instance with progress on the backup so that it's visible in the UI. This can be done by issuing a `POST` request to `/mvc/admin/backups/progress/client` with the token and the percentage completed as parameters:

REQUEST

```
curl -s \
  -u ${STASH_BACKUP_USER}:${STASH_BACKUP_PASS} \
  -X POST \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \

"${STASH_URL}/mvc/admin/backups/progress/client?token=${STASH_LOCK_TOKEN}&percent
age=${STASH_BACKUP_PERCENTAGE}"
```

The Stash instance will respond to this request with an empty 202 if everything is OK.

When displaying progress to users, Stash divides the 100 percent progress into 90 percent user DIY Backup, and 10 percent Stash preparation. This means, for example, if your script sends `percentage=0`, Stash may display up to 10 percent progress for its own share of the backup work.

Inform the Stash instance that the backup is complete

Once we've finished the backup process we must report to the Stash instance that progress has reached 100 percent. This is done using a similar request to the progress request. We issue a `POST` request to `/mvc/admin/backups/progress/client` with the token and 100 as the percentage:

REQUEST

```
curl -s \
  -u ${STASH_BACKUP_USER}:${STASH_BACKUP_PASS} \
  -X POST \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \

"${STASH_URL}/mvc/admin/backups/progress/client?token=${STASH_LOCK_TOKEN}&percent
age=100"
```

The Stash will respond with an empty 202 if everything is OK. The back up process is considered completed once the percentage is 100. This will unlatch the database and the filesystem for this Stash instance.

Unlock the Stash instance

The final step we need to do in the back up process is to unlock the Stash instance. This is done with a `DELETE` request to the `/mvc/maintenance/lock` REST point:

REQUEST

```
curl -s \
  -u ${STASH_BACKUP_USER}:${STASH_BACKUP_PASS} \
  -X DELETE \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \
  "${STASH_URL}/mvc/maintenance/lock?token=${STASH_LOCK_TOKEN}"
```

The Stash instance will respond to this request with an empty 202 if everything is OK, and will unlock access.

Lockout recovery process

This page describes how to recover administrator access for Stash 2.11 and later. For releases prior to that, refer to Restoring the Stash Administrator's Password.

As an administrator, you may find yourself locked out of Stash and unable to log in. This situation can arise when all users are managed externally from Stash, and Stash becomes unable to access those user directories for some reason, including:

- The external user directory server is not accessible (because the network is down, or the directory is down, or the directory has been moved to another IP address).
- Users are managed in JIRA and the Application Link from Stash to JIRA has been accidentally deleted.
- The admin password has been forgotten or lost.
- The admin account is shaded by a remote account in an LDAP or JIRA instance that is connected to Stash but which is unavailable.

The lockout recovery process for Stash is:

1. Edit the `<Stash installation directory>\bin\setenv.sh` file (or `setenv.bat` on Windows) and add the "`-Datlassian.recovery.password=temporarypassword`" argument to the `JVM_SUPPORT_RECOMMENDED_ARGS` property. The property value must be non-blank, and should look like this when you've done that:

   ```
   #
   # Occasionally Atlassian Support may recommend that you set some specific JVM arguments.
   # below to do that.
   #
   JVM_SUPPORT_RECOMMENDED_ARGS="-Datlassian.recovery.password=temporarypassword"
   ```

   Here we are using "`temporarypassword`", but you should use your own value.

2. Start Stash.
3. Log in to Stash using the 'recovery_admin' username and the temporary password specified in Step 1.
4. Repair your Stash configuration. We strongly recommend that you do not perform other actions while Stash is in recovery mode.
5. Confirm your ability to log in with your usual admin profile.
6. Shut down Stash, remove the `atlassian.recovery.password` argument from `setenv.sh`, and restart Stash as usual.

.

Scaling Stash

This page discusses performance and hardware considerations when using Stash Server.

Note that Stash Data Center, not discussed on this page, uses a cluster of Stash nodes to provide Active/Active failover, and is the deployment option of choice for larger enterprises that require high availability and performance at scale.

Hardware requirements

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

- The number and frequency of clone operations. Cloning a repository is one of the most demanding operations. One major source of clone operations is continuous integration. When your CI builds involve multiple parallel stages, Stash will be asked to perform multiple clones concurrently, putting significant load on your system.
- The size of your repositories – there are many operations in Stash that require more memory and more CPUs when working with very large repositories. Furthermore, huge Git repositories (larger than a few GBs) are likely to impact the performance of the Git client – see this discussion.
- The number of users.

The following are rough guidelines for choosing your hardware:

- Estimate the number of concurrent clones that are expected to happen regularly (look at continuous integration). Add one CPU for every 2 concurrent clone operations. Note that enabling the SCM Cache Plugin (bundled with Stash from version 2.5.0) can help to reduce the cloning load on the Stash server due to CI polling. See Scaling Stash for Continuous Integration performance.
- Estimate or calculate the average repository size and allocate 1.5 x number of concurrent clone operations x min(repository size, 700MB) of memory.

On this page:

- Hardware requirements
- Understanding Stash's resource usage
  - Memory
  - CPU
  - Clones examined
- Configuring Stash scaling options and system properties
- Database requirements

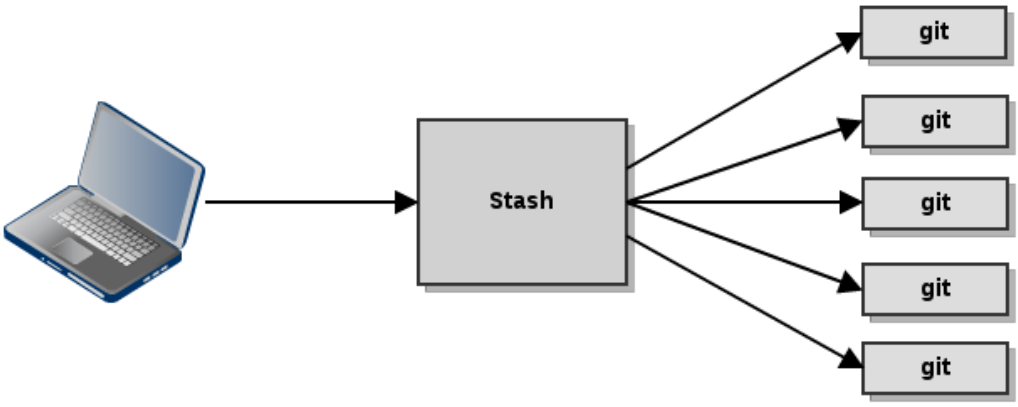
Related pages:

- Using Stash in the enterprise
- Resources for migrating to Git
- Stash production server data
- Scaling Stash for Continuous Integration performance
- Potential performance impact of embedded Crowd directory ordering
- Stash config properties

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 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.

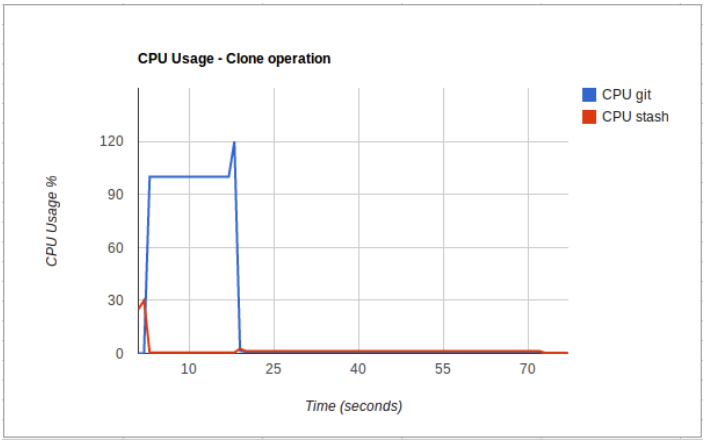
Encryption (either SSH or HTTPS) will have a significant CPU overhead if enabled. As for which of SSH or HTTPS is to be preferred, there's no clear winner, each has advantages and disadvantages as described in the following table.

	HTTP	HTTPS	SSH
Encryption	No CPU overhead for encryption, but plaintext transfer and basic auth may be unacceptable for security.	Encryption has CPU overhead, but thi s can be offloaded to a separate proxy server (if the secure connection is terminated there).	Encryption has CPU overhead.

Authentication	Authentication is slower – it requires remote authentication with the LDAP or Crowd server.	Authentication is much faster – it only requires a simple lookup.
Cloning	Cloning a repository is slightly slower – it takes at least 2 and sometimes more requests, each of which needs authentication and permission checks. The extra overhead is small though - usually in the 10-100ms range	Cloning a repository takes only a single request.

Clones examined

Since cloning a repository is the most demanding operation in terms of CPU and memory, it is worthwhile analyzing the clone operation a bit closer. The following graphs show the CPU and memory usage of a clone of a 220 MB repository:

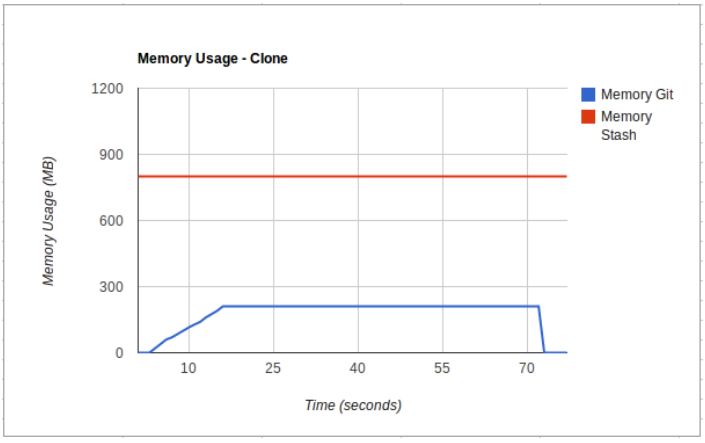


Git process (blue line)

- CPU usage goes up to 100% while the pack file is created on the server side.
- CPU peaks at 120% when the pack file is compressed (multiple CPUs used).
- CPU drops back to 0.5% while the pack file is sent back to the client.

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.

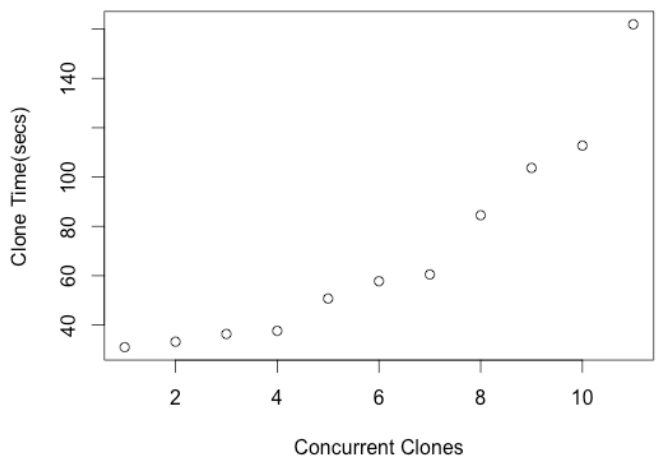


Git process (blue line)

- 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.



This graph shows how concurrency affects average response times for clones:

- Vertical axis: average response times.
- Horizontal axis: number of concurrent clone operations.

The measurements for this graph were done on a 4 CPU server with 12 GB of memory. Response times become exponentially worse as the number of concurrent clone operations exceed the number of CPUs.

Configuring Stash scaling options and system properties

Stash limits the number of Git operations that can be executed concurrently, to prevent the performance for all clients dropping below acceptable levels. These limits can be adjusted – see Stash config properties.

Database requirements

The size of the database required for Stash depends in large part on the number of repositories and the number of commits in those repositories.

A very rough guideline is: 100 + ((total number of commits across all repos) / 2500) MB.

So, for example, for 20 repositories with an average of 25,000 commits each, the database would need 100 + (20 * 25,000 / 2500) = 300MB.

Overview of Stash

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

This page provides an overview of the core strengths of Stash.

On this page:

- Git repository management
- Permissioning granularity - access control
- Enterprise focus
- User management
- Database support
- Supported platforms
- Security
- Code visibility
- Code reviews
- Integration with JIRA
- Integration with Bamboo CI
- Workflow strategies
- Atlassian ecosystem
- Atlassian support

Git repository management

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

will be just observers.

Projects for repository management

Since projects rarely consist of a single repository, Stash provides a convenient **Project structure**. This helps you to organise and manage repositories, and provides the basis on which to manage access to your repositories.

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.

Permissioning granularity - access control

Stash keeps you and your developers productive by providing a way to structure your repositories and manage permissions with a simple, yet powerful, user interface.

Global permissions

Delegate administration of projects to key users and groups, to give your developers the freedom to create and manage repositories

Projects permissions

Use simple permissions at the project level to control access to repositories for users and groups

The new permission screens provide a great overview of who has access to your projects, and managing permissions is even faster.

More...

Repository permissions

Branch permissions

Per-branch "write" permissions for individuals and groups ensure that stable branches remain stable, and development branches foster collaboration. It's a whole new level of Enterprise security.

Some development workflows require that specific developers oversee merges into the master or release branches, while other developers work on bug-fix and feature branches. Branch Permissions let you turn this "gentlemen's agreement" into a seamless, enforceable process, reducing confusion and time wasted backing out changes that were merged prematurely.

Read more about branch permissions in Stash.

Enterprise focus

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.

Enterprise licenses

From growing startups to the Fortune 100, Atlassian Enterprise offers products and services that are built to match the needs of the largest enterprise customers. With the new Enterprise tiers for Stash, Atlassian has you covered no matter how big your organisation is. And of course, just like JIRA and Confluence, the Stash Enterprise licenses come with 24x7 Personalized Phone Support.

User management

With **LDAP**, **Crowd** and **JIRA** support, you can manage a small team easily in Stash's internal directory, or 500 developers in your corporate directory.

Single sign-on with Crowd

Atlassian Crowd allows enterprise teams to integrate and deploy single sign-on (SSO) using popular directory servers such as Active Directory or OpenLDAP. Now your team members need only log in once to any of your Atlassian applications (JIRA, Confluence, Bamboo) to be able to access all of them, without repeatedly typing in their password.

- *IT administrators* can centralize user management through Crowd and provide SSO for all Atlassian apps with minimal configuration.
- *End-users* enjoy the convenience of logging in once to any Atlassian application, avoiding the interruption of repeated authentication across other applications. Log in once, and you're automatically logged in to all applications connected to Crowd, including Stash.

Read more about using SSO with Stash.

Database support

MySQL, PostgreSQL, SQL Server and Oracle support

Stash now has support for all these major databases: MySQL, Oracle, PostgreSQL and Microsoft SQL Server. Choose the database that best fits your needs, or your system administrator is most familiar with, or that your company is already using. *More...*

Database migration

Switch easily from the database embedded in Stash to your organisation's existing technology stack, and migrate painlessly if your system administrators change the infrastructure. Stash scales and adapts as your requirements change. *More...*

Supported platforms

Security

SSH support - authentication

Developed from the ground up with enterprise level security as a #1 priority, Stash now supports SSH in addition to HTTPS. You can either use standard HTTPS authentication, or set up your public keys and connect to Stash using SSH. This resolves Stash's #1 feature request, focused on adding support for SSH security.

Developers can manage their own SSH keys, and add as many as they like. Stash administrators can grant or revoke the SSH keys of any user. *More...*

Code visibility

Code search

File search

Stash's new file search ensures that you can quickly find any file in your repository, without needing to check out the source. Just start typing any part of the file name into the search field and you'll get a list of matches, fast. And you can filter by path, CamelCase (for example, AttrM to match AttributeMap) and file extension.

Image diffs

Stash makes diffs more accessible to everyone on your team, not just the back-end coders.

Have you ever tried to find the subtle difference between two images? That difference may be small like a text change or as large as a page redesign. Web designers, front-end developers, and maybe a few QA folks, rejoice and check out Stash's interactive image diff viewer.

Maybe even more useful is ediffs. When viewing a diff it can sometimes be difficult to distinguish textual changes. Stash solves this with the addition of ediffs so you can clearly see the textual changes added or removed between two revisions.

Code reviews

Pull requests

With Stash and Pull Requests, code reviews become an integral part of your development process. Development happens on branches and when code is ready to be merged into the main branch a Pull Request is opened. Unless the code has been reviewed as part of a Pull Request, it does not get integrated back into the main branch. All the benefits of code review baked right into your workflow!

Creating a pull request is like starting a discussion. Your reviewers can see the changes you have made, comment on those changes and commit further changes and improvements to the branch if required. When everyone agrees, the branch can then be merged back to master or your main development branch.

Getting your code reviewed has never been easier – simply click the **Pull Request** button in the repository header, select the branch you've been working on, the branch you want to merge to, then add a short description and you're done.

Read more about using pull requests in Stash.

Integration with JIRA

Issues

Pull requests

Developers create pull requests when their code is ready for peer review before merging a development branch into the main code line. To make pull requests most effective, reviewers need more context around the changes: What bug or feature is this pull request resolving? What are the details of those issues? Are any of the issues still open?

Pull requests now tightly integrate with JIRA, putting issue details front and center. View the status of an issue, along with it's assignee and description to get the scoop without ever leaving Stash. This allows reviewers to

- Gain contextual awareness into the task which is being worked on by looking at descriptions, comments and attachments
- Quickly review the requirements for a new feature or bugfix
- Click straight through into JIRA, to keep issues up to date for upcoming releases

Read more about pull requests in Stash.

Integration with Bamboo CI

Integration with Bamboo and other CI build tools

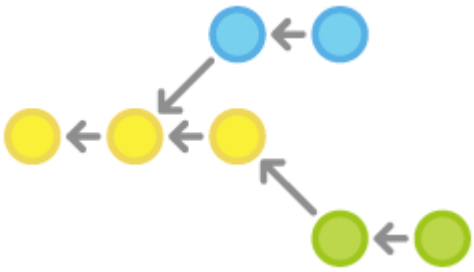
Workflow strategies

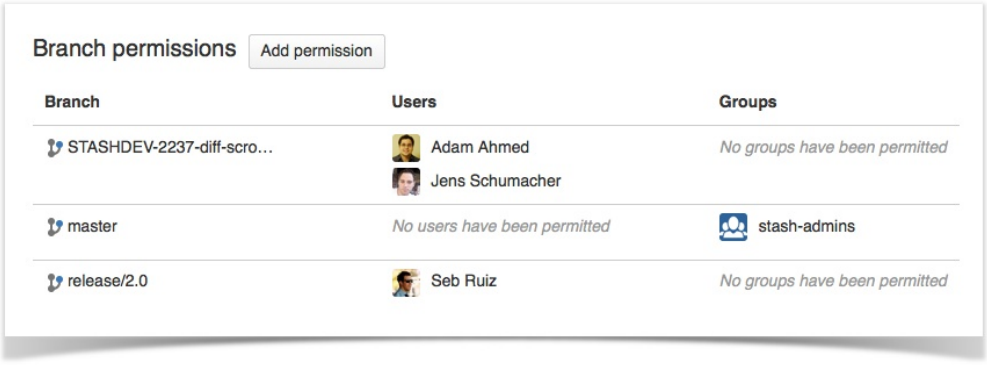
Git repository hooks

Git hooks allow you to customise your development team's workflow for any requirement they may have. Admins can enable and configure hooks for each repository, right within Stash, without having to install them on the file-system.

Stash comes bundled with repository hooks to let you start customising Stash straight off.

Read more about using repository hooks in Stash.





Atlassian ecosystem

Marketplace add-ons

Visualise information about your Git repository, comment on your code in-line, collect achievements when committing code or receive change notifications inHipChat. With almost a dozen add-ons available on the Atlassian Marketplace, you can extend Stash to suit your needs.

Public API for custom integrations

API for hook integrations

We've leveraged native Git hooks to create a new hooks API that allows developers to easily write their own hooks. Stash handles the persistence, packaging and per-repository configuration for your hooks, making it simple to extend Stash to suit your particular project's needs. Read more aboutwriting hooks for Stash in our revamped developer docs.

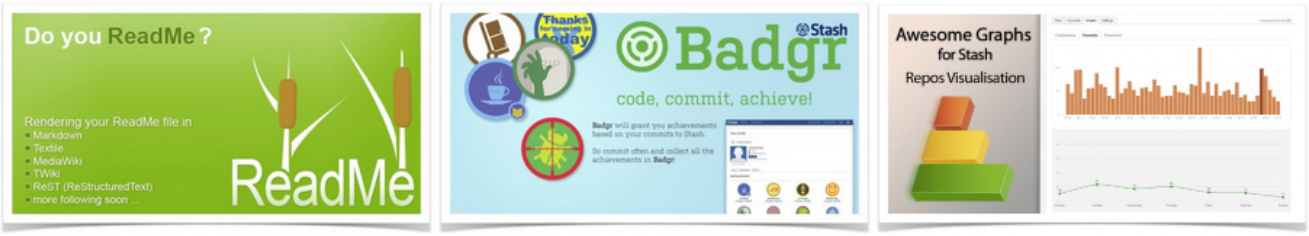
Furthermore, you can mix-and-match bundled Stash hooks with hooks from the Atlassian Marketplace. You can find and install these from within Stash – simply use the **Add hook** button on the hooks settings page to view available hooks from the marketplace.

API for build status

Picture this: you're about to click the 'merge' button, and then you pause to consider: *did these changes pass all their tests?* To answer that, it's important to know what the latest build status of the branch is before the changes

are integrated into master. But don't waste time navigating to your build system and sifting through test data to see whether the changes passed or failed. Stash 2.1 does it automatically. Our new build status API allows build servers, such as Bamboo and Jenkins, to publish build details to the pull request's overview, giving you a quick idea of whether the pull request is good to merge or not.

Read more about the build status API in our developer documentation.



Atlassian support

Scaling Stash for Continuous Integration performance

If you've got CI or other automatic tooling set up to poll Stash for changes, you can end up with high load on your Stash server. Consider for instance a CI server that has a number of builds set up for a given repository. Each of those builds polls Stash for changes and when it detects a change, it starts a new build. If your CI server supports parallel and/or chained build steps, each of these builds typically results in multiple clone operations of the same repository. The result: lots of polling for changes, and bursts of clones of a repository.

Caching

CI results in highly repetitive calls to Stash: polling for changes typically results in the same response 90% of the time and a build triggers a number of identical clone calls to Stash. Both operations can benefit greatly from caching when you experience repetitive load from CI.

Stash 2.5, and later versions, ship with the SCM Cache Plugin already bundled. The plugin is enabled by default, but note that ref advertisement is disabled by default – see the 'Limitations' section below. The plugin is also available from the Atlassian Marketplace.

On this page:

- Limitations
- Considerations
- Enabling and disabling caching
- Configuration
- REST API

Related pages:

- Scaling Stash
- Stash config properties

Limitations

- Caching can be applied to 'ref advertisement' (polling for changes), clone and shallow clone requests only. Fetch/pull operations are not cached, but these operations will still benefit from the 'ref advertisement' cache.
- As a precaution, ref advertisement caching is *disabled* by default. The openness of the plugin system means that plugins (or manually installed git hooks) could be updating refs in repository without the caching plugin detecting these changes. The result would be a stale refs cache and failing clone/fetch operations. However, if you know that there are no plugins or git hooks installed that make changes to the repository directly, you can enable the ref advertisement caching using the system properties listed in the Configuration section below. Note that as an additional precaution, the ref advertisement caches are configured to automatically expire after a minute.

Considerations

Cache data is stored on disk for both ref advertisements and for clone operations for a configurable period of

time. Since large instances can potentially consume an entire disk the SCM Cache Plugin monitors remaining disk space and is automatically disabled when the configured minimum free disk space is reached.

See Stash config properties for descriptions of the available system properties.

Enabling and disabling caching

Enable the SCM Cache Plugin from the admin area in Stash. Click **Manage Add-ons** (under 'Add-Ons') and filter for system add-ons. Click **SCM Cache Plugin for Stash** and then either **Enable** or **Disable**.

Configuration

The SCM Cache Plugin for Stash can be configured using either the REST endpoint (some settings, not all) or the system properties in `<STASH-HOME>/stash-config.properties`. Settings configured through REST are considered *before* the settings in `stash-config.properties`.

REST API

Method	Url	Description
GET	`/rest/scm-cache/latest/config/protocols`	Retrieve the protocols for which caching has been enabled.
PUT	`/rest/scm-cache/latest/config/protocols/{protocol}`	Enable caching of hosting requests for the protocol (HTTP or SSH).
DELETE	`/rest/scm-cache/latest/config/protocols/{protocol}`	Disable caching of hosting requests for the protocol (HTTP or SSH).
GET	`/rest/scm-cache/latest/config/refs/enabled`	Retrieve whether ref advertisement caching is enabled (true) or disabled (false).
PUT	`/rest/scm-cache/latest/config/refs/enabled/{status}`	Enable (status = true) or disable (status = false) ref advertisement caching.
GET	`/rest/scm-cache/latest/config/refs/ttl`	Retrieve the expiry in seconds for the ref advertisement caches.

PUT	`/rest/scm-cache/latest/config/refs/ttl/{expiryInSec}`	Set the expiry in seconds for the ref advertisement caches.
GET	`/rest/scm-cache/latest/config/upload-pack/enabled`	Retrieve whether clone caching is enabled (true) or disabled (false).
PUT	`/rest/scm-cache/latest/config/upload-pack/enabled/{status}`	Enable (status = true) or disable (status = false) clone caching.
GET	`/rest/scm-cache/latest/config/upload-pack/ttl`	Retrieve the expiry in seconds for the clone caches.
PUT	`/rest/scm-cache/latest/config/upload-pack/ttl/{expiryInSec}`	Set the expiry in seconds for the clone caches.
GET	`/rest/scm-cache/latest/caches`	Retrieve information about the current caches: size, number of cache hits and misses, etc.
DELETE	`/rest/scm-cache/latest/caches`	Clear all caches.
GET	`/rest/scm-cache/latest/caches/{projectKey}/{repoSlug}`	Retrieve information about the caches for the repository identified by projectKey and repoSlug: size, number of cache hits and misses, etc.
DELETE	`/rest/scm-cache/latest/caches/{projectKey}/{repoSlug}`	Clear the caches for the repository identified by projectKey and repoSlug.

Stash production server data

This page provides some data around the Stash production instance that we run internally at Atlassian. We're providing this to give some idea of how Stash performs in a production environment. Please realise that this information is entirely specific to this particular instance – the details of your own installation may result in

different performance data.

This data was collected with New Relic in February 2013, when the server was running a pre-release version of Stash 2.2.

On this page:

- Hardware
- Load
- Server load
- Git operations

Hardware

The performance data below was gathered from the Atlassian Stash production server running on:

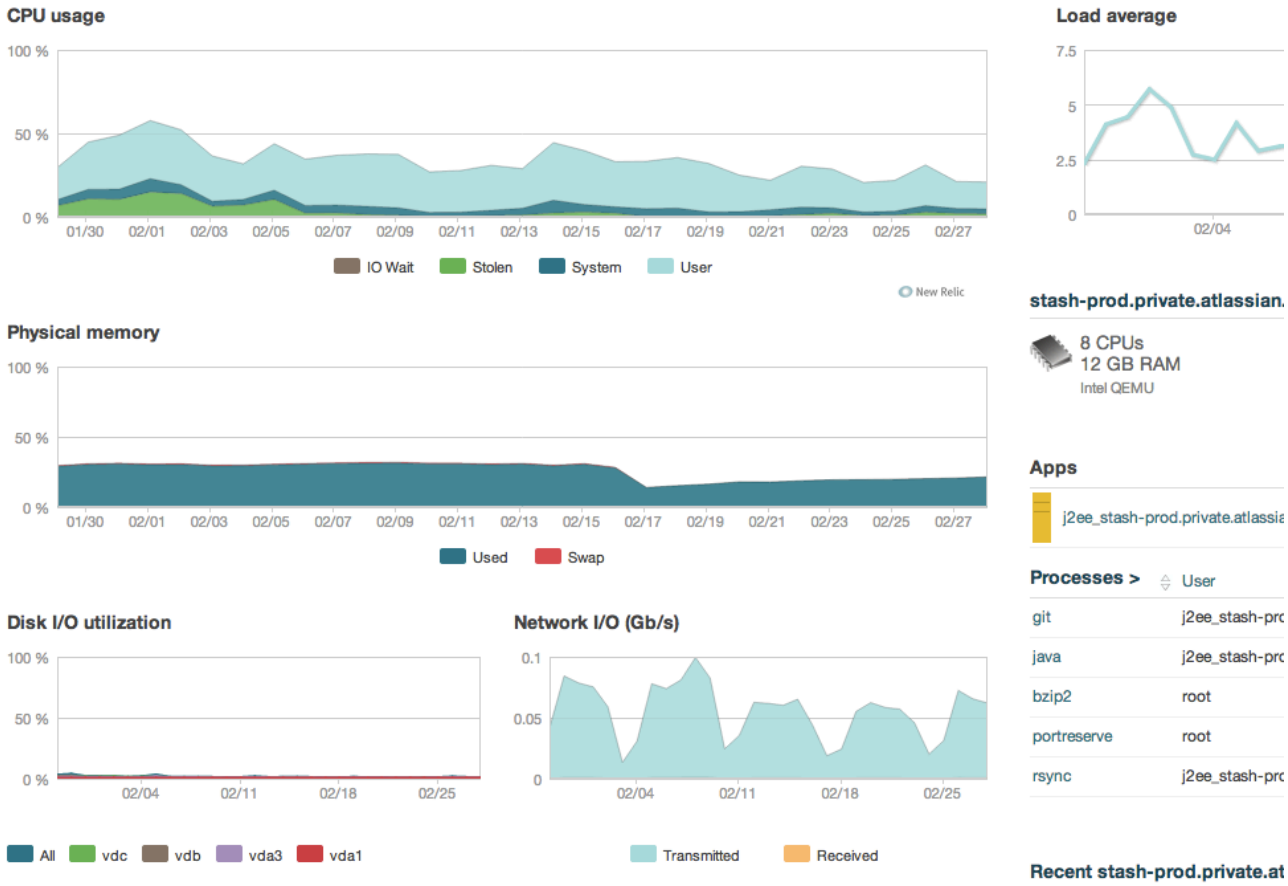
- Virtualised hardware
- 4 Hyper-threaded cores
- 12 GB RAM

Load

Load data summary for February 2013:

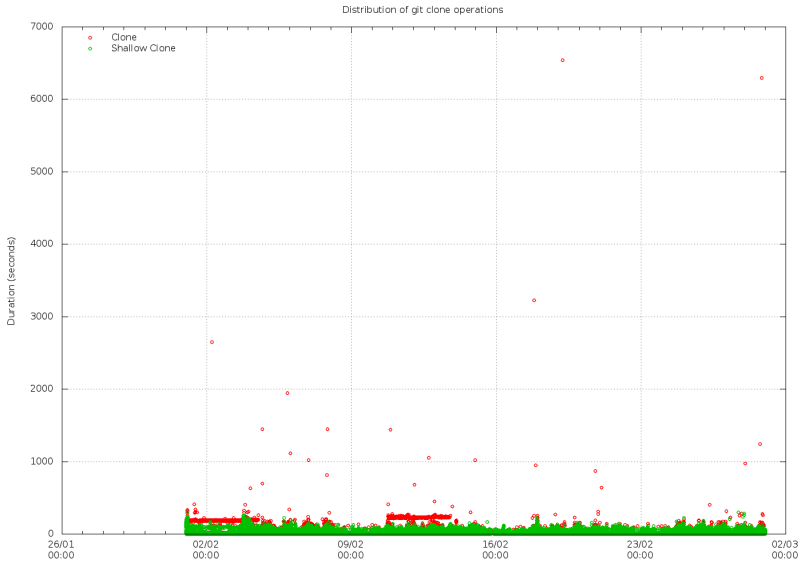
Type	Load
CPU usage	less than **30%** on average
Load average	less than **3** on average
Physical Memory	peaked at **31%**
Processes	Git: **17.3% CPU** Java: **18.8% CPU**
Clones	on average less than **300ms**
Git operations/hour	peaking at **11,000** with an average of about **3,500**
Concurrent connections/hour	peaking at **100** connections with an average of about **40** concurrent connections
CI running against Stash server	**3 build servers** with approximately **300 agents**

Server load

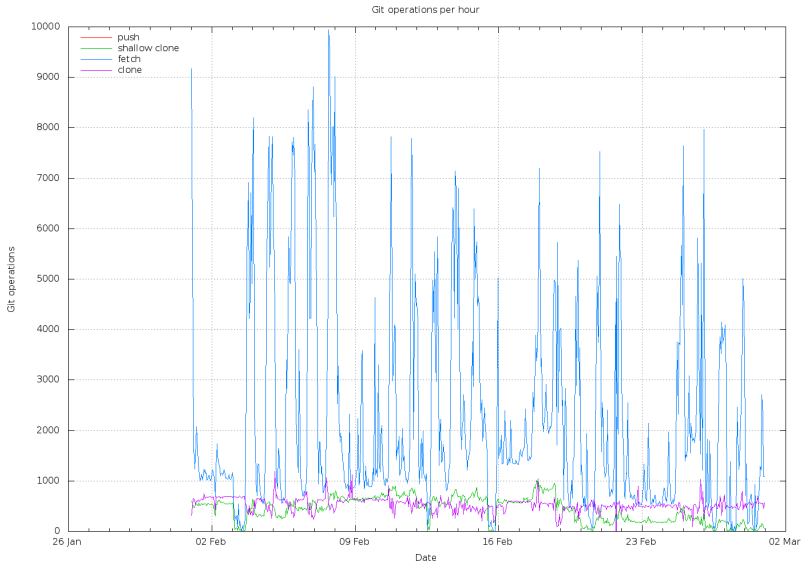


Git operations

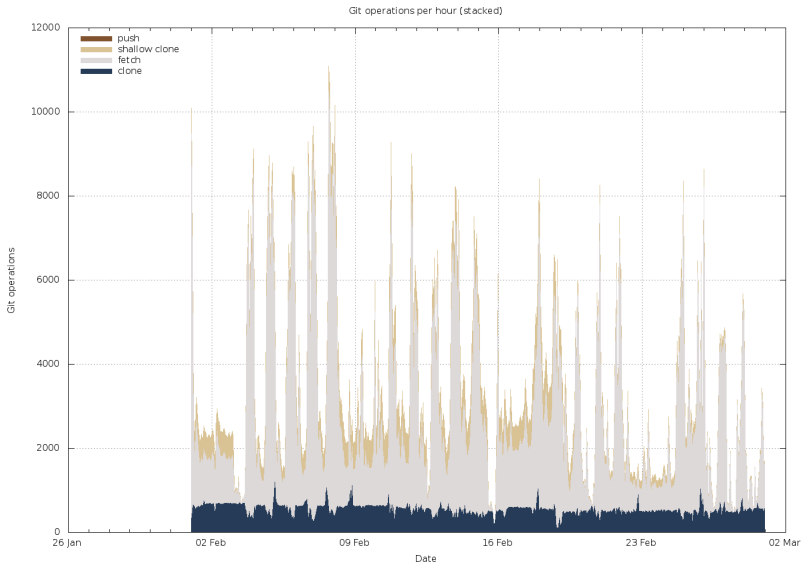
Git clone operations



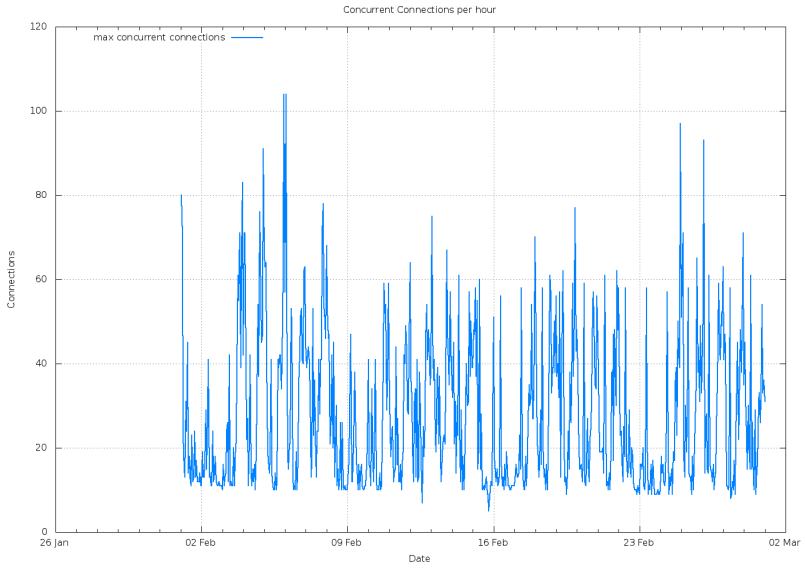
Git operations per hour



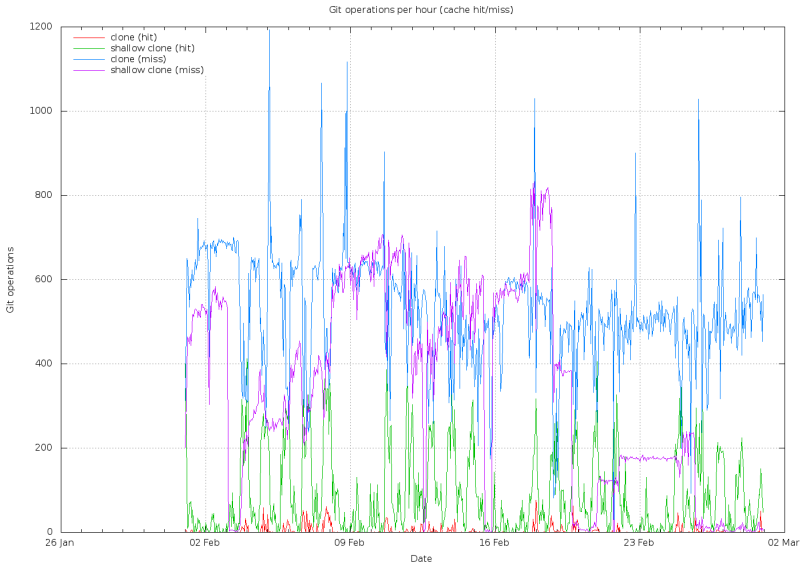
Git operations per hour (stacked)



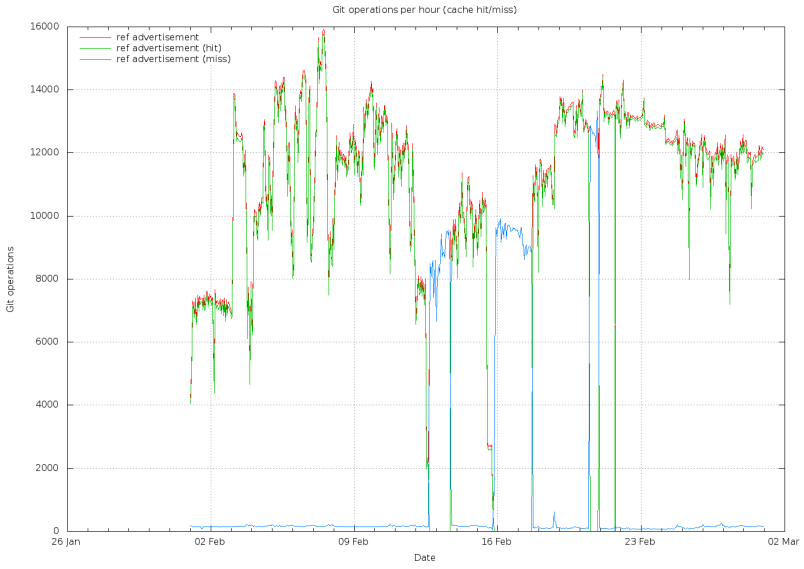
Concurrent connections per hour



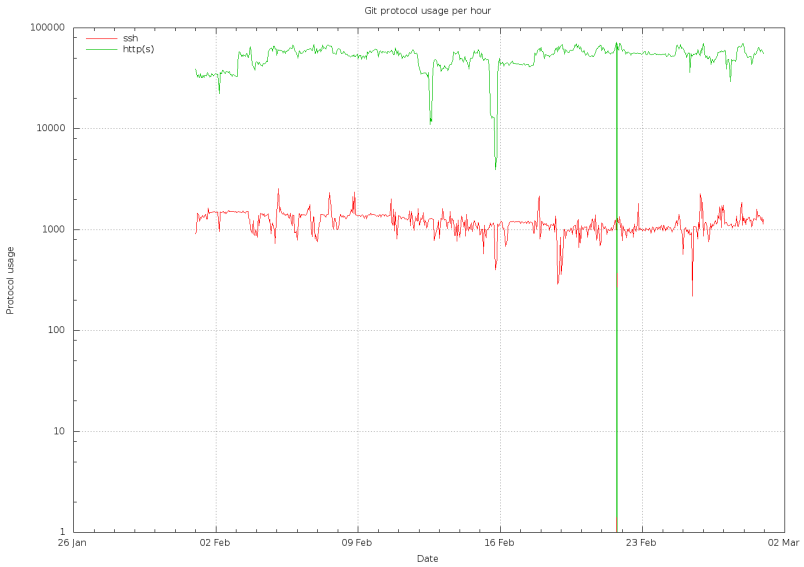
Git operations - cache hit/miss



Git operations - cache hit/miss



Git protocol usage per hour



High availability for Stash

This page...

... describes how to set up a single Stash server in a highly available configuration.

For production installs...

... we highly recommend that you first read Using Stash in the enterprise.

For Active/Active HA with Stash...

... see Stash Data Center inst ead.

If Stash is a critical part of your development workflow, maximizing application availability becomes an important consideration. There are many possible configurations for setting up a HA environment for Stash, depending on the infrastructure components and software (SAN, clustered databases, etc.) you have at your disposal. This guide provides a high-level overview and the background information you need to be able to set up a single Stash Server in a highly available configuration. The guide also describes one possible configuration in more detail.

Note that Atlassian's Stash Data Center product uses a cluster of Stash nodes to provide Active/Active failover. It is the deployment option of choice for larger enterprises that require high availability and performance at scale. Read about Failover for Stash Data Center.

Please note that your feedback and comments are welcome! We very much value additional lessons learned from your experience with alternative scenarios!

- High availability
- Understanding the availability requirements for Stash
- Failover options
- Automatic correction
- Cold standby
- System setup
- Example HA implementation
- Licensing

High availability

High availabilty describes a set of practices aimed at delivering a specific level of "availability" by eliminating and/or mitigating failure through redundancy. Failure can result from unscheduled down-time due to network errors, hardware failures or application failures, but can also result from failed application upgrades. Setting up a highly available system involves:

Proactive Concerns

- Change management (including staging and production instances for change implementation)
- Redundancy of network, application, storage and databases
- Monitoring system(s) for both the network and applications

Reactive Concerns

- Technical failover mechanisms, either automatic or scripted semi-automatic with manual switchover
- Standard Operating Procedure for guided actions during crisis situations

This guide assumes that processes such as change management are already covered and will focus on **redundancy / replication** and **failover procedures**. When it comes to setting up your infrastructure to quickly recover from system or application failure, you have different options. These options vary in the level of uptime they can provide. In general, as the required uptime increases, the complexity of the infrastructure and the knowledge required to administer the environment increases as well (and by extension the cost goes up as well).

Understanding the availability requirements for Stash

Central version control systems such as Subversion, CVS, ClearCase and many others require the central server to be available for any operation that involves the version control system. Committing code, fetching the latest changes from the repository, switching branches or retrieving a diff all require access to the central version control system. If that server goes down, developers are severely limited in what they can do. They can continue coding until they're ready to commit, but then they're blocked.

Git is a distributed version control system and developers have a full clone of the repository on their machines. As a result, most operations that involve the version control system don't require access to the central repository. When Stash is unavailable developers are not blocked to the same extent as with a central version control system.

As a result, the availability requirements for Stash *may* be less strict than the requirements for say Subversion.

Consequences of Stash unavailability	
Unaffected	**Affected**
Developer: • Commit code • Create branch • Switch branches • Diff commits and files • ... • Fetch changes from fellow developers	Developer: • Clone repository • Fetch changes from central repository • Push changes to central repository • Access Stash UI - create/do pull requests, browse code Build server: • Clone repository • Poll for changes Continuous Deployment: • Clone repository

Failover options

High availability and recovery solutions can be categorized as follows:

Failover option	Recovery time	Description	Possible with Stash
Automatic correction / restart	2-10 min (application failure) hours-days (system failure)	• Single node, no secondary server available • Application and server are monitored • Upon failure of production system, automatic restarting is conducted via scripting • Disk or hardware failure may require reprovisioning of the server and restoring application data from a backup	✓
Cold standby	2-10 min	• Secondary server is available • Stash is NOT running on secondary server • Filesystem and (optionally) database data is replicated between the 'active' server and the 'standby' server • All requests are routed to the 'active' server • On failure, Stash is started on the 'standby' server and shut down on the 'active' server. All requests are now routed to the 'standby' server, which becomes 'active'.	✓
Warm standby	0-30 sec	• Secondary service is available • Stash is running on both the 'active' server and the 'standby' server, but all requests are routed to the 'active' server • Filesystem and database data is replicated between the 'active' server and the 'standby' server • All requests are routed to the 'active' server • On failure, all requests are routed to the 'standby' server, which becomes 'active' • ✗ This configuration is currently not supported by Stash, because Stash uses in-memory caches and locking mechanisms. At this time, Stash only supports a single application instance writing to the Stash home directory at a time.	✗
Active/Active	< 5 sec	• Provided by Stash Data Center, using a cluster of Stash nodes and a load balancer. • Stash is running, and serving requests, on all cluster nodes. • Filesystem and database data is shared by all cluster nodes. Clustered databases are not yet supported. • All requests are routed to the load balancer, which distributes requests to the available cluster nodes. If a cluster node goes down, the load balancer immediately detects the failure and automatically directs requests to the other nodes within seconds. • Stash Data Center is the deployment option of choice for larger enterprises that require high availability and performance at scale.	✓

Automatic correction

Before implementing failover solutions for your Stash instance consider evaluating and leveraging automatic correction measures. These can be implemented through a monitoring service that watches your application and performs scripts to start, stop, kill or restart services.

1. A Monitoring Service detects that the system has failed.

2. A correction script attempts to gracefully shut down the failed system.
    a. If the system does not properly shut down after a defined period of time, the correction script kills the process.
3. After it is confirmed that the process is not running anymore, it is started again.
4. If this restart solved the failure, the mechanism ends.
    a. If the correction attempts are not or only partially successful a failover mechanism should be triggered, if one was implemented.

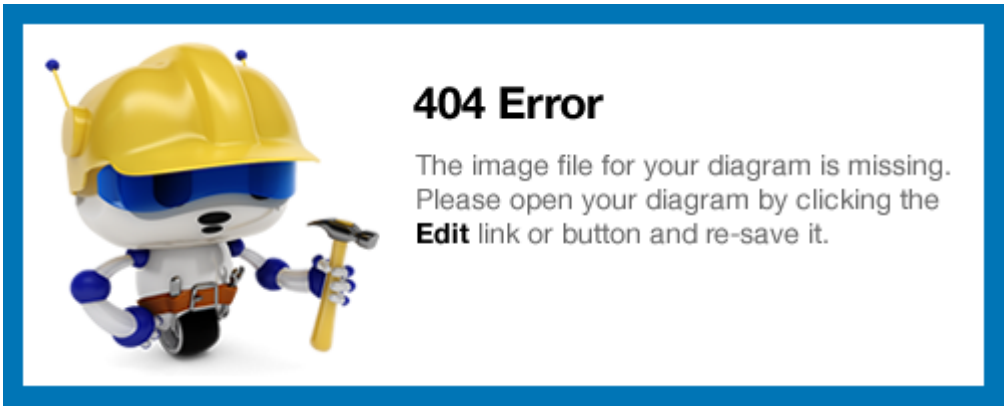
Cold standby

The cold standby (also called Active/Passive) configuration consists of two identical Stash servers, where only one server is ever running at a time. The Stash home directory on each of the servers is either a shared (and preferably highly available) network file system or is replicated from the active to the standby Stash server. When a system failure is detected, Stash is restarted on the active server. If the system failure persists, a failover mechanism is started that shuts down Stash on the active server and starts Stash on the standby server, which is promoted to 'active'. At this time, all requests should be routed to the newly active server.

For each component in the chain of high availability measures, there are various implementation alternatives. Although Atlassian does not recommend any particular technology or product, this guide gives examples and options for each step. In the following, each component in the system is described and an example configuration is used to illustrate the descriptions.

System setup

This section describes one possible configuration for how to set up a single instance of Stash for high availability.



Component	Description
Request Router	Forwards traffic from users to the active Stash instance.
High Availability Manager	Tracks the health of the application servers and decides when to fail over to a standby server and designate it as active. Manages failover mechanisms and sends notifications on system failure.

<table>
<tr><td>Stash server</td><td>Each server hosts an identical Stash installation (identical versions).

Only one server is ever running a Stash instance at any one time (know as the active server). All others are considered as standbys.

The Stash home directory resides on a replicated or shared file system visible to all application servers (described in more detail below).

The Stash home directory must never be modified when the server is in standby mode.</td></tr>
<tr><td>Stash DB</td><td>The production database, which should be highly available. How this is achieved is not explored in this document. See the following database vendor-specific information on the HA options available to you:

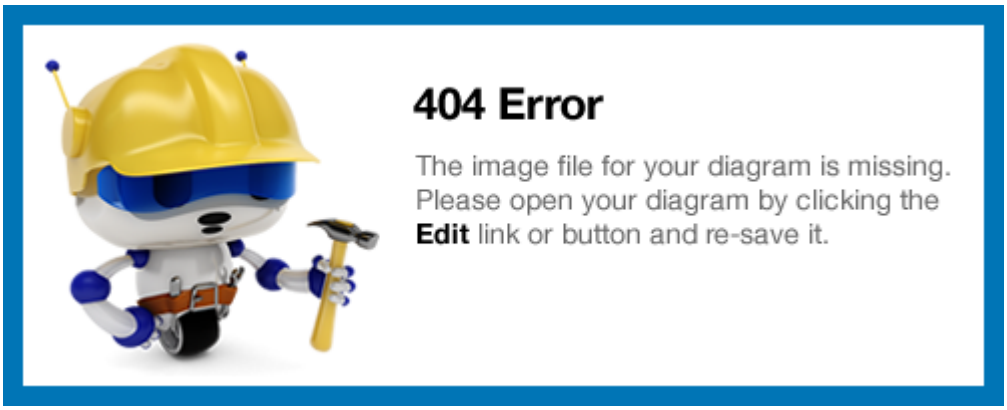
<table><tr><th>Database</th><th>More Information</th></tr><tr><td>Postgres</td><td>http://www.postgresql.org/docs/9.2/static/high-availability.html</td></tr><tr><td>MySQL</td><td>http://dev.mysql.com/doc/refman/5.5/en/ha-overview.html</td></tr><tr><td>Oracle</td><td>http://www.oracle.com/technetwork/database/features/availability/index.html</td></tr><tr><td>SQLServer</td><td>http://technet.microsoft.com/en-us/library/ms190202.aspx</td></tr></table></td></tr>
</table>

Example HA implementation

> This particular implementation is provided to illustrate the concepts, but hasn't been tested in production. We strongly recommend that you devise a solution that best fits your organisation's existing best practices and standards and is thoroughly tested for production readiness.

The example configuration that we'll use to illustrate the concepts consists of a Linux cluster of two nodes. Each node is a CentOS server with Java, Git and Stash installed. Stash's home directory is replicated between the nodes using DRBD, a block-level disk replication mechanism. The cluster is managed by CMAN. Pacemaker, a high availability resource manager, is used to manage two HA resources: Stash and a Virtual IP. Pacemaker runs on each machine, elects the 'primary' node for Stash and starts Stash on this node. The Virtual IP resource is configured to run on the same node as the Stash resource, removing the need for a separate 'request router' component. Pacemaker monitors Stash and when it detects a failure tries to restart Stash on the primary node. If the restart fails, or does not resolve the issues, it fails over to the secondary node. The Virtual IP resource is configured to run on the same node as the Stash resource and will also be moved to the secondary node.

Scripts to create a virtual network based on this example configuration using packer, vagrant and VirtualBox can be found in the stash-ha-example repository. Specifically, the scripts for installing the required software components can be found in the `packer/scripts` directory. The scripts for configuring the cluster can be found in the `vagrant/scripts` directory.



Request router

All high availability solutions are based on redundancy, monitoring and failover. In the cold standby approach, only one server is running Stash at a time. It is the request router's responsibility to route all incoming requests to the node that is currently the primary node. For full high availability, the request router should be highly available itself, meaning that the component is monitored by the HA manager and can be failed over to a redundant copy in the network.

Requirements

- Routes all incoming requests to the node that is currently the 'primary' node
- Should be highly available itself

Options

- HAproxy
- Keepalived
- LVS
- LinuxHA
- various commercial solutions

Solution in example HA implementation

The example HA implementation does not include a separate Request Router server. Instead it includes a virtual IP HA resource that is co-located with the Stash resource. The virtual IP resource is managed by Pacemaker and will be moved to the standby node when the Stash resource fails over to the standby node.

Data replication

Stash stores its data in two places: the Stash home directory and the database that you have configured. The Stash home directory contains, among other things, the Git repositories being managed by Stash (with some additional Stash-specific files and directories), installed plugins, caches and log files. The database contains, among other things, your project and repository information and metadata, pull requests data and the data for your installed plugins.

Data in Stash's home directory and in the database are *very* tightly coupled. For instance, repository pull requests have their metadata, participants and comments stored in the database but certain Git-oriented information around merging and conflicts (which are used to display the diffs in the user interface) are stored in the managed Git repositories. If the two were to fall out of sync you might see an incorrect pull request diff, you might be left unable to merge the pull request, or Stash may simply refuse to display the pull request at all. Similarly, Stash plugins are installed from jar files in the Stash home directory but their state is stored in the database. If the two were to fall out of sync then plugins may malfunction or not appear installed at all, thus degrading your Stash experience.

When designing a high availability solution for Stash based on a replicated file system and database, it's important that the file system replication is atomic. The replicated file system must be a consistent *snapshot* of the 'active' filesystem. This is important because changes to a Git repository happen in predictable ways: first the objects (files, trees and commits) are written to disk, followed by updates of the refs (branches and tags). Some synchronisation tools such as `rsync` perform file-by-file syncing, which can result in an

inconsistent Git repository if the repository is modified while the sync is happening (for example, if object files have not been synced, but the updated refs have been).

Furthermore, the tight coupling between the Stash home directory and database makes it essential that the Stash home directory and database are *always* consistent and in sync (see here for more information). By extension, this means that any high availability solution based on a replicated file system and database needs to ensure that the replicated file system and database are in sync. For example, if the replication is based on hourly synchronisation to a standby node, care must be taken to ensure that the synchronisation of the database and filesystem happen at the same time.

Requirements

- File system replication must replicate a consistent *snapshot* of Stash's home directory.
- The database and the file system must be replicated at the same time.

Options

- DRBD
- LVM
- SAN
- NAS
- various commercial solutions

Solution in example HA implementation

The example HA implementation uses a DRBD managed block device for its Stash home directory. By default, DRBD runs in a Primary/Secondary configuration in which only a single node can mount the DRBD managed volume at a time. In this configuration, DRBD should be managed by Pacemaker to ensure that the DRBD volume is co-located with the Stash resource.

In preparation for experimentation with an Active/Active configuration, the example HA implementation has configured DRBD in a dual-primary configuration, which allows both nodes to mount the DRBD managed volume at the same time.

Monitoring

To allow for monitoring in a high availability environment, Stash, since version 2.10, has supported a REST-based health check endpoint at `/status` that describes the current health of the instance. This endpoint supports only the GET verb and requires no authentication, XSRF protection header values, or mime-type headers. The `/status` endpoint has been designed to return sane output even when Stash is currently unavailable as a result of database migration or backup. Please note that other URLs such as `/log in` or `/rest/api/latest/application-properties` will redirect to the maintenance page when Stash is performing database migration or backup. Using these endpoints may unintentionally trigger failover when these URLs are used for monitoring the health of the system.

Example usage:

```
> curl -i -u user -X GET http://localhost:7990/stash/status
Enter host password for user 'user':
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-AREQUESTID: 1040x7x0
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Content-Type: application/json;charset=ISO-8859-1
Content-Length: 19
Date: Tue, 07 Jan 2014 17:20:04 GMT
{"state":"RUNNING"}
```

The following is a list of the responses the `/status` health check endpoint will return:

HTTP Status Code	Response entity	Description
200	{"state":"RUNNING"}	Stash is running normally
200	{"state":"MAINTENANCE"}	Stash is in maintenance mode
503	{"state":"STARTING"}	Stash is starting
503	{"state":"STOPPING"}	Stash is stopping
200	{"state":"FIRST_RUN"}	Stash is running for the first time and has not yet been configured
404		Stash failed to start up in an unexpected way (the web application failed to deploy)
500	{"state":"ERROR"}	Stash is in an error state

If a connection error occurs when trying to connect to the endpoint (but the server is reachable) then Tomcat has failed to start.

Monitoring frequency

Stash's health check is simple and not resource intensive. You should feel free to check as often as is deemed necessary to maximise continuity of Stash in your organisation. We do recommend, however, to not check more frequently than every 15 seconds so that the HA resource manager / cluster does not mistake transitory slowdowns such as stop-the-world garbage collection in Stash's JVM. We recommend a monitor timeout of 30 seconds because the first check after startup can be fairly slow. After startup completes, the check should take only a few milliseconds.

Requirements

- Monitoring scripts must use the `/status` URL. Any other URL may redirect to the maintenance page when a backup is being performed, unintentionally triggering failover.
- When a request to `/status` returns anything other than a 200 status code, Stash should be considered to be in an error state and should be failed over the standby node.

Solution in example HA implementation

The example HA implementation includes an OCF compliant script that's used for monitoring Stash's health. The script can be found here.

Failover

The following table outlines how we recommend that your HA resource manager responds to failure events:

Event	Response
Network connection from the request router to Stash is lost	Failover to a secondary node
Server failure	Failover to a secondary node
Stash crashes completely	Restart Stash on the active node
Stash reaches its memory limits (OOME)	Restart Stash on the active node
Stash loses connection to the database	Nothing. Stash will recover when the database comes back on line. Stash on another node will also fail to start if the database is unavailable.

The database is reported down	Nothing. Stash will recover when the database recovers. Stash on another node will also fail to start.
Stash fails to start up (e.g. wrong Git binary version)	Nothing. Manual intervention required. Stash on another node will also fail to start.

Split brain

A split-brain condition results when a cluster of nodes encounters a network partition and multiple nodes believe the others are dead and proceed to take over the cluster resources. In the context of a Stash HA installation this would involve multiple Stash instances running concurrently and making filesystem and database changes, potentially causing the filesystem and database to fall out of sync. As previously noted this must not be permitted to happen. There are several ways to address this:

Network redundancy

This involves configuring redundant and independent communications paths between nodes in the cluster. If you maximise the connectivity between nodes you minimise the likelihood of a network partition and a split brain. This is a preventative measure but it is still sometimes possible for the network to partition.

Resource fencing

This involves ensuring that the first node that believes the others are dead 'fences off' access to the resource that other nodes (which appear dead but may still be alive) may try to access. The losing nodes are prevented from making modifications, therefore maintaining consistency. In a Stash HA, the resources that would need to be fenced are the database and the replicated file system.

Node fencing or STONITH

This is a more aggressive tactic and again involves the first node that believes the others are dead, but instead of fencing off access to particular resources, it denies all resource access to them. This is most commonly achieved by power-cycling the losing nodes (aka "Shoot The Other Node In The Head" or STONITH). In a Stash HA, this would involve power-cycling the losing Stash servers.

Requirements

- The application should fail over to a secondary node when a server failure is detected by the cluster manager (that is, the whole node is down or unreachable).
- When an application failure is detected, the application should be restarted. If restarting does not resolve the issue, the application should be failed over to a secondary node.

Solution in example HA implementation

The example implementation uses Pacemaker to manage failover. Pacemaker in turn uses the provided OCF script to properly shut down the failing Stash and start Stash on the secondary node.

Please note that the vagrant provisioning script in the example implementation contains a simplified configuration that is aimed at testing failover. It configures Stash to immediately failover to a secondary node, without attempting to restart the application. It also disables the STONITH feature for ease of testing. In a production system, at least one restart should be attempted before failing over and STONITH should be enabled to handle 'split brain' occurrences.

Licensing

Developer licenses can be used for non-production installations of Stash deployed on a cold stand-by server. For more information see developer licenses.

Clustering with Stash Data Center

About Stash Data Center

Stash Data Center is the on-premises Git repository management solution for larger enterprises that require

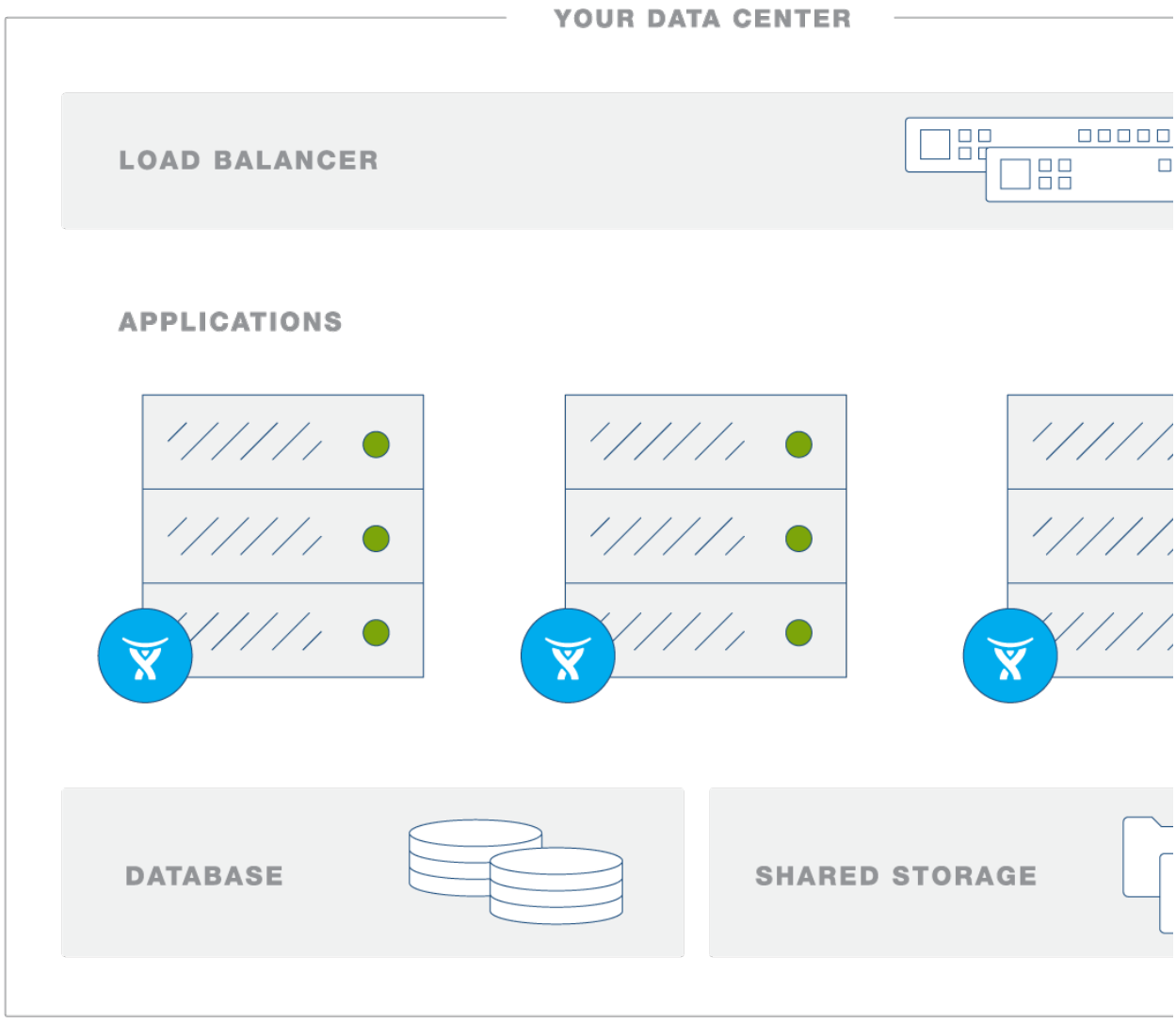
high availability and performance at scale. It allows everyone in your organization to easily collaborate on your Git repositories.

Stash Data Center is designed with enterprise scaling and infrastructure flexibility in mind for when you host Stash in your own data center. It provides enterprise teams with:

- **Performance at scale:** A cluster of many machines each running Stash can handle a greater load than a single machine.
- **High availability:** If one cluster node goes down, then the remaining cluster nodes can continue servicing requests so that users see little or no loss of availability.
- **Instant scalability:** You can rapidly provision extra capacity with almost no downtime.

A look at the architecture

Stash Data Center consists of a cluster of dedicated machines, connected like this:



Load balancer

The load balancer distributes requests from your users to the cluster nodes. If a cluster node goes down, the load balancer immediately detects the failure and automatically directs requests to the other nodes within seconds.

Application nodes

The cluster of Stash nodes share the workload of incoming requests. Failure of a cluster node causes virtually no loss of availability for users, because requests are immediately directed to other nodes.

Shared database and storage

Stash Data Center supports the same databases as Stash Server (except for MySQL).

A high-performance shared file system, accessible via NFS, stores repository, attachment and avatar data.

Learn more about Stash Data Center

- Stash Data Center Performance
- Failover for Stash Data Center
- Installing Stash Data Center
- Adding cluster nodes to Stash Data Center
- Stash Data Center Add-ons
- Stash Data Center FAQ
- Stash Enterprise Resources

Installing Stash Data Center

This page...

... describes how to migrate an existing instance of Stash to Stash Data Center.

For an overview, see Stash Data Center.

If you are installing Stash server...

... go straight to Getting started, instead.

We also recommend reading Using Stash in the enterprise.

If you just want to add another node...

... we suggest you take a look at Adding cluster nodes to Stash Data Center.

This guide assumes that you already have a production instance of Stash, and that you are aiming to migrate that to a Stash Data Center instance.

We recommend that you:

- Initiate the purchase of a Stash Data Center license by contacting us at https://www.atlassian.com/enterprise/contact.
- Set up and test Stash Data Center in your staging environment, before deploying to a production environment.
- Upgrade Stash, and then make a backup of your production instance of Stash.
- Restore a copy of this backup into your clustered staging environment.
- Test Stash Data Center with identical data (repositories, users, add-ons) to your production instance.

Regardless of the process you use, please smoke test your Stash Data Center instance every step of the way.

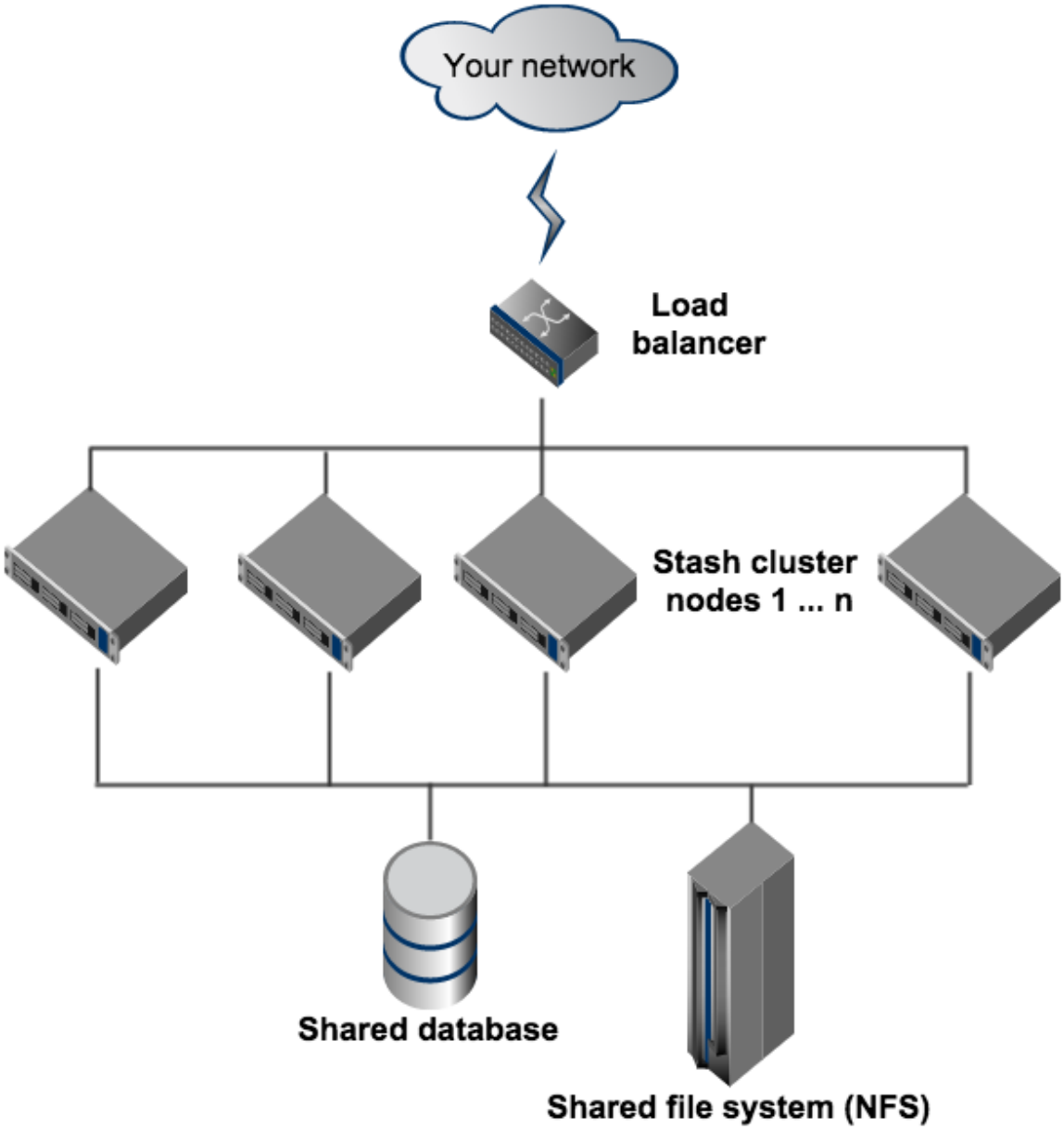
On this page:

- Overview and requirements
- 1. Upgrade your existing production instance of Stash
- 2. Back up your production instance
- 3. Provision your shared database
- 4. Provision your shared file system
- 5. Provision your cluster nodes
- 6. Start the first cluster node
- 7. Install and configure your load balancer
- 8. Add a new Stash cluster node to the cluster
- 9. Connect the new Stash cluster node to the load balancer
- 10. Repeat steps 8 and 9 for each additional cluster node
- 11. Congratulations!

Overview and requirements

It's worth getting a clear understanding of what you're aiming to achieve, before starting to provision your Stash Data Center.

A Stash Data Center instance consists of a cluster of dedicated machines connected like this:



The URL of the Stash Data Center instance will be the URL of the load balancer, so this is the machine that you will need to assign the name of your Stash instance in the DNS.

The remaining machines (Stash cluster nodes, shared database, and shared file system) do not need to be publicly accessible to your users.

Stash cluster nodes

The Stash cluster nodes all run the Stash Data Center web application.

- Each Stash cluster node must be a dedicated machine.
- The machines may be physical or virtual.
- The cluster nodes must be connected in a high speed LAN (that is, they must be physically in the same data center).
- The usual Stash supported platforms require ments, including those for Java and Git, apply to each cluster node.
- The cluster nodes do not all need to be absolutely identical, but for consistent performance we recommend they should be as similar as possible.

Load balancer

You can use the load balancer of your choice. Stash Data Center does ***not*** bundle a load balancer.

- Your load balancer should run on a dedicated machine.
- Your load balancer must have a high-speed LAN connection to the Stash cluster nodes (that is, it must be physically in the same data center).
- Your load balancer must support ***both*** HTTP mode (for web traffic) ***and*** TCP mode (for SSH traffic).
- Terminating SSL (HTTPS) at your load balancer and running plain HTTP from the load balancer to Stash is highly recommended for performance.

- All cluster nodes must run the same version of Stash Data Center.
- All cluster nodes must have synchronized clocks (for example, using NTP) and be configured with the identical timezone.

Shared database

You must run Stash Data Center on an external database. You can ***not*** use Stash's internal HSQL database with Stash Data Center.

- The shared database must run on a dedicated machine.
- The shared database must be available to all cluster nodes via a high-speed LAN (it must be in the same physical data center).
- All the usual database vendors in Stash's supported platforms are supported by Stash Data Center, with one exception: we do ***not*** recommend MySQL at this time due to inherent deadlocks that can occur in this database engine at high load.

- Your load balancer should support "session affinity" (also known as "sticky sessions").

If you don't have a preference for your load balancer, we provide instructions for `haproxy`, a popular Open Source software load balancer.

Shared file system

Stash Data Center requires a high performance shared file system such as a SAN, NAS, RAID server, or high-performance file server optimized for I/O.

- The shared file system must run on a dedicated machine.
- The file system must be available to all cluster nodes via a high-speed LAN (it must be in the same physical data center).
- The shared file system should be accessible via NFS as a single mount point.

What is stored on the shared file system?

- configuration files
- data directory, which includes:
  - repositories
  - attachments
  - avatars
- plugins

What is stored locally on each node?

- caches
- logs
- temporary files

1. Upgrade your existing production instance of Stash

Begin by upgrading your production Stash Server instance to the latest public release. This is necessary for several reasons:

- The Stash database and home directory layout often change in each release of Stash. Upgrading first will ensure that your production Stash Server instance and your Stash Data Center instance share identical data format, and you can switch between them at will.
- Any add-ons in your production instance can be verified as compatible with the latest release of Stash (or updated if not).
- Any performance or other comparisons between single-node Stash Server and multi-node Stash Data Center will be more meaningful.

Upgrade your Stash Server by following the instructions in the Stash upgrade guide.

2. Back up your production instance

Now, take a backup of your production Stash instance's database and home directory. For this you can:

- use the Stash backup client,
- use your own DIY backup solution, or
- just stop Stash and manually dump your database, and zip up the home directory.

3. Provision your shared database

Set up your shared database server. Note that clustered databases are not yet supported.

See Connecting Stash to an external database for more information.

> You ***must*** ensure your database is configured to allow enough concurrent connections. Stash by default uses up to 80 connections ***per cluster node***, which can exceed the default connection limit of some databases.
>
> For example, in PostgreSQL the default limit is usually 100 connections. If you use PostgreSQL, you may need to edit your `postgresql.conf` file, to increase the value of `max_connections`, and restart Postgres.

> We do ***not*** support MySQL for Stash Data Center at this time due to inherent deadlocks that can occur in this database engine at high load. If you currently use MySQL, you should migrate your data to another supported database (such as PostgreSQL) before upgrading your Stash Server instance to Stash Data Center. You can migrate databases (on a standalone Stash instance) using the Migrate database feature in Stash's Adminstration pages, or by using the Stash backup client.

4. Provision your shared file system

Set up your shared file server.

See Stash Data Center FAQ for performance guidelines when using NFS.

> You ***must*** ensure your shared file system server is configured with enough NFS server processes.
>
> For example, some versions of RedHat Enterprise Linux and CentOS have a default of 8 server processes. If you use one of these systems, you may need to edit your `/etc/sysconfig/nfs` file, increase the value of `RPCNFSDCOUNT`, and restart the `nfs` service.

> You ***must*** ensure your shared file system server has the NFS lock service ***enabled***. For example:
>
> - In some versions of Ubuntu Linux you must ensure that the `portmap` and `dbus` services are enabled for the NFS `lockd` to function.
> - In some versions of RedHat Enterprise Linux and CentOS, you must install the `nfs-utils` a nd `nfs-utils-lib` packages, and ensure the `rpcbind` and `nfslock` services are running.

Create a Stash user account (recommended name `atlstash`) on the shared file system server to own everything in the Stash shared home directory. This user account must have the same UID on all cluster nodes and the shared file system server. In a fresh Linux install the UID of a newly created account is typically 1001, but in general there is no guarantee that this UID will be free on every Linux system. Choose a UID for `atlstash` that's free on all your cluster nodes and the shared file system server, and substitute this for `1001` in the following command:

```
sudo useradd -c "Atlassian STASH" -u 1001 atlstash
```

> You ***must*** ensure that the atlstash user has the same UID on all cluster nodes and the shared file system server.

Then restore the backup you have taken in step 2 into the new shared database and shared home directory.

Only the `shared` directory in the Stash home directory needs to be restored into the shared home directory. The remaining directories (`bin`, `caches`, `export`, `lib`, `log`, `plugins`, and `tmp`) contain only caches and temporary files, and do not need to be restored.

You ***must*** ensure that the user running Stash (usually `atlstash`) is able to read and write everything in the Stash home directory, both the node-local part and the shared part (in NFS). The easiest way to do this is to ensure that:

1. `atlstash` owns all files and directories in the Stash home directory,
2. `atlstash` has the recommended `umask` of `0027`, and
3. `atlstash` has the same UID on all machines.

Do ***not*** run Stash as `root`. Many NFS servers squash accesses by `root` to another user.

5. Provision your cluster nodes

1. We highly recommend provisioning cluster nodes using an automated configuration management tool such as Chef, Puppet, or Vagrant, or by spinning up identical virtual machine snapshots.
2. On each cluster node, mount the shared home directory as `${STASH_HOME}/shared`. For example, suppose your Stash home directory is`/var/atlassian/application-data/stash`, and your shared home directory is available as an NFS export called `stash-san:/stash-shared`. Add the following line to `/etc/fstab` on each cluster node:

   /etc/fstab

   ```
   stash-san:/stash-shared /var/atlassian/application-data/stash/shared nfs
   nfsvers=3,lookupcache=pos,noatime,intr,rsize=32768,wsize=32768 0 0
   ```

   Only the `${STASH_HOME}/shared` directory should be shared between cluster nodes. All other directories, including `${STASH_HOME}`, should be node-local (that is, private to each node).

   Stash Data Center checks during startup that `${STASH_HOME}` is node local and `${STASH_HOME}/shared` is shared, and will fail to form a cluster if this is not true.

   Your shared file system ***must*** provide sufficient consistency for Stash and Git.

   Linux NFS clients require the `lookupcache=pos` mount option to be specified for proper consistency.

   NFSv4 may have issues in Linux kernels from about version 3.2 to 3.8 inclusive. The issues may cause very high load average, processes hanging in "uninterruptible sleep", and in some cases may require rebooting the machine. We recommend using NFSv3 unless you are 100% sure that you know what you're doing and your operating system is free from such issues.

   Linux NFS clients should use the `nfsvers=3` mount option to force NFSv3.

   Then mount it:

   ```
   mkdir -p /var/atlassian/application-data/stash/shared
   sudo mount -a
   ```

3. Ensure all your cluster nodes have synchronized clocks and identical timezone configuration. For example, in RedHat Enterprise Linux or CentOS:

```
sudo yum install ntp
sudo service ntpd start
sudo tzselect
```

In Ubuntu Linux:

```
sudo apt-get install ntp
sudo service ntp start
sudo dpkg-reconfigure tzdata
```

For other operating systems, consult your system documentation.

> The system clocks on your cluster nodes ***must*** remain reasonably synchronized (say, to within a few seconds or less). If your system clocks drift excessively or undergo abrupt "jumps" of minutes or more, then cluster nodes may log warnings, become slow, or in extreme cases become unresponsive and require restarting. You should run the NTP service on all your cluster nodes with identical configuration, and never manually tamper with the system clock on a cluster node while Stash Data Center is running.

4. Download the latest Stash Data Center distribution from https://www.atlassian.com/software/stash/download, and install Stash as normal on all the cluster nodes. See Getting started.

6. Start the first cluster node

Edit the file `${STASH_HOME}/shared/stash-config.properties`, and add the following lines:

```
# Use multicast to discover cluster nodes (recommended).
hazelcast.network.multicast=true

# If your network does not support multicast, you may uncomment the following
lines and substitute
# the IP addresses of some or all of your cluster nodes. (Not all of the cluster
nodes have to be
# listed here but at least one of them has to be active when a new node joins.)
#hazelcast.network.tcpip=true
#hazelcast.network.tcpip.members=192.168.0.1:5701,192.168.0.2:5701,192.168.0.3:57
01

# The following should uniquely identify your cluster on the LAN.
hazelcast.group.name=your-stash-cluster
hazelcast.group.password=your-stash-cluster
```

Using multicast to discover cluster nodes (`hazelcast.network.multicast=true`) is recommended, but requires all your cluster nodes to be accessible to each other via a multicast-enabled network. If your network does not support multicast then you can set `hazelcast.network.multicast=false`, `hazelcast.network.tcpip=true`, and `hazelcast.network.tcpip.members` to a comma-separated list of cluster nodes instead. Only enable ***one*** of `hazelcast.network.tcpip` or `hazelcast.network.multicast`, not both!

Choose a name for `hazelcast.group.name` and `hazelcast.group.password` that uniquely identifies the cluster on your LAN. If you have more than one cluster on the same LAN (for example, other Stash Data Center instances or other products based on similar technology such as Confluence Data Center) then you ***must*** assign each cluster a distinct name, to prevent them from attempting to join together into a "super cluster".

Then start Stash. See Starting and stopping Stash.

Then go to `http://<stash>:7990/admin/license`, and install the Stash Data Center license you were issued. Restart Stash for the change to take effect. If you need a Stash Data Center license, please contact us!

7. Install and configure your load balancer

You can use the load balancer of your choice, either hardware or software. Stash Data Center does ***not*** bund le a load balancer.

Your load balancer must proxy three protocols:

Protocol	Typical port on the load balancer	Typical port on the Stash cluster nodes	Notes
HTTP	80	7990	HTTP mode. Session affinity ("sticky sessions") should be enabled using the 52-character `JSESSIONID` cookie.
HTTPS	443	7990	HTTP mode. Terminating SSL at the load balancer and running plain HTTP to the Stash cluster nodes is highly recommended.
SSH	7999	7999	TCP mode.

For best performance, your load balancer should support session affinity ("sticky sessions") using the `JSESSIONID` cookie. By default, Stash Data Center assumes that your load balancer always directs each user's requests to the same cluster node. If it does not, users may be unexpectedly logged out or lose other information that may be stored in their HTTP session.

Stash Data Center also provides a property `hazelcast.http.sessions` that can be set in `${STASH_HOME}/shared/stash-config.properties` that provides finer control over HTTP session management. This property can be set to one of the following values:

- `local` (the default): HTTP sessions are managed per node. When used in a cluster, the load balancer ***must*** have session affinity ("sticky sessions") enabled. If a node fails or is shut down, users that were assigned to that node may need to log in again.
- `sticky`: HTTP sessions are distributed across the cluster with a load balancer configured to use session affinity ("sticky sessions"). If a node fails or is shut down, users should not have to log in again. In this configuration, session management is optimized for sticky sessions and will not perform certain cleanup tasks for better performance.
- `replicated`: HTTP sessions are distributed across the cluster. If a node fails or is shut down, users should not have to log in again. The load balancer does not need to be configured for session affinity ("sticky sessions"), but performance is likely to be better if it is.

Both the `sticky` and `replicated` options come with some performance penalty, which can be substantial if session data is used heavily (for example, in custom plugins). For best performance, `local` (the default) is recommended.

When choosing a load balancer, it must support the HTTP, HTTPS, and TCP protocols. Note that:

- Apache does ***not*** support TCP mode load balancing.
- HAProxy versions older than 1.5.0 do ***not*** support HTTPS.

If your load balancer supports health checks of the cluster nodes, configure it to perform a periodic HTTP GET of `http:// <stash>:7990/status`, where `<stash>` is the cluster node's name or IP address. This returns one of two HTTP status codes:

- 200 OK

- 500 Internal Server Error

If a cluster node does not return 200 OK within a reasonable amount of time, the load balancer should not direct any traffic to it.

You should then be able to navigate to `http://<load-balancer>/`, where `<load-balancer>` is your load balancer's name or IP address. This should take you to your Stash front page.

Example: HAProxy load balancer

If you don't have a particular preference or policy for load balancers, you can use HAProxy which is a popular Open Source software load balancer.

> If you choose HAProxy, you ***must*** use a minimum version of 1.5.0. Earlier versions of HAProxy do not support HTTPS.

Here is an example `haproxy.cfg` configuration file (typically found in the location `/etc/haproxy/haproxy.cfg`). This assumes:

- Your Stash cluster node is at address 192.168.0.1, listening on the default ports 7990 (HTTP) and 7999 (SSH).
- You have a valid SSL certificate at `/etc/cert.pem`.

haproxy.cfg Expand source

```
global
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

defaults
    log                     global
    option                  dontlognull
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
    tune.ssl.default-dh-param 1024
    errorfile               408 /dev/null # Workaround for Chrome 35-36 bug.  See
http://blog.haproxy.com/2014/05/26/haproxy-and-http-errors-408-in-chrome/

frontend stash_http_frontend
    bind *:80
    bind *:443 ssl crt /etc/cert.pem ciphers RC4-SHA:AES128-SHA:AES256-SHA
    default_backend stash_http_backend

backend stash_http_backend
    mode http
    option httplog
    option httpchk GET /status
    option forwardfor
    option http-server-close
    appsession JSESSIONID len 52 timeout 1h
    balance roundrobin
```

```
    cookie JSESSIONID prefix
    stick-table type string len 52 size 5M expire 30m
    stick store-response set-cookie(JSESSIONID)
    stick on cookie(JSESSIONID)
    server stash01 192.168.0.1:7990 check inter 10000 rise 2 fall 5
    #server stash02 192.168.0.2:7990 check inter 10000 rise 2 fall 5
    # The following "backup" servers are just here to show the startup page when
all nodes are starting up
    server backup01 192.168.0.1:7990 backup
    #server backup02 192.168.0.2:7990 backup

frontend stash_ssh_frontend
    bind *:7999
    default_backend stash_ssh_backend
    timeout client 15m
    maxconn 50

backend stash_ssh_backend
    mode tcp
    balance roundrobin
    server stash01 192.168.0.1:7999 check port 7999
    #server stash02 192.168.0.2:7999 check port 7999
    timeout server 15m

listen admin
    mode http
```

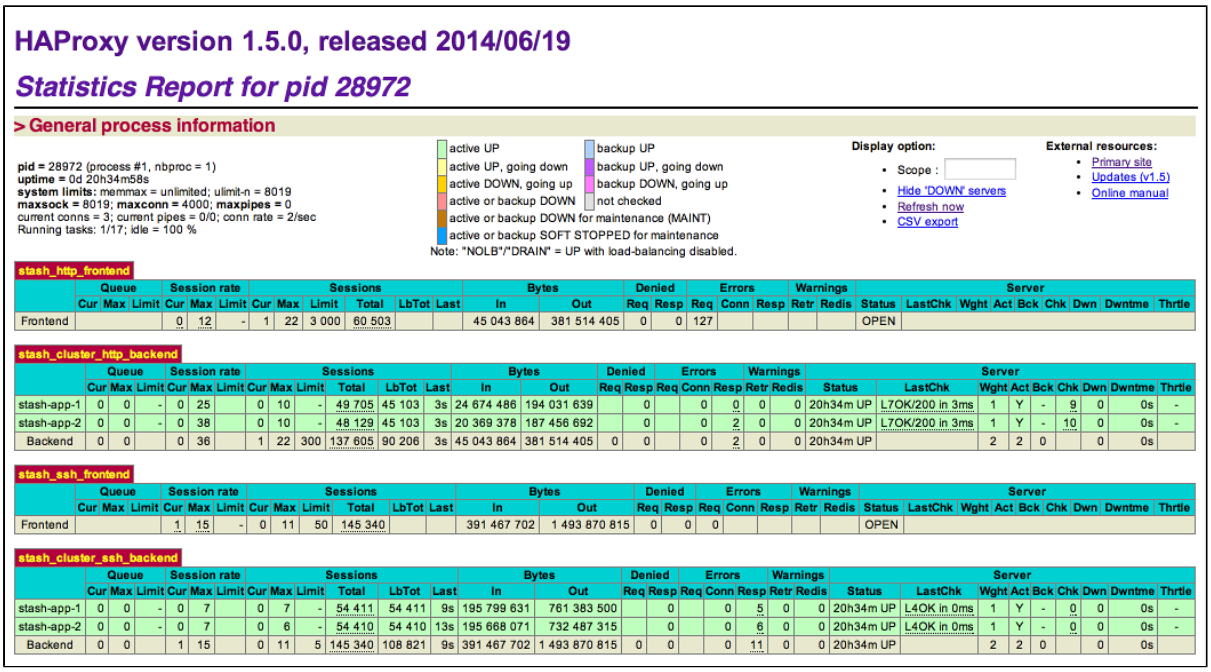
```
    bind *:8090
    stats enable
    stats uri /
```

> Review the contents of the `haproxy.cfg` file carefully, and customize it for your environment. See http://www.haproxy.org/ for more information about installing and configuring `haproxy`.

Once you have configured the `haproxy.cfg` file, start the `haproxy` service.

```
sudo service haproxy start
```

You can also monitor the health of your cluster by navigating to HAProxy's statistics page at `http://<load-balancer>:8090/`. You should see a page similar to this:



HAProxy version 1.5.0, released 2014/06/19

Statistics Report for pid 28972

> General process information

pid = 28972 (process #1, nbproc = 1)
uptime = 0d 20h34m58s
system limits: memmax = unlimited; ulimit-n = 8019
maxsock = 8019; **maxconn** = 4000; **maxpipes** = 0
current conns = 3; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/17; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
backup UP
backup UP, going down
backup DOWN, going up
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

stash_http_frontend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	12	-	1	22	3 000	60 503			45 043 864	381 514 405	0	0	127					OPEN								

stash_cluster_http_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
stash-app-1	0	0	-	0	25		0	10	-	49 705	45 103	3s	24 674 486	194 031 639		0		0	0	0	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	9	0	0s	-
stash-app-2	0	0	-	0	38		0	10	-	48 129	45 103	3s	20 369 378	187 456 692		0		0	2	0	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	10	0	0s	-
Backend	0	0		0	36		1	22	300	137 605	90 206	3s	45 043 864	381 514 405	0	0		0	2	0	0	20h34m UP		2	2	0		0	0s	

stash_ssh_frontend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	15	-	0	11	50	145 340			391 467 702	1 493 870 815	0	0	0					OPEN								

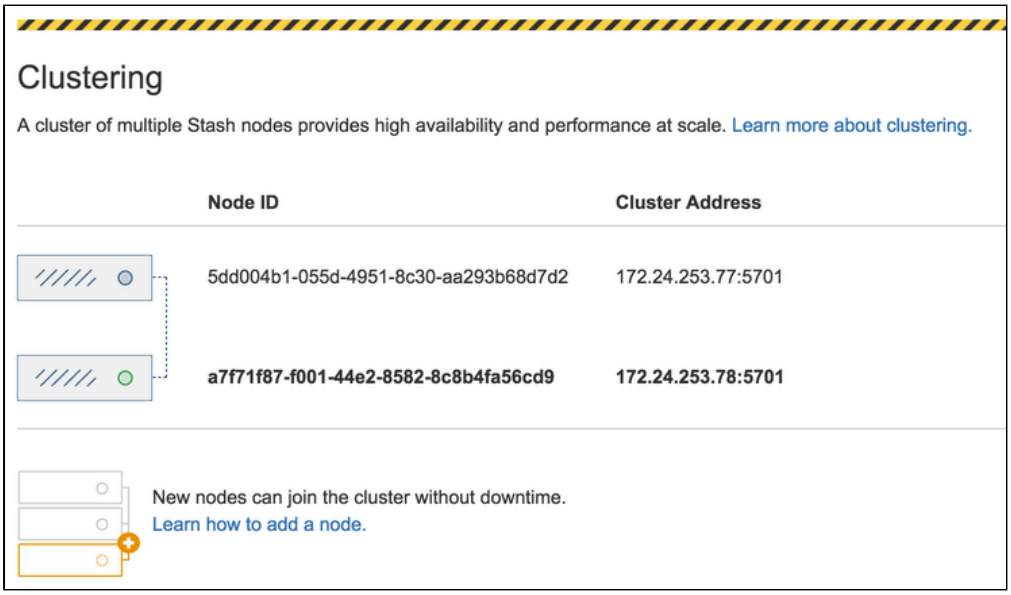
stash_cluster_ssh_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
stash-app-1	0	0	-	0	7		0	7	-	54 411	54 411	9s	195 799 631	761 383 500		0		0	5	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
stash-app-2	0	0	-	0	7		0	6	-	54 410	54 410	13s	195 668 071	732 487 315		0		0	6	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0		1	15		0	11	5	145 340	108 821	9s	391 467 702	1 493 870 815	0	0		0	11	0	0	20h34m UP		2	2	0		0	0s	

8. Add a new Stash cluster node to the cluster

Go to a new cluster node, and start Stash. See Starting and stopping Stash.

Once Stash has started, go to `http://<load-balancer>/admin/clustering`. You should see a page similar to this:



Verify that the new node you have started up has successfully joined the cluster. If it does not, please check your network configuration and the `${STASH_HOME}/log/atlassian-stash.log` files on all nodes. If you are unable to find a reason for the node failing to join successfully, please contact Atlassian Support.

9. Connect the new Stash cluster node to the load balancer

If you are using your own hardware or software load balancer, consult your vendor's documentation on how to add the new Stash cluster node to the load balancer.

If you are using HAProxy, just uncomment the lines

```
server stash02 192.168.0.2:7990 check inter 10000 rise 2 fall 5
```

```
server stash02 192.168.0.2:7999 check port 7999
```

in your `haproxy.cfg` file and restart `haproxy`:

```
sudo service haproxy restart
```

Verify that the new node is in the cluster and receiving requests by checking the logs on each node to ensure both are receiving traffic and also check that updates done on one node are visible on the other.

10. Repeat steps 8 and 9 for each additional cluster node

11. Congratulations!

You have now set up a clustered instance of Stash Data Center! We are very interested in hearing your feedback on this process – please contact us!

> For any issues please raise a support ticket and mention that you are following the Installing Stash Data Center page.

Please see Using Stash in the enterprise for information about using Stash in a production environment.

Adding cluster nodes to Stash Data Center

This page...

... describes how to add another cluster node to an existing instance of Stash Data Center.

If you are moving to Stash Data Center...

... go straight to Installing Stash Data Center, instead.

If you are new to Stash Data Center...

... we suggest you take a look at Clustering with Stash Data Center.

Provisioning a cluster node

You can rapidly scale the capacity of Stash Data Center, with very little downtime, by provisioning extra cluster nodes.

We highly recommend provisioning cluster nodes using an automated configuration management tool such as Chef, Puppet, or Vagrant, or by spinning up identical virtual machine snapshots.

The Stash cluster nodes all run the Stash Data Center web application:

- Each Stash cluster node must be a dedicated machine.
- The machines may be physical or virtual.
- The cluster nodes must be connected in a high speed LAN (that is, they must be physically in the same data center).
- The usual Stash supported platforms requirements, including those for Java and Git, apply to each cluster node.
- The cluster nodes do not all need to be absolutely identical, but for consistent performance we recommend they should be as similar as possible.
- All cluster nodes must run the same version of Stash Data Center.
- All cluster nodes must have synchronized clocks (for example, using NTP) and be configured with the identical timezone.

Provisioning a cluster node involves the following steps:

- 1. Mount the shared home directory on the node
- 2. Install Stash Data Center on the node
- 3. Add the node to the cluster
- 4. Connect the node to the load balancer

1. Mount the shared home directory on the node

The Stash Data Center makes use of a shared file system that lives on a dedicated machine and is accessible using NFS. See Installing Stash Data Center for more information.

Mount the shared home directory as `${STASH_HOME}/shared`. For example, suppose your Stash home directory is `/var/atlassian/application-data/stash`, and your shared home directory is available as an NFS export called `stash-san:/stash-shared`. Add the following line to `/etc/fstab` on the cluster node:

/etc/fstab

```
stash-san:/stash-shared /var/atlassian/application-data/stash/shared nfs
nfsvers=3,lookupcache=pos,noatime,intr,rsize=32768,wsize=32768 0 0
```

Then mount it:

```
mkdir -p /var/atlassian/application-data/stash/shared
sudo mount -a
```

2. Install Stash Data Center on the node

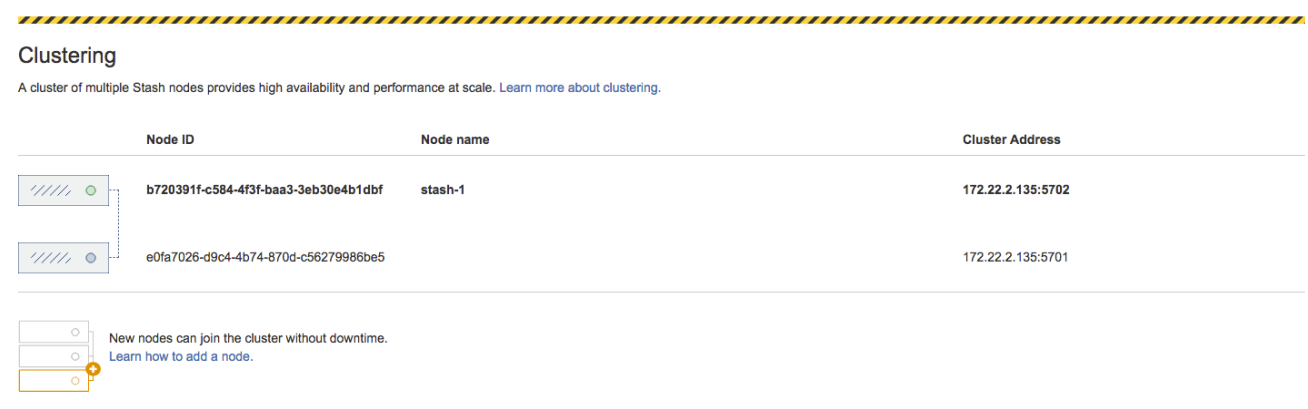
Download the latest Stash Data Center distribution from https://www.atlassian.com/software/stash/download, and install Stash as normal on the cluster node. See Getting started.

3. Add the node to the cluster

Start Stash on the new node. See Starting and stopping Stash. You can optionally give the node a persistent, human readable name by setting a `node.name` system property under `JVM_SUPPORT_RECOMMENDED_ARGS` in `setenv.sh`. For example:

```
-Dnode.name=stash-1
```

Once Stash has started, go to `http://<load-balancer>/admin/clustering`. You should see the new node listed, similarly to this:



Verify that the new node you have started up has successfully joined the cluster. If it does not, please check your network configuration and the `${STASH_HOME}/log/atlassian-stash.log` files on all nodes. If you are unable to find a reason for the node failing to join successfully, please contact Atlassian Support.

4. Connect the node to the load balancer

The Stash Data Center makes use of a load balancer to distribute requests from your users to the cluster nodes. If a cluster node goes down, the load balancer immediately detects the failure and automatically directs requests to the other nodes within seconds. See Installing Stash Data Center for more information.

If you are using a hardware or software load balancer other than HAProxy, consult your vendor's documentation on how to add the new Stash cluster node to the load balancer.

If you are using HAProxy, simply add the following lines to your `haproxy.cfg` file:

```
# In the backend stash_http_backend section, add:
server stash<xx> 192.168.0.<x>:7990 check inter 10000 rise 2 fall 5
```

```
# In the backend stash_ssh_backend section, add:
server stash<xx> 192.168.0.<x>:7999 check port 7999
```

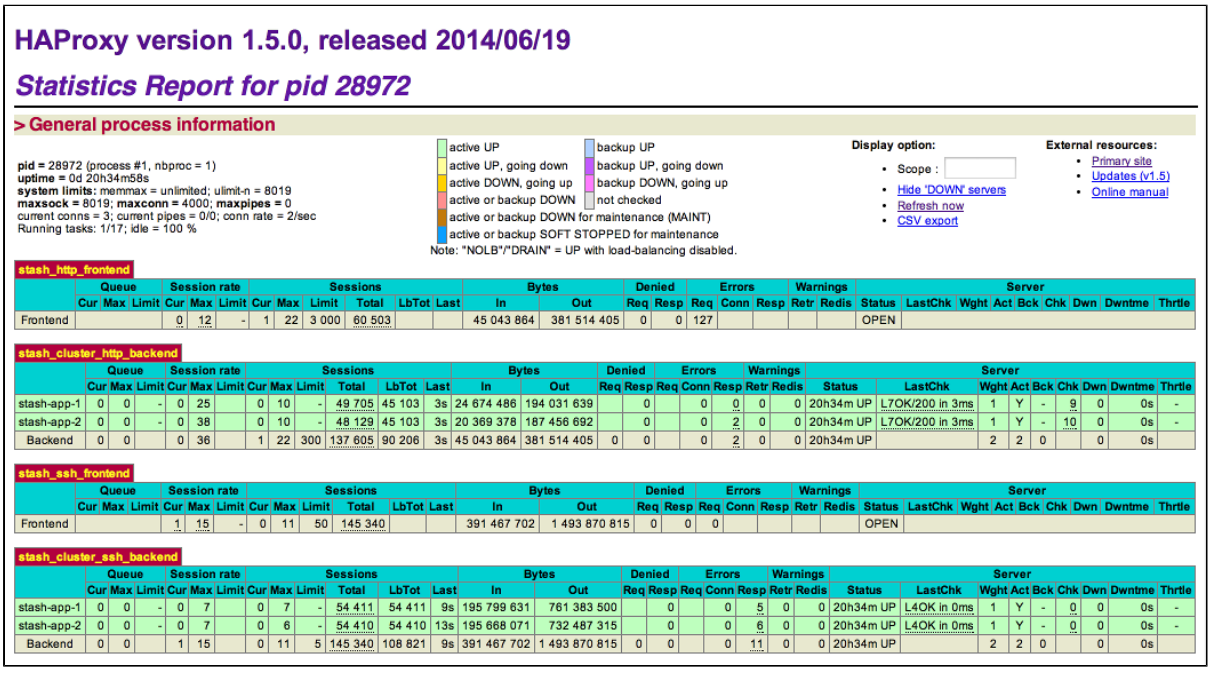
where the values for <x> and <xx> don't conflict with an existing node.

Now restart HAProxy:

```
sudo service haproxy restart
```

Verify that the new node is in the cluster and receiving requests by checking the logs on each node to ensure that all are receiving traffic. Also check that updates done on one node are visible on the other nodes.

You can monitor the health of your cluster by navigating to HAProxy's statistics page at `http://<load-balancer>:8090/`. You should see a page similar to this:



HAProxy version 1.5.0, released 2014/06/19

Statistics Report for pid 28972

> General process information

pid = 28972 (process #1, nbproc = 1)
uptime = 0d 20h34m58s
system limits: memmax = unlimited; ulimit-n = 8019
maxsock = 8019; **maxconn** = 4000; **maxpipes** = 0
current conns = 3; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/17; idle = 100 %

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

stash_http_frontend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	12	-	1	22	3 000	60 503			45 043 864	381 514 405	0	0	127					OPEN								

stash_cluster_http_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
stash-app-1	0	0	-	0	25		0	10	-	49 705	45 103	3s	24 674 486	194 031 639		0		0	0	0	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	9	0	0s	-
stash-app-2	0	0	-	0	38		0	10	-	48 129	45 103	3s	20 369 378	187 456 692		0		0	2	0	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	10	0	0s	-
Backend	0	0		0	36		1	22	300	137 605	90 206	3s	45 043 864	381 514 405	0	0		0	2	0	0	20h34m UP		2	2	0		0	0s	

stash_ssh_frontend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	15	-	0	11	50	145 340			391 467 702	1 493 870 815	0	0	0					OPEN								

stash_cluster_ssh_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
stash-app-1	0	0	-	0	7		0	7	-	54 411	54 411	9s	195 799 631	761 383 500		0		0	5	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
stash-app-2	0	0	-	0	7		0	6	-	54 410	54 410	13s	195 668 071	732 487 315		0		0	6	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0		1	15		0	11	5	145 340	108 821	9s	391 467 702	1 493 870 815	0	0		0	11	0	0	20h34m UP		2	2	0		0	0s	

For any issues please raise a support ticket and mention that you are following the Adding cluster nodes to Stash Data Center page.

Enabling JMX counters for performance monitoring

This article describes how to expose JMX MBeans within Stash for monitoring with a JMX client.

What is JMX?

JMX (Java Management eXtensions) is a technology for monitoring and managing Java applications. JMX uses objects called MBeans (Managed Beans) to expose data and resources from your application.

On this page

- What is JMX?
- Expose JMX MBeans within Stash
- Expose JMX MBeans when Stash is run as a Windows service
- Verify JMX is configured correctly

Related reading

- Understanding JMX (Oracle)

Why would I want to enable JMX monitoring within Stash?

For large instances of Stash or Stash Data Center, enabling JMX allows you to more easily monitor the consumption of application resources. This enables you to make better decisions about how to maintain and optimize machine resources.

What can I monitor with JMX?

It is possible to monitor various statistics using JMX counters within Stash. Below are some examples of some statistics that can be monitored.

Stash repository statistics

- Total number of projects
- Total number of repositories
- Git pushes and pulls
- Various thread pools and attributes

Thread pools

Thread pool	Description
IoPumpThreadPool	Threads that handle external process IO
ScheduledThreadPool	Thread pool that takes care of several miscellaneous scheduled tasks
EventThreadPool	Threads that dispatch events to @EventListenermethods

Thread pool attributes

Name	Description
ActiveCount	Returns the approximate number of threads that are actively executing tasks.
MaximumPoolSize	Returns the maximum allowed number of threads.
PoolSize	Returns the current number of threads in the pool.

QueueLength	The number of tasks awaiting execution by the thread pool.
LargestPoolSize	The largest number of threads that have ever been simultaneously in the pool.
CompletedTaskCount	The approximate total number of tasks that have completed execution. Because the states of tasks and threads may change dynamically during computation, the returned value is only an approximation, but one that does not ever decrease across successive calls.

Interesting 3rd party library attributes

Stash exposes the JMX attributes from number of third party libraries. Listed below is a sample of the attributes that are particularly interesting from an operations perspective.

BoneCP (com.jolbox.bonecp.BoneCP)

Name	Description
TotalLeased	Returns total number of connections currently in use by stash
ConnectionWaitTimeAvg	Return the average time it takes for a getConnection request to be services (in ms).
StatementExecutionTimeAvg	Return the average execution time for prepared statements to execute (in ms).

Hibernate (*org.hibernate.com.stash.core.org.hibernate.stat.Statistics.org.hibernate.stat.internal.ConcurrentStatisticsImpl)*

Name	Description
QueryCacheHitCount	Get the global number of cached queries successfully retrieved from cache
QueryCacheMissCount	Get the global number of cached queries *not* found in cache
SecondLevelCacheHitCount	Global number of cacheable entities/collections successfully retrieved from the cache
SecondLevelCacheMissCount	Global number of cacheable entities/collections not found in the cache and loaded from the database.

Expose JMX MBeans within Stash

To enable Stash to publish specific statistics using JMX you need to

1. Modify the `stash-config.properties` file.
2. Create a JMX password file for secure access to JMX monitoring.
3. Modify the `setenv.sh` file to enable Stash to expose JMX Mbeans.

These changes will not take effect until Stash has been restarted.

Modify the stash-config properties file

To modify (or create) the stash-config.properties file

1. Create the `stash-config.properties` file, in the `shared` folder of your Stash home directory. Take care to use the standard format for Java properties files.

> The `stash-config.properties` file is created automatically if you previously performed a database migration.

2. Add this property to the file.

```
jmx.enabled=true
```

Set up the JMX password file

To set up a JMX password file to secure access to JMX monitoring

1. Create a file named *`jmx.access.`*

> This file will contain password information. Ensure the file is only readable by the secure user Stash will run under. However, note that if the Stash user cannot read the file Stash will fail to start.

2. Edit the `jmx.access` file to include this property and save the file.

```
monitorRole=<password>
```

If you wish to user a username other than `monitorRole` or `controlRole` you will need to modify the *jmxremote.access* file that is bundled with stash in the `.install4j/jre.bundle/Contents/Home/jre/lib/management/` directory.

Modify the Stash environment file

To modify the `setenv.sh` (for Windows `setenv.bat`) files to enable JMX monitoring for Stash

1. Within the `bin` directory, locate the file `setenv.sh` (for Windows `setenv.bat`) and change these properties.

```
JMX_REMOTE_AUTH=password
JMX_REMOTE_PORT=3333
RMI_SERVER_HOSTNAME=-Djava.rmi.server.hostname=<hostname>
JMX_PASSWORD_FILE=<path>/jmx.access
```

2. Restart Stash.

Expose JMX MBeans when Stash is run as a Windows service

To expose JMX MBeans when Stash is run as a Windows service.

1. Stop the Stash service.
2. Open the command line prompt and enter.

```
cmd
```

3. Navigate to the Stash `bin` directory.

```
cd <Stash Installation dir>\bin
```

4. Run this command.

```
tomcat8w //ES//AtlassianStash
```

5. In the window that appears, click on the Java tab to see the list of current startup options. Under "Java Options:" form, input the value

```
-Dcom.sun.management.jmxremote.port=<JMX_REMOTE_PORT>
-Djava.rmi.server.hostname=<hostname>
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.password.file=<JMX_PASSWORD_FILE>
```

> Ensure the owner of this password file is the secure user Stash will run as. If the Stash user cannot read the file, Stash will fail to start.

6. Replace the values within the < > characters.

```
JMX_REMOTE_PORT=3333
JMX_PASSWORD_FILE=<path>\jmx.access
```

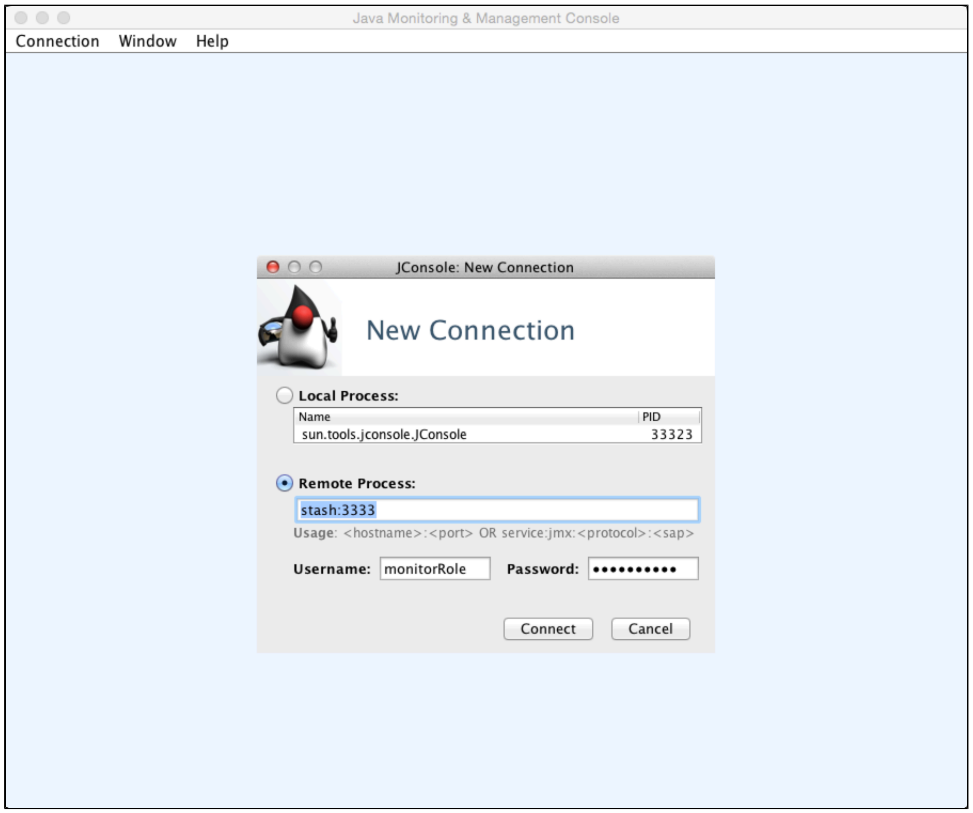
7. Restart Stash Service.

Verify JMX is configured correctly

These steps use JConsole to test that JMX has been configured correctly. JConsole is a utility that ships with the Oracle JDK.

1. To start the jconsole utility, from a command line prompt enter

```
jconsole
```



2. Create a new JConsole connection with similar connection settings.

stash	the hostname of the instance of Stash to monitor
3333	the JMX port number previously configured.
username, password	values configured within the JMX password file `jmx.access`.

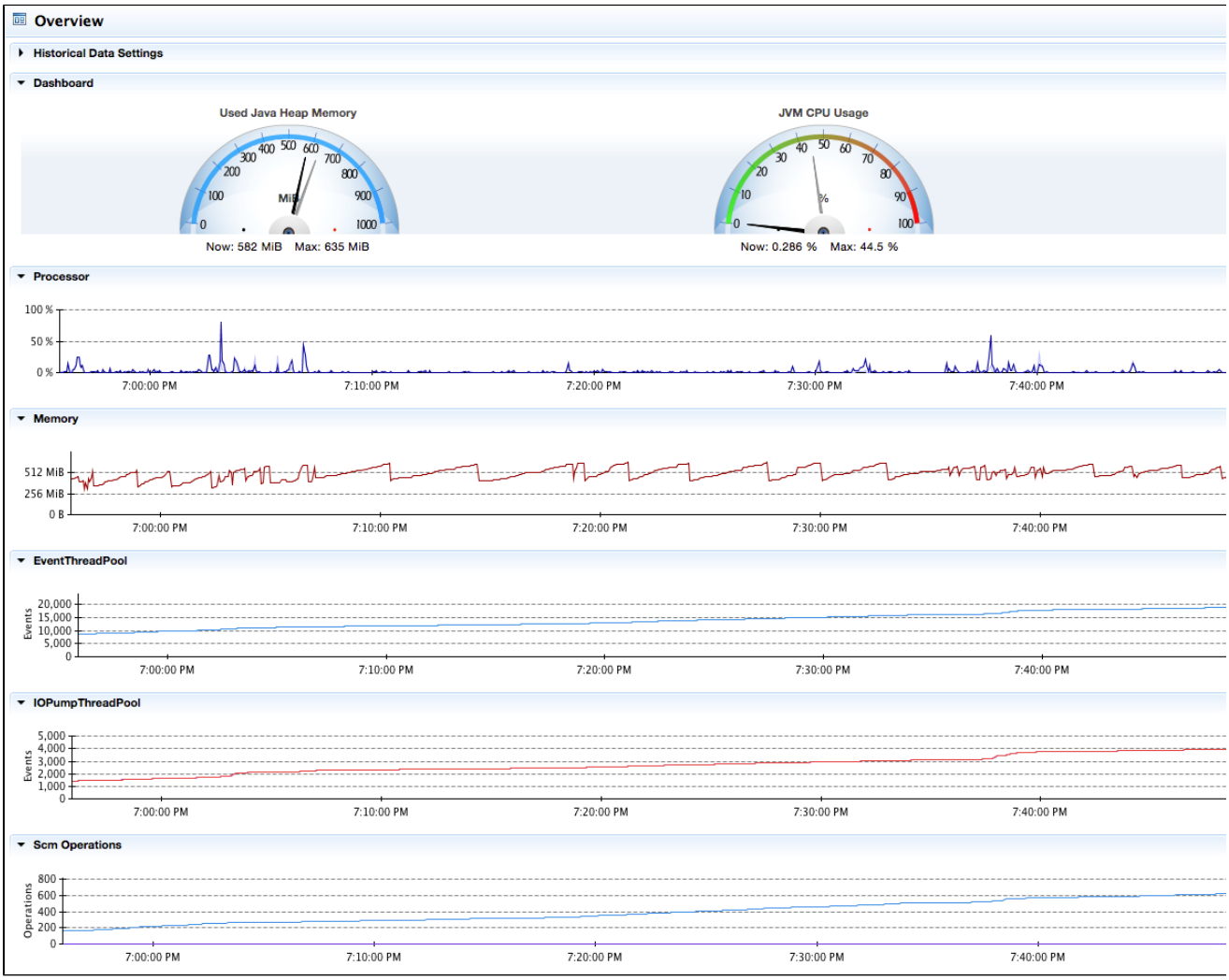
3. Click **Connect**.

When configured correctly, you will see these properties.

com.atlassian.stash	• Projects • Repositories • ScmStatistics • Tickets
com.atlassian.stash.thread-pools	• EventThreadPool • IoPumpThreadPool • ScheduledThreadPool

Example performance dashboard

This dashboard was generated using Java Mission Control that ships with the Oracle JDK (since 1.7u40). See the documentation that comes with your JMX client of choice for more information.



Configuring JMX to use SSL

You can find information about the options for configuring JMX to use SSL in the setenv files. Comprehensive documentation is available from Oracle.

Getting started with Stash and AWS

This page gives an overview of using Stash with AWS	To get started in AWS quickly ... set up a CloudFormation stack using our Quick Start guide.	To learn more about the deployment options for Stash in AWS ... see launching Stash in

AWS manually.

Running Stash in the Amazon Web Services (AWS) cloud can give you scalable computing capacity without the need to invest in hardware up front. To this end, Atlassian provides:

- an Amazon Machine Image (AMI) that you can launch in AWS as a "turnkey" deployment of Stash Server, or use as the starting point for customizing your own more complex deployments,
- an Amazon CloudFormation template that automates the process of spinning up a Stash Server instance in EC2, and
- tools and guidelines for backing up, restoring, sizing, and administering your Stash Server instances in AWS.

Running Stash Data Center in AWS is not supported at this time.

Quick Start guide

The simplest way to launch Stash Server in AWS is to use Atlassian's public Amazon CloudFormation template. See Quick Start with Stash and AWS.

Launching Stash in AWS manually

For more precise control over the components enabled within the Atlassian Stash AMI, including AWS-specific configuration, network and security settings, Launching Stash in AWS manually describes how to launch the AMI by running the EC2 launch wizard.

Performance guidelines

To get the best performance out of your Stash Server deployment in AWS, it's important not to under-provision your instance's CPU, memory, or I/O resources. We provide specific recommendations on choosing AWS EC2 and EBS settings for best performance when running Stash in AWS. See Recommendations for running Stash Server in AWS.

Backing up Stash in AWS

Atlassian also provides Stash DIY Backup utilities that back up and restore your Stash instance in AWS using native AWS snapshots. This provides a number of advantages:

- Performance: AWS snapshots occur asynchronously resulting in shorter backup downtime for your instances.
- Durability: The underlying storage of AWS snapshots is in Amazon S3, which is stored redundantly and with high durability.
- Availability: AWS snapshots are available across an entire AWS region, and are available for restore even in the event of an outage affecting an entire Availabilty Zone (AZ).

To learn more about how to back up and restore a Stash instance in AWS, see Using Stash DIY Backup in AWS.

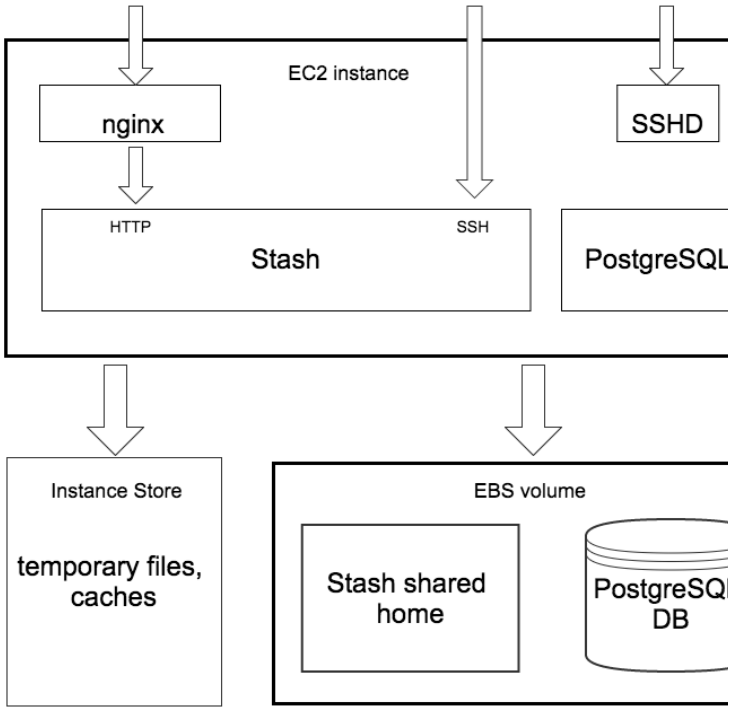
The Atlassian Stash AMI

The Atlassian Stash AMI provides a typical Stash Server deployment in AWS, pre-configured and ready to launch. See Launching Stash in AWS manually.

The components bundled in the Atlassian Stash AMI are

- Stash (either the latest version or a version of your choice),
- an external PostgreSQL database,
- nginx as a reverse proxy,
- the Stash DIY Backup utilities pre-configured

for native AWS snapshots,
- an EBS Volume and Instance Store to hol
the data.



Administering Stash in AWS

See Administering Stash in AWS for information about performing administration tasks on a Stash instance within AWS, including

- configuring variables when launching Stash in AWS
- maintaining, resizing, upgrading, migrating, and customizing your Stash deployment in AWS
- additional details about the components within the Atlassian Stash AMI

Securing Stash within AWS

AWS is accessed over the public Internet, so it is important to apply appropriate security measures when running Stash there. See Best practices for securing Stash in AWS for security guidance on a range of security topics, including Amazon Virtual Private Cloud (VPC), Security Groups, and SSL.

Quick Start with Stash and AWS

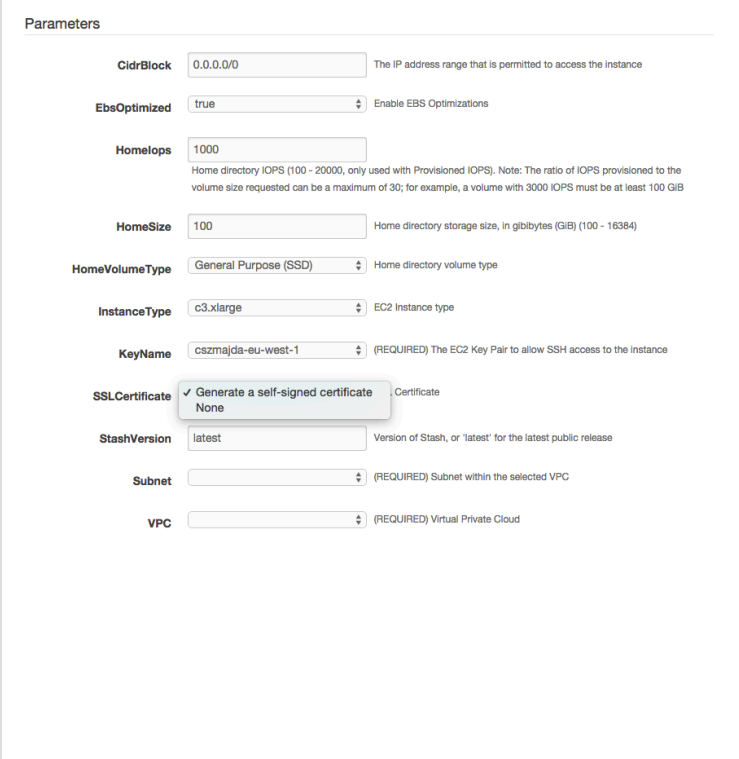
Set up a CloudFormation stack using the Atlassian Stash Server template.

To get started using Stash in AWS

1. Sign in to the AWS Console, and go to **Services > CloudFormation**.
2. Click **Create Stack**.
3. Select **Specify an Amazon S3 template URL field** and paste in

```
http://atlassian-software.s3.amazonaws.com/templates/stash/StashServer.template
```

4. When filling in Parameters, make sure you fill in the parameters appropriately for your organization.



CidrBlock: You can optionally restrict access to your instance to an IP address range in CIDR notation. NOTE: using 0.0.0.0/0 means unrestricted access.

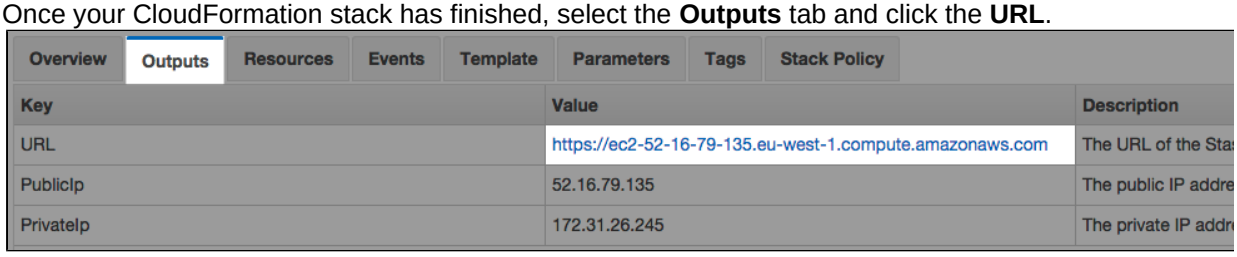
KeyName (REQUIRED): Make sure you have access to the private key file for the EC2 Key Pair you have selected. Without this file, you won't be able to SSH into your instance. See Creating an EC2 Key Pair.

SSLCertificate: You can optionally generate a self-signed SSL certificate, forcing all Web access to your instance to use HTTPS. See Installing an SSL certificate in your Stash instance.

VPC and Subnet (REQUIRED): Choose the right public or private VPC for your account, and make sure the Subnet is within the VPC.

See Securing Stash in AWS for more information about these options.

5. Once your CloudFormation stack has finished, select the **Outputs** tab and click the **URL**.



What's next?

Now you're ready to configure your Stash instance in AWS.

- Complete the Stash Setup Wizard, and begin using this like any other instance of Stash.
- Review and update your security settings for AWS.
- Migrate your existing Stash instance into AWS.
- Be sure to see the rest of the Administering Stash documentation.

Launching Stash in AWS manually

> This page describes how to launch the Atlassian Stash AMI manually, giving you complete control over the components enabled in the AMI and over AWS-specific configuration, network and security settings. If you are just looking for an automated way to spin up Stash in AWS, see Quick Start with Stash and AWS.

You can launch the Atlassian Stash AMI directly from the AWS Console, and running the EC2 launch wizard. See Launching EC2 Instances for detailed instructions.

On this page
- Finding the Atlassian Stash AMI
- Choosing an instance type
- Configure instance details
- Add storage
- Configure your Security Group
- What's next?

Finding the Atlassian Stash AMI

You can find the Atlassian Stash AMI by clicking **AWS Marketplace** and searching for **Atlassian Stash (2015.04.02_0403)**.

Be sure to use the correct AMI ID for your specific region. The following table lists the AMI ID of the Atlassian Stash AMI in each region.

Region Code	Region Name	AMI ID
`ap-northeast-1`	Asia Pacific (Tokyo)	ami-aa07fbaa
`ap-southeast-1`	Asia Pacific (Singapore)	ami-661d2e34
`ap-southeast-2`	Asia Pacific (Sydney)	ami-4d3a4877
`eu-central-1`	EU (Frankfurt)	ami-e47448f9
`eu-west-1`	EU (Ireland)	ami-1f781d68
`sa-east-1`	South America (São Paulo)	ami-27d9633a
`us-east-1`	US East (N. Virginia)	ami-a41a2bcc
`us-west-1`	US West (N. California)	ami-3d50b079
`us-west-2`	US West (Oregon)	ami-23ad8413

Choosing an instance type

When choosing an EC2 Instance type, see Recommendations for running Stash Server in AWS for recommended instance sizing.

> ⚠ ***Minimum hardware requirements***
>
> The default t2.micro (Free tier eligible), small, and medium instance types do not meet Stash's minimum hardware requirements, and are not supported for production deployments. See Recommendations for running Stash Server in AWS for the EC2 instance types supported by Stash.

Configure instance details

When configuring your EC2 instance these are some important details to consider.

IAM Role

It is recommended to launch your instance with an Identity and Access Management (IAM) Role that allows native AWS DIY Backup to run without explicit credentials. See IAM Roles for Amazon EC2 for more information.

From **Step 3: Configure Instance Details** of the EC2 Launch wizard, you can create a new IAM Role by clicking **Create new IAM role.** The role should contain at least the following policy:

```
{
    "Statement": [
        {
            "Resource": [
                "*"
            ],
            "Action": [
                "ec2:AttachVolume",
                "ec2:CreateSnapshot",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:DescribeSnapshots",
                "ec2:DescribeVolumes",
                "ec2:DetachVolume"
            ],
            "Effect": "Allow"
        }
    ],
    "Version": "2012-10-17"
}
```

> ⚠ ***IAM Role must be configured at launch time***
>
> An IAM Role can only be configured for your EC2 instance ***during initial launch***. You cannot associate an IAM role with a running EC2 instance ***after*** launch. See IAM Roles for more information.

Advanced Details

The Atlassian Stash AMI can be configured in a number of different ways at launch time:

- The built-in PostgreSQL and Nginx components (enabled by default) can be disabled,
- Self-signed SSL certificate generation (disabled by default) can be enabled.

You can control these options supplying User Data to your instance under **Advanced Details** in **Step 3: Configure Instance Details** of the EC2 launch wizard. All user-configurable behavior in the Atlassian Stash AMI can be controlled by creating a file `/etc/atl` containing shell variable definitions. On first boot, the Atlassian Stash AMI will source the file `/etc/atl` (if it exists), allowing its built-in default variable definitions to be overridden.

For example, to enable self-signed SSL certificate generation (and force all Web access to Stash to use HTTPS), you can add User Data (**As text**) as follows:

```
#!/bin/bash
echo "ATL_SSL_SELF_CERT_ENABLED=true" >>/etc/atl
```

For a complete list of variables that can be overridden in User Data at launch time, see Launching your Stash instance.

User Data is flexible and allows you to run arbitrary BASH commands on your instance at launch time, in

addition to overriding variables in `/etc/atl`. See Running Commands on Your Linux Instance at Launch for more information.

> ***Security considerations***
>
> See Securing Stash in AWS for more details about enabling HTTPS and self-signed certificates in the Atlassian Stash AMI.

Add storage

When attaching EBS volumes, use these storage device settings for your instance.

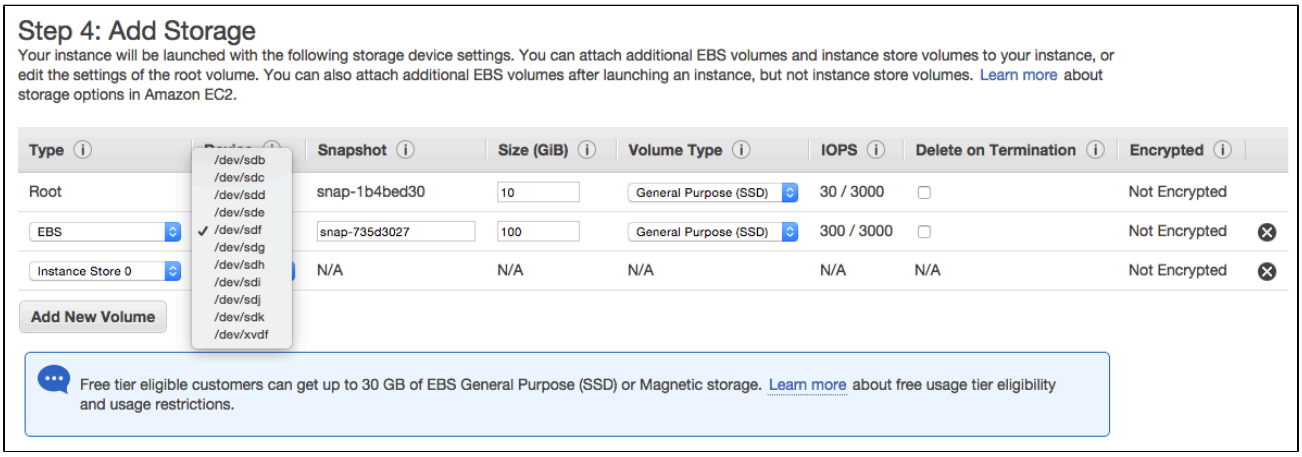
Type	Device	Purpose	Size (GiB)	Volume Type	IOPS	Delete on Termination
Root	`/dev/xvda`	Linux root volume	10	General Purpose (SSD)	30	No
EBS	`/dev/xvdf`	Stash data: repositories, attachments, avatars, etc.	100+	General Purpose (SSD) / Provisioned IOPS *	300+ *	No
Instance Store	`/dev/xvdb`	Stash temporary files and caches	N/A	N/A	N/A	N/A

* Provisioned IOPS with at least 500 – 1000 IOPS is recommended for instances with more than 500 active users. See Recommendations for running Stash Server in AWS for more information.

The Atlassian Stash AMI will not use any other block devices attached to the instance. The EBS volume for `/dev/xvdf` will be initialized and formatted at launch time, unless a snapshot id is provided (see the capture below in the page), in which case it will only format it if it's not already formatted. See Managing EBS Volumes for more information about storage options in Amazon EC2.

Attach an existing EBS snapshot

You can also attach an existing EBS volume based on a snapshot during launch. To attach an existing EBS volume, within the *Device* field, change the EBS volume device to `/dev/sdf` and enter the Snapshot ID of the snapshot.



See Administering Stash in AWS - Moving your Stash data volume between instances for more details.

Configure your Security Group

When configuring your Security Group, you must allow allow incoming traffic to all the following ports. For more information, see Using Security Groups.

Type	Protocol	Port	Description

SSH	TCP	22	SSH port, allowing access to administrative functions
HTTP	TCP	80	
HTTPS	TCP	443	
Custom TCP Rule	TCP	7999	Stash SSH port for Git hosting operations

What's next?

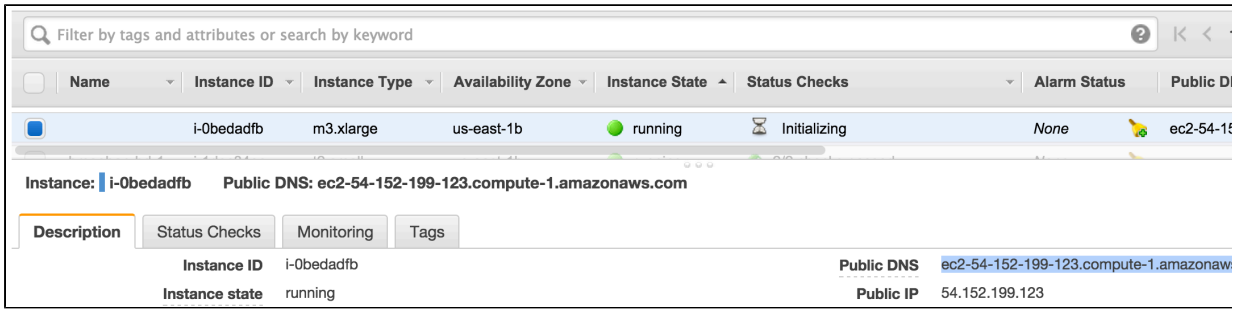
Now you're ready to configure your AWS version of Stash.

View your new instance

Once your new EC2 instance has launched, find it within the EC2 console and navigate to the URL provided so you can continue to configuring Stash.

To find the URL of your new EC2 instance

1. From within the EC2 Console, in the Description tab of your new instance, copy the **Public DNS**.



2. Paste the URL into a browser window to view start using Stash.

Set up your AWS instance of Stash

Once you've followed the URL of the EC2 instance you are presented with the Stash Setup Wizard.

Once you have launched Stash within AWS you can use it like any other Stash Server instance. So be sure to check out the rest of the Getting Started with Stash documentation.

Administering Stash in AWS

This page describes the Atlassian Stash Amazon Machine Image (AMI), what's inside it, how to launch it, and how to perform administration tasks on your Stash instance in the Amazon Web Services (AWS) environment.

On this page:

- The Stash AMI
- Components of the Stash AMI
- Launching your Stash instance
- Connecting to your Stash instance using SSH
- Installing an SSL certificate in your Stash instance
- Backing up your Stash instance
- Upgrading your Stash instance
- Stopping and starting your EC2 instance
- Migrating your existing Stash instance into AWS
- Resizing the data volume in your Stash instance
- Moving your Stash data volume between instances

The Stash AMI

The Atlassian Stash AMI provides a typical deployment of Stash in AWS. It bundles all the components used in a typical Stash deployment (reverse proxy, external database, backup tools, data volume, and temporary storage), pre-configured and ready to launch.

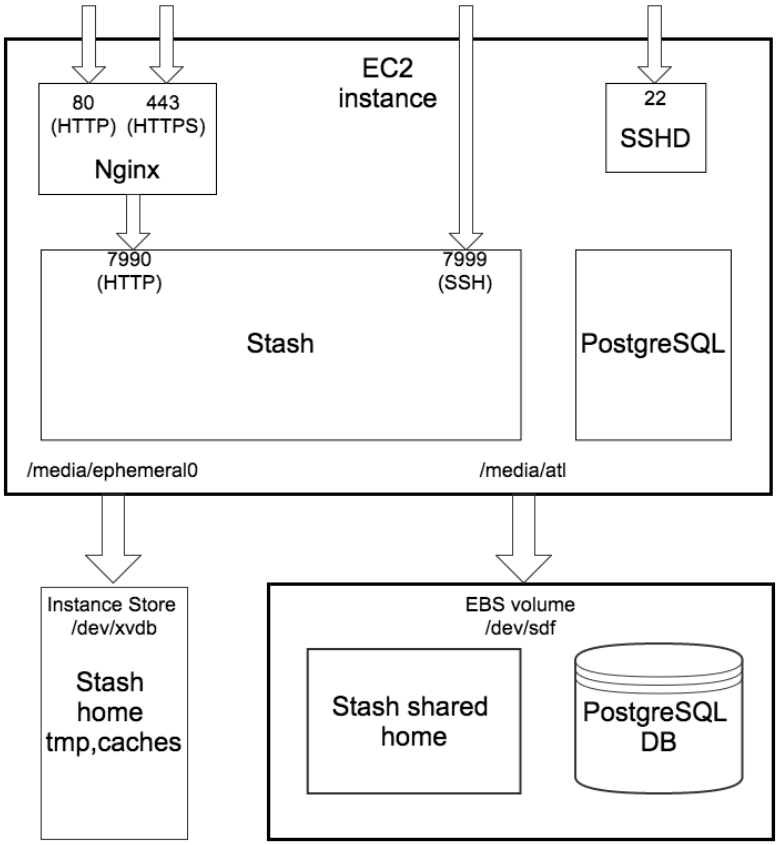
You can use the Atlassian Stash AMI as a "turnkey" deployment of a Stash instance in AWS, or use it as the starting point for customizing your own, more complex Stash deployments.

Components of the Stash AMI

An instance launched from the Atlassian Stash AMI contains the following

components:

- Stash (either the latest version or a version of your choice),
- an external PostgreSQL database,
- nginx as a reverse proxy,
- the Stash DIY Backup utilities pre-configured for native AWS snapshots,
- an EBS Volume and Instance Store to hold the data.



Operating system	Amazon Linux 64-bit, 2014.09.1
Stash	Stash (latest public version or a version of your choice) is downloaded and installed on launch.
Administrative tools	atlassian-stash-diy-backup pre-installed and configured for AWS native backup, accessible over SSH.

Reverse proxy	nginx, configured as follows: • listens on port 80 and (optionally) 443, • (optionally) terminates SSL (HTTPS) and passes through plain HTTP to Stash, • displays a static HTML page when the Stash service is not running.
Database	PostgreSQL 9.3
Block devices	1. An EBS volume (`/dev/xvdf`, mounted as /media/atl), that stores: • the Stash shared home directory, containing all of Stash's repository, attachment, and other data, • PostgreSQL's data directory. 2. An EC2 Instance Store (`/dev/xvdb`, mounted on /media/ephemeral0) to store Stash's temporary and cache files.

Launching your Stash instance

The Atlassian Stash AMI can be launched by either

- Using a CloudFormation template which automates creation of the associated Security Group and IAM Role, see Quick Start with Stash and AWS.
- Manually using the AWS Console which gives finer control over the optional components to enable in the instance and AWS-specific network, security, and block device settings, see Launching Stash in AWS manually.

On first boot, the Atlassian Stash AMI reads the file `/etc/atl` (if any), which can override variables that enable each of the installed components. So for example to enable a self-signed SSL certificate, you can supply user data to the instance at launch time like this:

```
#!/bin/bash
echo "ATL_SSL_SELF_CERT_ENABLED=true" >>/etc/atl
```

The following variables can be configured:

Variable name	Default value	Description
`ATL_NGINX_ENABLED`	`true`	Set to `false` to disable the Nginx reverse proxy, and leave Stash's `server.xml` configured to listen on port 7990 with no proxy.
`ATL_POSTGRES_ENABLED`	`true`	Set to `false` to disable the PostgreSQL service, and leave Stash configured with its internal HSQL database.
`ATL_SSL_SELF_CERT_ENABLED`	`false`	Set to `true` to enable a self-signed SSL certificate to be generated at launch time, and for Stash's `server.xml` and Nginx's `nginx.conf` to be configured for HTTPS. Requires `ATL_NGINX_ENABLED` also to be `true`.

See Proxying and securing Stash for more information about Stash's `server.xml` configuration file.

Connecting to your Stash instance using SSH

When connecting to your instance over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

The `ec2-user` has `sudo` access. The Atlassian Stash AMI does not allow SSH access by `root`.

Installing an SSL certificate in your Stash instance

If launched with a self-signed SSL certificate (you selected **SSLCertificate** Generate a self-signed certificate in Quick Start with Stash and AWS or you set `ATL_SSL_SELF_CERT_ENABLED=true` in Launching Stash in AWS manually), Stash will be configured to force HTTPS and redirect all plain HTTP requests to the equivalent `https://` URL.

It is highly recommended to replace this self-signed SSL certificate with a proper one for your domain, obtained from a Certification Authority (CA), at the earliest opportunity. See Securing Stash in AWS. Once you have a true SSL certificate, install it as soon as possible.

To replace the self-signed SSL certificate with a true SSL certificate

1. Place your certificate file at (for example) `/etc/nginx/ssl/my-ssl.crt`
2. Place your *password-less* certificate key file at `/etc/nginx/ssl/my-ssl.key`
3. Edit `/etc/nginx/nginx.conf` as follows:
    a. Replace references to `/etc/nginx/ssl/self-ssl.crt` with `/etc/nginx/ssl/my-ssl.crt`
    b. Replace references to `/etc/nginx/ssl/self-ssl.key` with `/etc/nginx/ssl/my-ssl.key`
4. Append the contents of `/etc/nginx/ssl/my-ssl.crt` to the default system PKI bundle (`/etc/pki/tls/certs/ca-bundle.crt`) to ensure scripts on the instance (such as DIY backup) can `curl` successfully.
5. Restart nginx.

Backing up your Stash instance

The Atlassian Stash AMI includes a complete set of Stash DIY Backup scripts which has been built specifically for AWS. For instructions on how to backup and restore your instance please refer to Using Stash DIY Backup in AWS.

Upgrading your Stash instance

To upgrade to a later version of Stash in AWS you first must connect to your instance using SSH, then follow the steps in the Stash upgrade guide.

Stopping and starting your EC2 instance

An EC2 instance launched from the Atlassian Stash AMI can be stopped and started just as any machine can be powered off and on again.

When stopping your EC2 instance, it is important to first

1. Stop the `atlstash` and `postgresql93`
2. Unmount the `/media/atl` filesystem.

If your EC2 instance becomes unavailable after stopping and restarting

When starting your EC2 instance back up again, if you rely on Amazon's automatically assigned public IP address (rather than a fixed private IP address or Elastic IP address) to access your instance, your IP address may have changed. When this happens, your instance can become inaccessible and display a "The host name for your Atlassian instance has changed" page. To fix this you need to update the hostname for your Stash instance.

To update the hostname for your Stash instance

1. Connect to your instance over SSH and run
   `sudo /opt/atlassian/bin/atl-update-host-name.sh`
2. Wait for Stash to restart.
3. If you have also set up Stash's Base URL to be the public DNS name or IP address you should also update Stash's base URL in the administration screen to reflect the change.

Migrating your existing Stash instance into AWS

Migrating an existing Stash instance to AWS involves moving consistent backups of your `${STASH_HOME}`

and your database to the AWS instance.

To migrate your existing Stash instance into AWS

1. Check for any known migration issues in the Stash Knowledge Base.
2. Alert users to the forthcoming Stash service outage.
3. Create a user in the Stash Internal User Directory with `SYSADMIN` permissions to the instance so you don't get locked out if the new server is unable to connect to your User Directory.
4. Take a backup of your instance with either the Stash Backup Client or the Stash DIY Backup.
5. Launch Stash in AWS using the Quick Start instructions, which uses a CloudFormation template.
6. Connect to your AWS EC2 instance with SSH and upload the backup file.
7. Restore the backup with the same tool used to generate it.
8. If necessary, update the JDBC configuration in the `${STASH_HOME}/shared/stash-config.properties` file.

Resizing the data volume in your Stash instance

By default, the application data volume in an instance launched from the Atlassian Stash AMI is a standard Linux ext4 filesystem, and can be resized using the standard Linux command line tools.

To resize the data volume in your Stash instance

1. Stop the `atlstash` and `postgresql93` services.
2. Unmount the `/media/atl` filesystem.
3. Create a snapshot of the volume to resize.
4. Create a new volume from the snapshot with the desired size, in the same availability zone as your EC2 instance.
5. Detach the old volume and attach the newly resized volume as `/dev/sdf`.
6. Resize `/dev/sdf` using `resize2fs`, verify that its size has changed, and remount it on `/media/atl`.
7. Start the `postgresql93` and `atlstash` and services.

For more information, see Expanding the Storage Space of an EBS Volume on Linux, Expanding a Linux Partition, and the Linux manual pages for `resize2fs` and related commands.

Moving your Stash data volume between instances

Occasionally, you may need to move your Stash data volume to another instance–for example, when setting up staging or production instances, or when moving to an instance to a different availability zone.

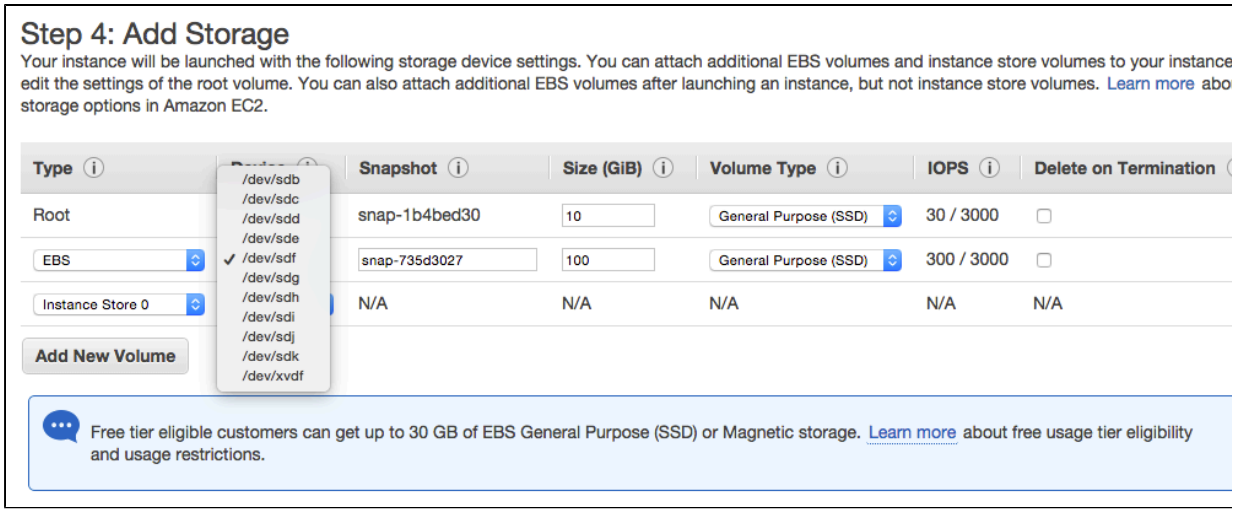
There are two approaches to move your Stash data volume to another instance

1. Take a backup of your data volume with Stash DIY Backup, and restore it on your new instance. See Using Stash DIY Backup in AWS for this option.
2. Launch a new instance from the Atlassian Stash AMI with a snapshot of your existing data volume.

> A Stash data volume may only be moved to a Stash instance of the same or higher version than the original.

To launch a new instance from the Stash AMI using a snapshot of your existing Stash data volume

1. Stop the `atlstash` and `postgresql93` services on your existing Stash instance.
2. Unmount the `/media/atl` filesystem.
3. Create a snapshot of the Stash data volume (the one attached to the instance as `/dev/sdf`).
4. Once the snapshot generation has completed, launch a new instance from the Atlassian Stash AMI as described in Launching Stash in AWS manually. When adding storage, change the EBS volume device to `/dev/sdf` as seen below and enter the id of the created snapshot.



5. If the host name (private or public) that users use to reach your Stash instance has changed as a result of moving availability zones (or as a result of stopping an instance and starting a new one) you will need to SSH in and run
   `sudo /opt/atlassian/bin/atl-update-host-name.sh` *<newhostname>*
   where *<newhostname>* is the new host name.
6. Once Stash has restarted your new instance should be fully available.
7. If the host name has changed you should also update the JDBC URL configuration in the `stash-configuration.properties` file (typically located in `/var/atlassian/application-data/stash/shared/`), as well as Stash's base URL in the administration screen to reflect this.

Recommendations for running Stash Server in AWS

To get the best performance out of your Stash deployment in AWS, it's important not to under-provision your instance's CPU, memory, or I/O resources. Note that the very smallest instance types provided by AWS do not meet Stash's minimum hardware requirements and are not recommended in production environments. If you do not provision sufficient resources for your workload, Stash may exhibit slow response times, display a Stash is reaching resource limits banner, or fail to start altogether.

On this page

- Recommended EC2 and EBS instance sizes
- Other supported instance sizes
- Advanced: Monitoring your Stash instance to tune instance sizing

Recommended EC2 and EBS instance sizes

The following table lists the recommended EC2 and EBS configurations for Stash Server in AWS under typical workloads.

Active Users	EC2 instance type	EBS Optimized	EBS Volume type	IOPS
0 – 250	c3.large	No	General Purpose (SSD)	N/A
250 – 500	c3.xlarge	Yes	General Purpose (SSD)	N/A

500 – 1000	c3.2xlarge	Yes	Provisioned IOPS	500 – 1000

In Stash instances with high hosting workload, I/O performance is often the limiting factor. It is recommended to pay particular attention to EBS volume options, especially the following:

- The size of an EBS volume also influences I/O performance. Larger EBS volumes generally have a larger slice of the available bandwidth and I/O operations per second (IOPS). A minimum of 100 GiB is recommended in production environments.
- The IOPS that can be sustained by General Purpose (SSD) volumes is limited by Amazon's I/O credits. If you exhaust your I/O credit balance, your IOPS will be limited to the baseline level. You should consider using a larger General Purpose (SSD) volume or switching to a Provisioned IOPS (SSD) volume. See Amazon EBS Volume Types for more information.
- New EBS volumes in particular have reduced performance the first time each block is accessed. See Pre-Warming Amazon EBS Volumes for more information.

The above recommendations are based on a ***typical*** workload with the specified number of active users. The resource requirements of an actual Stash instance may vary with a number of factors, including:

- The number of continuous integration servers cloning or fetching from Stash: Stash will use more resources if you have many build servers set to clone or fetch frequently from Stash.
- Whether continuous integration servers are using push mode notifications or polling repositories regularly to watch for updates.
- Whether continuous integration servers are set to do full clones or shallow clones.
- Whether the majority of traffic to Stash is over HTTP, HTTPS, or SSH, and the encryption ciphers used.
- The number and size of repositories: Stash will use more resources when you work on many very large repositories.
- The activity of your users: Stash will use more resources if your users are actively using the Stash web interface to browse, clone and push, and manipulate Pull Requests.
- The number of open Pull Requests: Stash will use more resources when there are many open Pull Requests, especially if they all target the same branch in a large, busy repository.

See Scaling Stash and Scaling Stash for Continuous Integration performance for more detailed information on Stash resource requirements.

Other supported instance sizes

The following Amazon EC2 instances also meet or exceed Stash's minimum hardware requirements. These instances provide different balances of CPU, memory, and I/O performance, and can cater for workloads that are more CPU-, memory-, or I/O-intensive than the typical.

Model	vCPU	Mem (GiB)	Instance Store (GB)	EBS optimizations available	Dedicated EBS Throughput (Mbps)
c3.large	2	3.75	2 x 16 SSD	-	-
c3.xlarge	4	7.5	2 x 40 SSD	Yes	-
c3.2xlarge	8	15	2 x 80 SSD	Yes	-
c3.4xlarge	16	30	2 x 160 SSD	Yes	-
c3.8xlarge	32	60	2 x 320 SSD	-	-
c4.large	2	3.75	-	Yes	500
c4.xlarge	4	7.5	-	Yes	750
c4.2xlarge	8	15	-	Yes	1,000
c4.4xlarge	16	30	-	Yes	2,000
c4.8xlarge	36	60	-	Yes	4,000

hs1.8xlarge	16	117	24 x 2000	-	-
i2.xlarge	4	30.5	1 x 800 SSD	Yes	-
i2.2xlarge	8	61	2 x 800 SSD	Yes	-
i2.4xlarge	16	122	4 x 800 SSD	Yes	-
i2.8xlarge	32	244	8 x 800 SSD	-	-
m3.large	2	7.5	1 x 32 SSD	-	-
m3.xlarge	4	15	2 x 40 SSD	Yes	-
m3.2xlarge	8	30	2 x 80 SSD	Yes	-
r3.large	2	15.25	1 x 32 SSD	-	-
r3.xlarge	4	30.5	1 x 80 SSD	Yes	-
r3.2xlarge	8	61	1 x 160 SSD	Yes	-
r3.4xlarge	16	122	1 x 320 SSD	Yes	-
r3.8xlarge	32	244	2 x 320 SSD	-	-

In all AWS instance types, Stash only supports "large" and higher instances. "Micro", "small", and "medium" sized instances do not meet Stash's minimum hardware requirements and are not recommended in production environments.

Stash does not support D2 instances, Burstable Performance (T2) Instances, or Previous Generation Instances.

In any instance type with available Instance Store device(s), a Stash instance launched from the Atlassian Stash AMI will configure one Instance Store to contain Stash's temporary files and caches. Instance Store can be faster than an EBS volume but the data does not persist if the instance is stopped or rebooted. Use of Instance Store can improve performance and reduce the load on EBS volumes. See Amazon EC2 Instance Store for more information.

Advanced: Monitoring your Stash instance to tune instance sizing

> This section is for advanced users who wish to monitor the resource consumption of their instance and use this information to guide instance sizing.

The above recommendations provide guidance for ***typical*** workloads. The resource consumption of every Stash instance, though, will vary with the mix of workload. The most reliable way to determine if your Stash instance is under- or over-provisioned in AWS is to monitor its resource usage regularly with Amazon CloudWatch. This provides statistics on the actual amount of CPU, I/O, and network resources consumed by your Stash instance.

The following simple example BASH script uses

- the AWS CLI,
- gnuplot,
- jq

to gather CPU, I/O, and network statistics and display them in a simple chart that can be used to guide your instance sizing decisions.

Click here to expand...

```
#!/bin/bash
# Example AWS CloudWatch monitoring script
# Usage:
#   (1) Install gnuplot and jq (minimum version 1.4)
#   (2) Install AWS CLI
```

```
(http://docs.aws.amazon.com/cli/latest/userguide/installing.html) and
configure it with
#       credentials allowing cloudwatch get-metric-statistics
#   (3) Replace "xxxxxxx" in volume_ids and instance_ids below with the ID's
of your real instance
#   (4) Run this script

export start_time=$(date -v-14d +%Y-%m-%dT%H:%M:%S)
export end_time=$(date +%Y-%m-%dT%H:%M:%S)
export period=1800
export volume_ids="vol-xxxxxxxx"    # REPLACE THIS WITH THE VOLUME ID OF YOUR
REAL EBS VOLUME
export instance_ids="i-xxxxxxxx"    # REPLACE THIS WITH THE INSTANCE ID OF
YOUR REAL EC2 INSTANCE

# Build lists of metrics and datafiles that we're interested in
ebs_metrics=""
ec2_metrics=""
cpu_datafiles=""
iops_datafiles=""
queue_datafiles=""
net_datafiles=""
for volume_id in ${volume_ids}; do
  for metric in VolumeWriteOps VolumeReadOps; do
    ebs_metrics="${ebs_metrics} ${metric}"
    iops_datafiles="${iops_datafiles} ${volume_id}-${metric}"
  done
done
for volume_id in ${volume_ids}; do
  for metric in VolumeQueueLength; do
    ebs_metrics="${ebs_metrics} ${metric}"
    queue_datafiles="${queue_datafiles} ${volume_id}-${metric}"
  done
done
for instance_id in ${instance_ids}; do
  for metric in DiskWriteOps DiskReadOps; do
    ec2_metrics="${ec2_metrics} ${metric}"
    iops_datafiles="${iops_datafiles} ${instance_id}-${metric}"
  done
done
for instance_id in ${instance_ids}; do
  for metric in CPUUtilization; do
    ec2_metrics="${ec2_metrics} ${metric}"
    cpu_datafiles="${cpu_datafiles} ${instance_id}-${metric}"
  done
done
for instance_id in ${instance_ids}; do
  for metric in NetworkIn NetworkOut; do
    ec2_metrics="${ec2_metrics} ${metric}"
    net_datafiles="${net_datafiles} ${instance_id}-${metric}"
  done
done

# Gather the metrics using AWS CLI
for volume_id in ${volume_ids}; do
  for metric in ${ebs_metrics}; do
    aws cloudwatch get-metric-statistics --metric-name ${metric} \
                                         --start-time ${start_time} \
                                         --end-time ${end_time} \
                                         --period ${period} \
                                         --namespace AWS/EBS \
                                         --statistics Sum \
                                         --dimensions
```

```
Name=VolumeId,Value=${volume_id} | \
      jq -r '.Datapoints | sort_by(.Timestamp) | map(.Timestamp + " " + (.Sum
| tostring)) | join("\n")' >${volume_id}-${metric}.data
  done
done

for metric in ${ec2_metrics}; do
  for instance_id in ${instance_ids}; do
    aws cloudwatch get-metric-statistics --metric-name ${metric} \
                                         --start-time ${start_time} \
                                         --end-time ${end_time} \
                                         --period ${period} \
                                         --namespace AWS/EC2 \
                                         --statistics Sum \
                                         --dimensions
Name=InstanceId,Value=${instance_id} | \
      jq -r '.Datapoints | sort_by(.Timestamp) | map(.Timestamp + " " + (.Sum
| tostring)) | join("\n")' >${instance_id}-${metric}.data
  done
done

cat >aws-monitor.gnuplot <<EOF
set term pngcairo font "Arial,30" size 1600,900
set title "IOPS usage"
set datafile separator whitespace
set xdata time
set timefmt "%Y-%m-%dT%H:%M:%SZ"
set grid
set ylabel "IOPS"
set xrange ["${start_time}Z":"${end_time}Z"]
set xtics "${start_time}Z",86400*2 format "%d-%b"
set output "aws-monitor-iops.png"
plot \\
EOF
for datafile in ${iops_datafiles}; do
  echo "  \"${datafile}.data\" using 1:(\$2/${period}) with lines title
\"${datafile}\", \\" >>aws-monitor.gnuplot
done

cat >>aws-monitor.gnuplot <<EOF

set term pngcairo font "Arial,30" size 1600,900
set title "IO Queue Length"
set datafile separator whitespace
set xdata time
set timefmt "%Y-%m-%dT%H:%M:%SZ"
set grid
set ylabel "Queue Length"
set xrange ["${start_time}Z":"${end_time}Z"]
set xtics "${start_time}Z",86400*2 format "%d-%b"
set output "aws-monitor-queue.png"
plot \\
EOF
for datafile in ${queue_datafiles}; do
  echo "  \"${datafile}.data\" using 1:2 with lines title \"${datafile}\", \\"
>>aws-monitor.gnuplot
done

cat >>aws-monitor.gnuplot <<EOF

set term pngcairo font "Arial,30" size 1600,900
set title "CPU Utilization"
set datafile separator whitespace
```

```
set xdata time
set timefmt "%Y-%m-%dT%H:%M:%SZ"
set grid
set ylabel "%"
set xrange ["${start_time}Z":"${end_time}Z"]
set xtics "${start_time}Z",86400*2 format "%d-%b"
set output "aws-monitor-cpu.png"
plot \\
EOF
for datafile in $cpu_datafiles; do
  echo "  \"${datafile}.data\" using 1:2 with lines title \"${datafile}\", \\"
>>aws-monitor.gnuplot
done

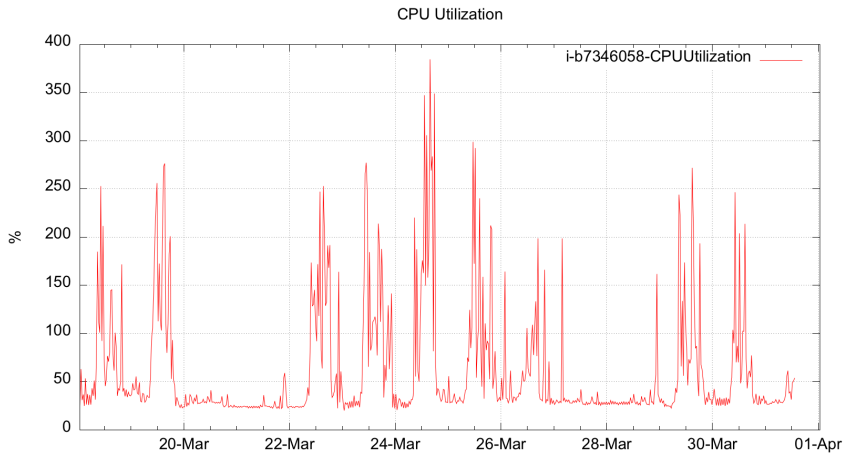
cat >>aws-monitor.gnuplot <<EOF

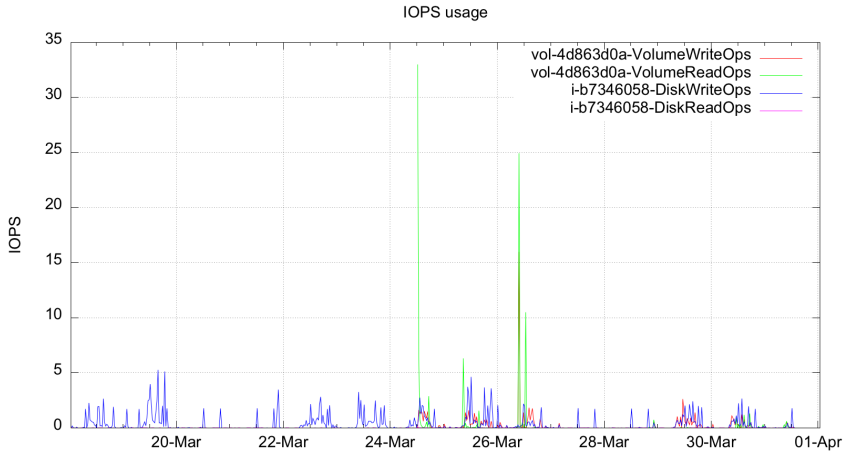
set term pngcairo font "Arial,30" size 1600,900
set title "Network traffic"
set datafile separator whitespace
set xdata time
set timefmt "%Y-%m-%dT%H:%M:%SZ"
set grid
set ylabel "MBytes/s"
set xrange ["${start_time}Z":"${end_time}Z"]
set xtics "${start_time}Z",86400*2 format "%d-%b"
set output "aws-monitor-net.png"
plot \\
EOF
for datafile in $net_datafiles; do
  echo "  \"${datafile}.data\" using 1:(\$2/${period}/1000000) with lines
title \"${datafile}\", \\" >>aws-monitor.gnuplot
```

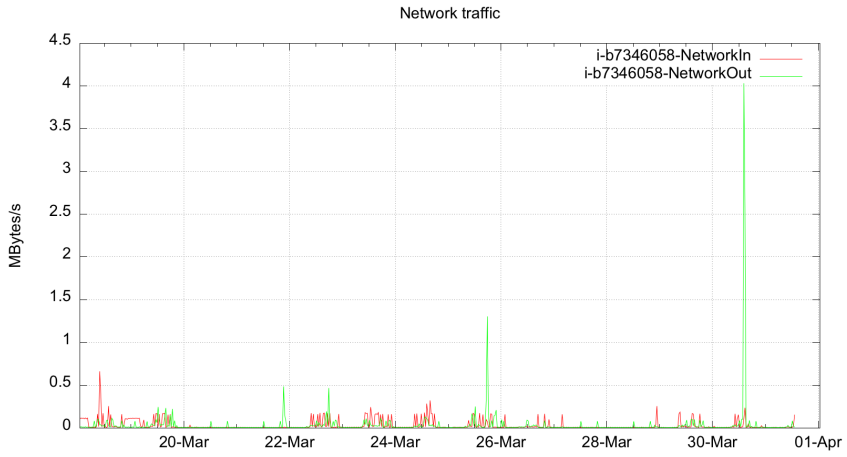
```
done

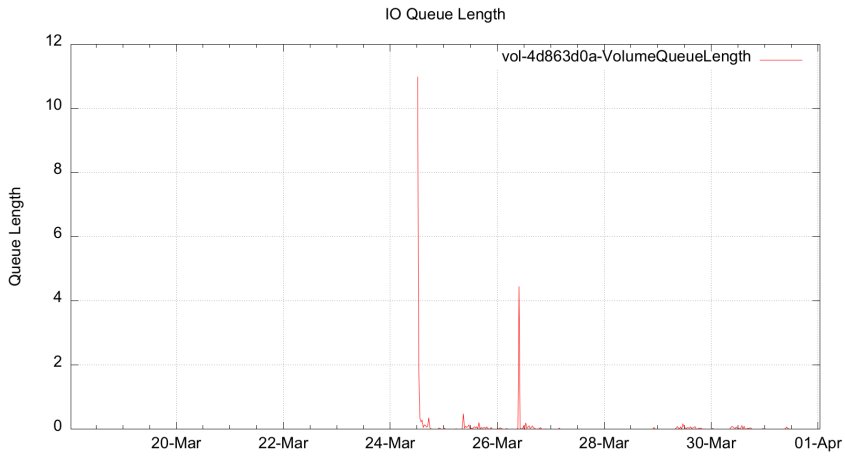
gnuplot <aws-monitor.gnuplot
```

When run on a typical Stash instance, this script produces charts such as the following:









You can use the information in charts such as this to decide whether CPU, network, or I/O resources are over- or under-provisioned in your instance.

If your instance is frequently saturating the maximum available CPU (taking into account the number of cores in your instance size), then this may indicate you need an EC2 instance with a larger CPU count. (Note that the CPU utilization reported by Amazon CloudWatch for smaller EC2 instance sizes may be influenced to some extent by the "noisy neighbor" phenomenon, if other tenants of the Amazon environment consume CPU cycles from the same physical hardware that your instance is running on.)

If your instance is frequently exceeding the IOPS available to your EBS volume and/or is frequently queuing I/O requests, then this may indicate you need to upgrade to an EBS optimized instance and/or increase the Provisioned IOPS on your EBS volume. See EBS Volume Types for more information.

If your instance is frequently limited by network traffic, then this may indicate you need to choose an EC2 instance with a larger available slice of network bandwidth.

Securing Stash in AWS

This page describes security best practices for running and maintaining Stash in AWS.

On this page:

- Amazon Virtual Private Cloud (VPC) and Subnets
- Security Groups
- HTTPS
- Self-signed SSL certificates
- Domain name
- Trusted CA-issued certificates
- Keeping your system up-to-date

Amazon Virtual Private Cloud (VPC) and Subnets

Amazon VPC enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. See Amazon EC2 and Amazon Virtual Private Cloud for more information.

A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a subnet that you select. Use a public subnet for resources that must be connected to the Internet, and a private subnet for resources that won't be connected to the Internet.

See Amazon's article called Your VPC and Subnets for a general overview of VPCs and subnets.

To bolster the security of your VPC you may wish to enable one or more of the following:

- Secure your VPC with a firewall virtual appliance / AMI to defend against unauthorised network activity
- Configure a site-to-site VPN to ensure information is transferred securely between Stash and its users
- Configure an intrusion prevention or intrusion detection virtual appliance to detect when unauthorised network activity has occurred
- Enable Amazon CloudTrail to log VPC API operations and keep an audit trail of network changes

Security Groups

A security group acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances. You can modify the rules for a security group at any time; the new rules are automatically applied to all instances that are associated with the security group. See Amazon EC2 Security Groups for Linux Instances for more information.

We recommend you restrict the security groups that apply to the Stash instance to the absolute minimum required. As an instance can have up to a hundred security groups applied to it, it can be difficult to understand which restrictions are in effect. It is for this reason we recommend you condense the applied security groups to as few as possible.

HTTPS

By default, the Stash AMI configures Stash to serve requests over HTTP not HTTPS. If you are not connected to the AWS VPC your Stash resides in via a Virtual Private Gateway, then all passwords and data will be sent unencrypted over the public Internet. If you intend for your Stash instance to be Internet facing, setting `ATL_SSL_SELF_CERT_ENABLED=true` is recommended to enable HTTPS to your instance at launch time.

Self-signed SSL certificates

If HTTPS is enabled via ATL_SSL_SELF_CERT_ENABLED=true a self-signed certificate will be generated for your Statsh instance.

If you continue to use the self-signed certificate:

- most browsers will display security warnings that must be ignored before proceeding to the Stash Web interface,
- Git clients will refuse to connect to Stash over HTTPS unless configured to ignore the self-signed certificate with `git config --global http.sslVerify false`, and
- Application links and/or integrations with other applications that use Stash's REST API and do not accept self-signed certificates may fail.

The self-signed certificate should be replaced with a certificate obtained from a trusted certificate authority (CA) at the earliest opportunity to improve your security and improve the experience of your users.

Domain name

In order to use a trusted CA-issued certificate with your Stash instance and to avoid the problems outlined above with self-signed certificates you will first need a static public domain name associated with your instance. Amazon Route 53 and other DNS providers can provide you with this. You will need to ensure you update your DNS record every time your EC2 instance's IP address changes. Using Amazon's Elastic IP Address helps minimise the IP address changes of your instance and thus minimise its day-to-day administration.

Trusted CA-issued certificates

Once you have a static domain name for your EC2 instance you can request a trusted certificate authority issue a certificate for use with this domain / instance. Installing the certificate is a straight-forward process as long as you first set up your instance to use a self-signed certificate.

Keeping your system up-to-date

It is essential to keep your Stash instance up-to-date with patches and updates to maximise security and minimise opportunity for exploits and misadventure. On first boot a Stash AMI instance will download the latest official release of Stash at that time so you are assured of having the very latest version of Stash when you first start using Stash in AWS.

> Please be sure to always perform a backup of your instance before attempting any update.

Amazon Linux Security Updates

The Stash AMI is based on Amazon Linux and the latest version of this is used whenever we cut a new release of the Stash AMI. Occasionally vulnerabilities in libraries and utilities used in Amazon Linux will be

detected and updates posted in the Amazon Linux AMI yum repository. Atlassian will issue new versions of the Stash AMI where necessary to ensure new Stash AWS instances start with these updates but if you are managing an existing instance you may need to apply these updates yourself. By default, Amazon Linux applies all security updates on reboot. Alternatively you can run "yum update --security".

From time-to-time you may also wish to apply other updates from the Amazon Linux AMI yum repository to your Stash instance. You must ensure that any updated packages are supported by the version of Stash you are running. Stash version requirements can always be found on the Support Platform page.

Stash Updates

The Atlassian Stash team have a strong release cadence and routinely issue releases including new features, performance and security fixes. It is strongly recommended you keep Stash as up to date as possible. To update Stash in an existing instance please follow the Stash Upgrade Guide.

Using Stash DIY Backup in AWS

This page describes how to execute a DIY Backup and Restore of a Stash instance deployed in AWS.

On this page

- About the Stash DIY Backup for AWS
- Configure the Stash DIY Backup script
- Back up your instance
- Restore your instance
- Advanced configuration

About the Stash DIY Backup for AWS

Stash DIY Backup for AWS leverages AWS infrastructure to backup and restore Stash. The provided scripts take EBS snapshots of the volume containing the Stash shared home directory and the database data directory. These snapshots can later be used to create a new volume and attach it to your instance thus restoring Stash to a specific point in time.

Other approaches include using the Stash Backup Client or manually running the same steps as the Stash DIY Backup for AWS yourself. The benefits of using the Stash DIY Backup over these approaches are:

- taking AWS native snapshots are faster than filesystem level copying
- downtime is kept to an absolute minimum
- snapshots are stored with the redundancy and durability of S3
- it makes it easy to relocate an instance to a different Availability Zone in the future

The scripts use the AWS CLI toolset, which is included in all instances launched from the AMI, regardless of launch method. The template creates an IAM role with a policy that grants the instance the permissions required to backup and restore the EBS volume. See the Advanced configuration section for an example policy with similar permissions.

Configure the Stash DIY Backup script

The provided script, `stash.diy-aws-backup.vars.sh`, comes with sensible defaults for your AWS environment, but you need to modify variables in the script for your specific setup. These variables indicate how to lock Stash for backup, to ensure consistency by preventing writes to the volume.

To locate the Stash DIY Backup scripts available in the default installation directory:

1. Connect to your Stash instance on AWS over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

2. Go to the `stash-diy-backup` directory.

```
cd /opt/atlassian/stash-diy-backup
```

3. Pull the latest version of the scripts. The installation directory contains a clone of the Stash DIY Backup scripts repository. You can update to the latest changes at any time.

```
git pull
```

4. Locate the variables script.

```
stash.diy-aws-backup.vars.sh
```

5. Modify the variables within the script for your specific instance.

Variables

These variables must be customized.

Variable	Explanation
STASH_BACKUP_USER	The username of a Stash user with the SYSADMIN role.
STASH_BACKUP_PASS	The password to the STASH_BACKUP_USER account.

> Users who have attached their home directory volume to a device name other than `/dev/xvdf` may also need to update the HOME_DIRECTORY_DEVICE_NAME variable. See the Advanced configuration section for more information.

Back up your instance

To back up your Stash instance within AWS you need to run the `stash.diy-aws-backup.sh` script. This will take the values you configured above to perform the backup.

To run the Stash DIY Backup scripts

1. Connect to your Stash instance on AWS over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

2. Go to the `stash-diy-backup` directory.

```
cd /opt/atlassian/stash-diy-backup
```

3. Run the `stash.diy-aws-backup.sh`.

```
./stash.diy-aws-backup.sh
```

Running the scripts will produce an output similar to this:

Click here to see example output...

```
[http://localhost:7990]  INFO: Using IAM instance role
Stash-DIY-Backup-StashBackupRole-1GJ4BL2XHPDBG
[http://localhost:7990]  INFO: Looking up volume for device name /dev/sdf
[http://localhost:7990]  SUCC: Found volume vol-aae66abd for device name
/dev/sdf
[http://localhost:7990]  INFO: locked with
'603c1d2c41121d1b6f42c0b03d4e03ee8a22577b'
[http://localhost:7990]  INFO: backup started with
'a62d9b002747877e57d7ff32cdaedc92ba66db79'
[http://localhost:7990]  INFO: Waiting for DRAINED state.. done
[http://localhost:7990]  INFO: db state 'DRAINED'
[http://localhost:7990]  INFO: scm state 'DRAINED'
[http://localhost:7990]  INFO: Backup progress updated to 50
[http://localhost:7990]  INFO: Freezing filesystem at mount point /media/atl
[http://localhost:7990]  INFO: Performing backup of home directory
[http://localhost:7990]  SUCC: Taken snapshot snap-2f133304 of volume
vol-aae66abd
[http://localhost:7990]  INFO: Tagged snap-2f133304 with
Name=stash-20150326-013036-405
[http://localhost:7990]  INFO: Unfreezing filesystem at mount point /media/atl
[http://localhost:7990]  INFO: Backup progress updated to 100
[http://localhost:7990]  INFO: Stash instance unlocked
[http://localhost:7990]  SUCC: Successfully completed the backup of your Stash
instance
[http://localhost:7990]  INFO: Cleaning up...
[http://localhost:7990]  INFO: Unfreezing filesystem at mount point /media/atl
```

You can review the newly created snapshot in the AWS EC2 console.

Restore your instance

Restoring your instance is done by replacing the existing volume with a new one created using an existing EBS snapshot.

To restore your Stash instance using the Stash DIY Restore scripts

1. Connect to your Stash instance on AWS over SSH, use `ec2-user` as the user name, for example:

   ```
   ssh -i keyfile.pem ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
   ```

2. Go to the `stash-diy-backup` directory.

   ```
   cd /opt/atlassian/stash-diy-backup
   ```

3. Stop your Stash instance.

   ```
   sudo /etc/init.d/atlstash stop
   ```

4. Stop your database (if available).

   ```
   sudo /etc/init.d/postgresql93 stop
   ```

5. Unmount any volumes using the configured mount point.

```
sudo umount /media/atl
```

6. Detach any volumes using the `/dev/sdf` device name. These commands export the instance region for AWS CLI tools, retrieve the ID for the volume attached to the device name, and detach the volume from the instance.

```
export AWS_DEFAULT_REGION=`curl -s -f
http://169.254.169.254/latest/meta-data/placement/availability-zone | sed
-e 's:\([0-9][0-9]*\)[a-z]*\$:\\1:'`
DETACH_VOLUME_ID=$(aws ec2 describe-volumes --filter
Name=attachment.instance-id,Values=`curl -s -f
http://169.254.169.254/latest/meta-data/instance-id`
Name=attachment.device,Values=/dev/xvdf | jq -r '.Volumes[0].VolumeId')
aws ec2 detach-volume --volume-id ${DETACH_VOLUME_ID}
```

> The previous commands assume the home directory volume is attached to device name `/dev/xvdf`

7. All snapshot taken using the DIY backup script (see above) will be tagged with key "Name" and value which includes the configured `INSTANCE_NAME` and a timestamp for when the backup was taken. To see which snapshots are available for your `INSTANCE_NAME`, run the script without any arguments.

```
./stash.diy-aws-restore.sh
```

The snapshots are sorted alphabetically to keep the latest snapshots at the top of the list.

Click here to see example output...

```
[http://localhost:7990]  INFO: Using IAM instance role
Stash-DIY-Backup-StashBackupRole-1GJ4BL2XHPDBG
[http://localhost:7990]  INFO: Usage: ./stash.diy-aws-restore.sh
<snapshot-tag>
Available snapshot tags:
stash-20150327-013036-405
stash-20150326-234633-252
stash-20150326-231048-106
stash-20150326-231038-587
stash-20150326-224449-846
stash-20150326-224012-288
stash-20150326-222700-117
stash-20150326-222522-962
stash-20150326-220256-274
stash-20150326-083409-384
```

8. After you select a tag, run the script again with the instance tag as an argument.

```
./stash.diy-aws-restore.sh stash-20150326-013036-405
```

Click here to see example output...

```
[http://localhost:7990]  INFO: Using IAM instance role
Stash-DIY-Backup-StashBackupRole-1GJ4BL2XHPDBG
[http://localhost:7990]  INFO: Restoring from tag
stash-20150326-013036-405
[http://localhost:7990]  INFO: Checking for existing volumes using
device name /dev/sdf
[http://localhost:7990]  INFO: Found EBS snapshot snap-2f133304 for tag
stash-20150326-013036-405
[http://localhost:7990]  INFO: Restoring home directory from snapshot
snap-2f133304 into a gp2 volume
[http://localhost:7990]  SUCC: Restored snapshot snap-2f133304 into
volume vol-dae16dcd
[http://localhost:7990]  INFO: Waiting for volume vol-dae16dcd to be
attached. This could take some time
[http://localhost:7990]  INFO: Volume vol-dae16dcd state: attaching
[http://localhost:7990]  INFO: Volume vol-dae16dcd state: attached
[http://localhost:7990]  SUCC: Attached volume vol-dae16dcd to device
/dev/sdf at instance i-6acb8a8e
[http://localhost:7990]  SUCC: Mounted device /dev/sdf to /media/atl
[http://localhost:7990]  INFO: Performed restore of home directory
snapshot
[http://localhost:7990]  SUCC: Successfully completed the restore of
your Stash instance
```

You can review the newly created volume in the AWS EC2 console.

9. Start your instance.

```
sudo /etc/init.d/postgresql93 start
sudo /etc/init.d/atlstash start
```

Advanced configuration

Variable	Explanation
INSTANCE_NAME	A name for your instance. It cannot contain spaces. It must be under 100 characters in length. This will be used as a prefix when tagging your instance snapshot.
STASH_URL	The Stash base URL. It must not end with '/'
STASH_HOME	The path to the Stash home directory. For example `/var/atlassian/application-data/stash`
BACKUP_DATABASE_TYPE	'`ebs-collocated`' means the database data directory is located in the same volume as the STASH_HOME.
BACKUP_HOME_TYPE	'`ebs-home`' means the Stash home (and the database data directory if ebs-collocated has been specified) will be backed up by taking a snapshot of an EBS volume.
HOME_DIRECTORY_MOUNT_POINT	The mount point for the volume holding the STASH_HOME directory as it appears in `/etc/fstab` (for example `/media/atl`)

HOME_DIRECTORY_DEVICE_NAME	The device name on to which the STASH_HOME volume is attached as it appears in the Amazon console (for example /dev/sdf)
AWS_AVAILABILITY_ZONE	The availability zone for your AWS instance. If left unchanged from the template it will be retrieved from the metadata endpoint
AWS_REGION	The region for your AWS instance. If left unchanged from the template it will be derived from the AWS_AVAILABILITY_ZONE
RESTORE_HOME_DIRECTORY_VOLUME_TYPE	The type of volume to create from the snapshot (one of io1, gp2, and standard)
RESTORE_HOME_DIRECTORY_IOPS	The provisioned IOPS for the new volume. Only necessary if RESTORE_HOME_DIRECTORY_VOLUME_TYPE is 'io1'
HIPCHAT_URL	The url for the HipChat API
HIPCHAT_ROOM	The HipChat room to which notifications should be delivered
HIPCHAT_TOKEN	The authentication token for the HipChat API
CURL_OPTIONS	A set of options to pass to the curl commands executed by the scripts
STASH_VERBOSE_BACKUP	If FALSE info and print level logging will be skipped

Setting up the instance role

The DIY backup and restore scripts use the AWS CLI toolset to do their job. These tools need to authenticate with AWS in order to gain access to your resources (EBS volumes, snapshots, etc). The recommended way of providing credentials to the instance is by launching it with an instance role that has a suitable policy attached. If you are using the Stash CloudFormation template, it will take care of creating a policy for you and attach it to the instance at launch time.

If you need to create your own policy, you can use this JSON object as an example of the minimum permissions required for an instance:

```
{
    "Statement": [
        {
            "Resource": [
                "*"
            ],
            "Action": [
                "ec2:AttachVolume",
                "ec2:CreateSnapshot",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:DescribeSnapshots",
                "ec2:DescribeVolumes",
    "ec2:DetachVolume"
            ],
            "Effect": "Allow"
        }
    ],
    "Version": "2012-10-17"
}
```

For other ways of configuring the AWS CLI toolset, please refer to the documentation.

Releases

The following pages can be found in the latest documentation for Stash:

- the Stash upgrade guide
- the Stash security advisories
- the End of support announcements for Stash
- the full release notes for every Stash release.

You can get automated notifications about major and minor Stash releases by subscribing to the Atlassian dev tools blog.

The list below is a summary of the Stash releases. The change logs included in the release notes (linked below) have details of the related bug-fix releases.

Stash 3.9

14 May 2015

- Installer improvements
- Improved handling of unlicensed/unauthorized users
- 15+ public issues resolved

Read more in the Stash 3.9 release notes and change log.

See the Stash upgrade guide.

Stash 3.8

1 April 2015

- Completely provision Stash automatically
- Monitor Stash performance with JMX counters
- Deep linking into source file lines
- Cleaner handling of unlicensed and unauthorized users
- Tomcat's `server.xml` file has been relocated

- Stash Data Center node IDs stability
- Better support for the `go get` command

Read more in the Stash 3.8 release notes and change log.

See the Stash upgrade guide.

Stash 3.7

24 February 2015

- Customize the commit message when merging
- Like and reply to comments, from email notifications
- Installer improvements

Read more in the Stash 3.7 release notes and change log.

See the Stash upgrade guide.

Stash 3.6

13 January 2015

Small improvements and bug fixes:

- Create a branch starting from a tag
- Support for MariaDB
- Support for SMTPS for email notifications
- Stash is more accessible
- Syntax highlighting for fenced code blocks in markdown
- Support for Internet Explorer 9 is deprecated
- 30+ public issues resolved

Read more in the Stash 3.6 release notes and change log.

See the Stash upgrade guide.

Stash 3.5

25 November 2014

- Syntax highlighting in side-by-side diffs and source view
- Comment likes - amplify review feedback
- Tags are now displayed in the Commits list
- Support for Java 7 deprecated
- Support for Git versions earlier than 1.8 on the server is deprecated

Read more in the Stash 3.5 release notes and change log.

See the Stash upgrade guide.

Stash 3.4

21 October 2014

- Batched email notifications
- Support for Microsoft Office and OpenOffice/LibreOffice MIME types
- Aggregated group membership option for multiple user directories
- Disabling pull request assistive URLs in the console
- JMX support

Read more in the Stash 3.4 release notes and change log.

See the Stash upgrade guide.

Stash 3.3

10 September 2014

- Tasks for pull requests
- Tomcat 8 is now bundled
- Pull request URLs are displayed in the console after pushing

Read more in the Stash 3.3 release notes and change log.

See the Stash upgrade guide.

Stash 3.2

30 July 2014

- Improved workflow for creating pull requests
- Improved comment navigation in diffs - jump to next/previous comment
- Landing page for new users
- Stash analytics disclosure

Read more in the Stash 3.2 release notes and change log.

See the Stash upgrade guide.

Stash 3.1

24 June 2014

- Code search in pull request diffs
- Attachments for pull request comments and descriptions
- Stash installer for Linux, Mac OS X and Windows
- Microsoft SQL Server 2014 is now supported
- Oracle 12c is now supported
- Transcoding is now supported for diff views

Read more in the Stash 3.1 release notes and change log.

See the Stash upgrade guide.

Stash 3.0

20 May 2014

- Branch compare
- Sidebar redesign
- Webcam capture of user avatars
- Stash internationalization
- Java 6 support removed. Stash now requires Java 7 at least.
- Java 8 is now supported.
- Internet Explorer 8 support removed.
- Stash JavaScript API published
- Stash APIs deprecated in Stash 2.x releases (before 2.12) have now been removed.

Read more in the Stash 3.0 release notes and change log.

See the Stash upgrade guide.

Stash 2.12

25 March 2014

- Custom avatar images
- User name linking to profile page
- Read/write access keys
- Access key bulk revocation
- Diff hunk map
- DIY Backup
- Comment display toggling
- Markdown rendering in the Source view
- Markdown table syntax in comments

Read more in the Stash 2.12 release notes and change log.

See the Stash upgrade guide.

Stash 2.11

25 February 2014

- Commit comments
- File comments for pull requests and commits
- Side-by-side diffs
- Lock out recovery process
- MySQL 5.6.16+ support

Read more in the Stash 2.11 release notes and change log.

See the Stash upgrade guide.

Stash 2.10

17 December 2013

Small improvements:

- Branch Utils REST API
- The SSH clone URL is now shown for admins who don't have a user key, when a project or repository access key is available
- The permission screens for projects and repositories have been tidied
- Log files in Stash now use UTF-8 encoding when you start Stash from IDEA or Maven
- The URL now updates correctly when you switch revisions via the file history, when viewing the 'Diff to previous' tab for a repo
- Stash has been updated to use AUI 5.3
- Atlassian platform upgrade for 2.10

Read the Stash 2.10 release notes and change log.

Stash 2.9

19 November 2013

- Branch listing improvements
- SSH access keys for projects and repositories
- Pull request inbox – my pull requests
- Build status during branch creation
- Support for PostgrSQL 9.3
- Extra keyboard shortcuts
- Extra merge strategy option

- Support for changing usernames

Read the Stash 2.9 release notes and change log.

Stash 2.8

1 October 2013

- Stash branching model
- Branch creation from within Stash
- Branch creation from within JIRA
- Automated merges
- Branch listing page
- Move Git repositories between Stash projects
- Improved integration with Atlassian SourceTree
- Small improvements

Read the Stash 2.8 release notes and change log.

Stash 2.7

20 August 2013

- JIRA issue transitions
- Support for multiple JIRA instances
- Autolink JIRA issues in markdown
- Backup and restore client (beta)
- Small improvements

Read the Stash 2.7 release notes and change log.

Stash 2.6

22 July 2013

- Fork synchronisation
- Audit logging
- Repository Quicksearch
- Application navigator
- Public repositories list
- Small improvements

Read the Stash 2.6 release notes and change log.

Stash 2.5

12 June 2013

- Public access to projects and repositories
- Edit a pull request's destination branch
- Get more context in diffs
- OpenJDK is now supported
- Small improvements

Read the Stash 2.5 release notes and change log.

Stash 2.4

6 May 2013

- Forks
- Repository permissions
- Personal repositories
- Small improvements
- Deprecation of Java 6

Read the Stash 2.4 release notes and change log.

Stash 2.3

26 March 2013

- Crowd single sign-on (SSO)
- Branch deletion
- Git submodule recognition
- SCM Cache plugin for Stash

Read the Stash 2.3 release notes and change log.

Stash 2.2

05 March 2013

- Git repository hooks
- API for hook integrations
- Merge checks for pull requests

Read the Stash 2.2 release notes and change log.

Stash 2.1

05 February 2013

- Pull request integration with JIRA
- Build status API
- Project avatars
- Pull request inbox
- Improved pull request title and description generation

Read the Stash 2.1 release notes.

See the change log for Stash 2.1.x minor releases.

Stash 2.0

04 December 2012

- Branch permissions
- Markdown support
- Mentions
- Enterprise licenses for 1000 and 2000 users
- Deprecation of Internet Explorer 8

Read the Stash 2.0 release notes.

See the change log for Stash 2.0.x minor releases.

Stash 1.3

09 October 2012

- Pull requests
- Notifications
- Improved keyboard shortcuts
- README – simple project documentation

Read the Stash 1.3 release notes.

See the change log for Stash 1.3.x minor releases.

Stash 1.2

07 August 2012

- MySQL, PostgreSQL, SQL Server and Oracle support
- Database migration
- File search
- Add-ons ecosystem
- Small improvements

Read the Stash 1.2 release notes.

See the change log for Stash 1.2.x minor releases.

Stash 1.1

19 June 2012

- SSH support
- Fast browsing
- Simple permissions
- Image diffs

Read the Stash 1.1 release notes.

See the change log for Stash 1.1.x minor releases.

Stash 1.0 is released!

1st May 2012

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

See the change log for Stash 1.0.x minor releases.

Stash upgrade guide

This page describes how to update your Stash installation to the latest version. We *strongly recommend* that you update Stash by performing the steps below.

Note that:

- This update process does *not* perform an in-place upgrade, but installs the new version of Stash into a fresh installation directory. The new Stash uses your existing Stash home directory and external database.
- You can update from any previous version to the latest version of Stash.
- For production environments we recommend that you test the Stash update on a QA server before deploying to production.

1. Review the upgrade notes

There are specific upgrade notes further down this page for each version of Stash.

You should read the relevant sections for each version between your current version of Stash and the version you are upgrading to.

Note that you can update from any previous version to the latest version of Stash.

2. Back up your Stash data

See Data recovery and backups for more information.

3. Stop Stash

See Starting and stopping Stash. For Stash Data Center instances, repeat the process on every cluster node.

4. Install Stash

This update process does *not* perform an in-place upgrade, but installs the new version of Stash into a fresh installation directory. The new Stash uses your existing Stash home directory and external database.

Check that you have all the system requirements for the new version of Stash, including Perl, to avoid any trouble.

See also the End of support announcements for Stash.

You can install Stash by either:

- Using the Stash installer, the recommended approach.
- Installing Stash from an archive file.

For Stash Data Center instances, repeat the update process for every cluster node.

Use the Stash installer to update your Stash installation (Recommended)

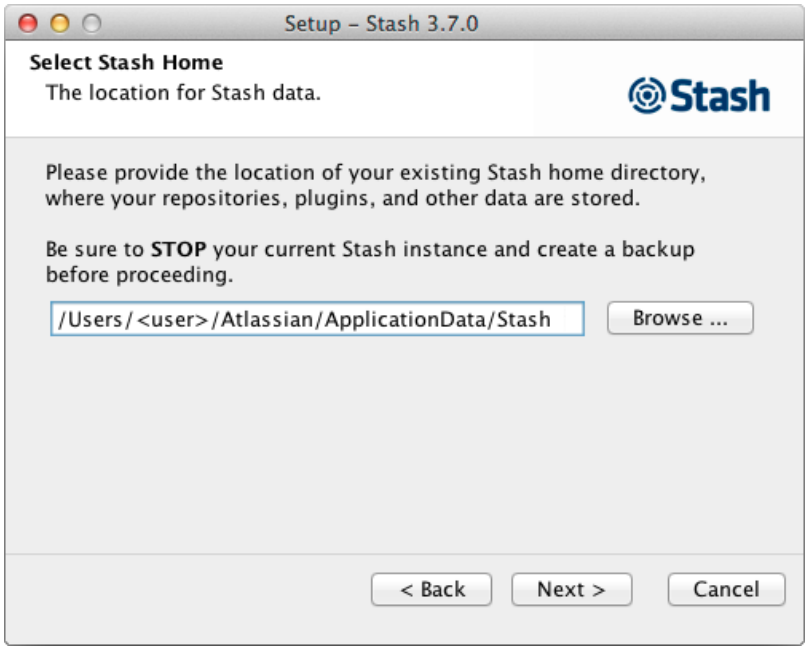
Download the Stash installer from the Atlassian download site. You can run the installer in GUI, console or unattended modes.

On Linux, you need to set the executable flag on the installer file before running it:

```
chmod +x atlassian-stash-x.x.x-x64.bin
```

At the 'Welcome' step, choose the **Upgrade an existing Stash instance** option:

At the 'Select Stash Home' step, browse to your existing Stash home directory (as defined by the `STASH_HOME` environment variable – see Stash home directory):



Stash 3.5 and later versions no longer allow the Stash home directory to be the same directory as, or a subdirectory of, the Stash installation directory. The Stash home directory, as defined by the `STASH_HOME` environment variable, must be in a separate location – otherwise Stash will fail on startup.

If upgrading from a previous version of Stash where the Stash home directory is owned by a local user, the ownership of all files will be updated to `atlstash` . Please allow sufficient time for the installer to complete this process (it could take up to 30 minutes) .

Proceed with the remaining installer steps.

Use an archive file to update your Stash installation

You should only need to use an archive file to update Stash if the Stash installer, which is the recommended approach, is not suitable for your situation.

Follow the instructions at Installing Stash from an archive.

Note that on Windows, Stash 3.5 and later versions will refuse to start if the installation path contains spaces, when Stash is installed from an archive file. The Stash installer prevents you from creating such a path.

5. Update server.xml (upgrading from 3.7 or earlier)

If you are upgrading from Stash 3.8 or later you can skip to step 6, starting Stash.

If you are upgrading from Stash 3.7 or earlier and you have made custom changes to the `server.xml` fil e in your Stash installation, you have to make those changes in the `server.xml` file of your new installation as well, which is in a different location as of Stash 3.8.

You would have made these changes if you modified the port, context path, or the access protocol, or if you are running Stash behind a proxy and modified the Connector element.

Locating the server.xml file

The location of the `server.xml` file changed in Stash 3.8, to keep editable configuration files in the Stash home directory and out of the (version specific) Stash installation directories.

Release	Directory
Stash 3.7 and earlier	`<Stash installation directory>/conf/server.xml`
Stash 3.8 and later	`<Stash home directory>/shared/server.xml`

Applying your customizations

Note that the default contents of `server.xml` was not always identical in past versions of Stash. Rather than just copying `server.xml` from your existing `<Stash installation directory>/conf` to `<Stash home directory>/shared` of your latest Stash instance, you should carefully review the differences between your customized version and the default version and re-apply just your custom changes to the new server.xml file.

Once you have applied your custom changes in `<Stash home directory>/shared/server.xml`, you will not need to perform this step again in future upgrades.

Custom configurations in Stash Data Center

For Stash Data Center, using Stash 3.8 or later, the single `server.xml` file in the `<Stash home directory> /shared` directory replaces all the copies of `server.xml` located in the `<Stash installation directory>/conf` directories of the cluster nodes using previous versions of Stash.

6. Start Stash

If you installed Stash using the installer, Stash will already have been started by the installer.

If you installed Stash manually from an archive file then see Starting and stopping Stash. For Stash Data Center instances, repeat the process on every cluster node.

Either way, note that the database schema migration task that runs when Stash is started after an update can take a while, especially if the update has skipped a few releases (for example, upgrading from Stash 2.2

to 2.7). Stash should never be interrupted while this is happening, even if Stash appears to have hung – allow the server to either come up, or fail to come up (when it will provide an explanation of what went wrong).

Home directory migration

When you update from Stash 3.1 or older to Stash 3.2 or later, an update task migrates directories in your Stash home directory to new locations. This is irreversible – once you update to Stash 3.2 or later you can not revert to Stash 3.1 or earlier, because of changes to the Stash home directory format.

For most installations, Stash is able to perform these moves automatically and transparently. However, in the rare instance where Stash can't perform the update automatically, please refer to Upgrading your Stash home directory for Stash 3.2 manually.

Rollback

If necessary, rolling back an update can only be performed by restoring a backup of *both* the Stash home directory and the Stash database – rolling back requires a consistent home directory and database. You can then reinstall the previous version of Stash to the `<Stash installation directory>`.

Version-specific update notes

This section provides specific update notes for each version of Stash. These notes supplement the primary update guide above.

You should read the relevant sections for each version between your current version of Stash and the version you are upgrading to.

- Locating the server.xml file
- Applying your customizations
- Custom configurations in Stash Data Center
- Stash 3.9 update notes
- Stash 3.8 update notes
- Stash 3.7 update notes
- Stash 3.6 update notes
- Stash 3.5 update notes
- Stash 3.4 update notes
- Stash 3.3 update notes
- Stash 3.2 update notes
- Stash 3.1 update notes
- Stash 3.0 update notes
- Stash 2.12 update notes
- Stash 2.11 update notes
- Stash 2.10 update notes
- Stash 2.9 update notes
- Stash 2.8 update notes
- Stash 2.7 update notes
- Stash 2.6 update notes
- Stash 2.5 update notes
- Stash 2.4 update notes
- Stash 2.3 update notes
- Stash 2.2 update notes
- Stash 2.1 update notes
- Stash 2.0 update notes

Stash 3.9 update notes

Please also see:

- the general Upgrade steps section above.
- the Stash 3.9 release notes.
- the Stash Supported platforms page. Note that Stash 3.9 does not support Git 2.0.2 or 2.0.3.
- the End of support announcements for Stash. Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Are you using JIRA as a Crowd server?

Stash 3.9 cannot use JIRA 6.4 as a Crowd server due to a bug in JIRA 6.4. Please upgrade to JIRA 6.4.1 before upgrading Stash. (Note that Stash 3.7.2 and 3.8.0 contained a workaround for the bug but the workaround was removed in Stash 3.9)

Known issues

P	Key	Summary	Status
↑	STASH-7216	Pull Request race condition between push and attributed activity author	OPEN
↑	STASH-7297	In a pull request, the methode contentService.streamFile(Repository, String, String, TypeAwareOutputSupplier) throws NoSuchPathException if the file was moved in the destination branch	OPEN
↑	STASH-7398	Performance regression for instances with a large disparity of licensed to unlicensed users	OPEN
↓	STASH-7296	Web-section "stash.pull-request.toolbar.section.user-actions" isn't being shown for pull-request author	OPEN
↓	STASH-7335	Clone Dialog mixed with the background on FF when is on top of the screen and a source file is open	OPEN
↓	STASH-7293	401 Error received on Pull Request merge REST API when a conflict is met	OPEN
↓	STASH-7260	Anchor links in markdown broken when not on the default branch	OPEN
↓	STASH-7354	Misspelled error message when entering Safe Mode	OPEN

8 issues

Stash 3.8 update notes

Please also see:

- the general Upgrade steps section above.
- the Stash 3.8 release notes.
- the Stash Supported platforms page. Note that Stash 3.8 does not support Git 2.0.2 or 2.0.3.
- the End of support announcements for Stash. Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Have you made customizations to Tomcat's `server.xml` file?

For Stash 3.8 (and future versions) the `server.xml` file is now located in the `<stash home directory>/shared` directory. The benefit of this move is that your customizations to `server.xml` (such as those described in step 5) will not have to be redone for future upgrades.

⚠ You do still need to update your custom configurations in `shared/server.xml` for the upgrade to Stash 3.8, as described in step 5.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See Syntax highlight changes for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

P	Key	Summary	Status
↑	STASH-7216	Pull Request race condition between push and attributed activity author	OPEN
↑	STASH-7297	In a pull request, the methode contentService.streamFile(Repository, String, String, TypeAwareOutputSupplier) throws NoSuchPathException if the file was moved in the destination branch	OPEN
↑	STASH-7398	Performance regression for instances with a large disparity of licensed to unlicensed users	OPEN
↓	STASH-7296	Web-section "stash.pull-request.toolbar.section.user-actions" isn't being shown for pull-request author	OPEN
↓	STASH-7335	Clone Dialog mixed with the background on FF when is on top of the screen and a source file is open	OPEN
↓	STASH-7293	401 Error received on Pull Request merge REST API when a conflict is met	OPEN
↓	STASH-7260	Anchor links in markdown broken when not on the default branch	OPEN
↓	STASH-7354	Misspelled error message when entering Safe Mode	OPEN

8 issues

Stash 3.7 update notes

Please also see:

- the general Upgrade steps section above.
- the Stash 3.7 release notes.
- the Stash Supported platforms page. Note that Stash 3.7 does not support Git 2.0.2 or 2.0.3.
- the End of support announcements for Stash. Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Do you use JIRA 6.4?

> JIRA 6.4 has a known issue with Stash versions 3.4.3 to 3.7.1, which prevents the user synchronisation from working. This will only affect you if you use JIRA to manage your users in Stash, and is fixed in Stash 3.7.2.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See Syntax highlight changes for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

P	Key	Summary	Status
↓	STASH-7151	Stash web merge-help dialogue should indicate that it's performing a manual merge	OPEN

1 issue

Stash 3.6 update notes

Please also see:

- the general update steps section above.
- the Stash 3.6 release notes.
- the Stash Supported platforms page. Note that Stash 3.6 does not support Git 2.0.2 or 2.0.3.

- the End of support announcements for Stash. Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Secure email notifications

Stash 3.6 now:

- Allows you to require STARTTLS support on the mail server when using SMTP – if it is not supported, mail is not sent.
- Supports SMTPS (where the whole protocol conversation uses SSL/TLS).

See Setting up your mail server for more information.

Note that if you use either SMTP with STARTTLS or SMTPS and connect to a self-signed mail server you may need to import the server's certificate and set up a custom cacerts file for Stash (just as you do for any outbound SSL/TLS connection to a self-signed server). See this Stash knowledge base article for information about how to do that.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See Syntax highlight changes for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

P	Key	Summary	Status
↑	STASH-7203	Mentioning a commit SHA creates a link to a wrong repository	OPEN
↑	STASH-6983	New Issues In Already Open Pull Request Don't Fire Triggers	OPEN
↓	STASH-7135	Error when creating tasks on declined pull requests	OPEN
↓	STASH-7090	Merge conflict message not shown on the overview tab to read only users	OPEN
↓	STASH-7093	Syntax highlighting for Scala does not recognise symbol literals	OPEN
↓	STASH-7190	stash GUI breaks selection on linux/X11	OPEN
↓	STASH-6958	Small UI issues with tasks in IE11	OPEN
↓	STASH-7085	LDAP Credentials Not Persisted After "Test Settings"	OPEN
↓	STASH-7213	In Overview comment doesn't show context - likely because of Comment Drift	OPEN
↓	STASH-7177	README.md files are rendered differently from other *.md files	OPEN
↓	STASH-7390	java.io.UTFDataFormatException error when hook rejection message is > 65535 characters and contains non-ASCII characters	OPEN
↓	STASH-7083	Adding JIRA User Directory With URL Context Path Containing "services" Fails	OPEN

12 issues

Stash 3.5 update notes

Please also see:

- the general update steps section above.
- the Stash 3.5 release notes.
- the Stash Supported platforms page. Note that Stash 3.5 does not support Git 2.0.2 or 2.0.3.
- the End of support announcements for Stash. Note that, in particular, Stash 4.0 will not support Java 7, nor versions of Git earlier than 1.8 on the server.

Stash home directory location

Stash 3.5 and later versions no longer allow the Stash home directory to be the same directory as, or a subdirectory of, the Stash installation directory. The Stash home directory, as defined by the `STASH_HOME` environment variable, must be in a separate location – Stash will fail on startup otherwise.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine in Stash has been changed from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See Syntax highlight changes for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

JIRA Issues Macro: JIRA project does not exist or you do not have permission to view it.

Stash 3.4 update notes

Please also see:

- the general update steps section above.
- the Stash 3.4 release notes.
- the Stash Supported platforms page. Note that Stash 3.4 does not support Git 2.0.2 or 2.0.3.
- the End of support announcements for Stash.

Changed group membership aggregation with multiple user directories

Stash 3.4 uses new schemes to determine the effective group memberships when Stash is connected to multiple user directories, and duplicate user names and group names are used across those directories. The new schemes are:

- 'aggregating membership'
- 'non-aggregating membership'.

These group membership schemes are only used to determine authorisation. Authentication is determined on the basis of the group mappings in each directory.

See Effective memberships with multiple directories in the Crowd documentation for more information.

When you update to Stash 3.4, an update task will apply one of those schemes as follows:

Aggregating membership will be applied to your instance:

- If there is only one active directory.
- If there are multiple active directories but only a single directory applies group memberships to any particular user.
- For example, if directory-1 contains user-a in group-x, and directory-2 contains user-b in group-y, then Stash 3.4 will apply aggregating membership, and permissions will not be affected.

Non-aggregating membership will be applied to your instance:

- If there are multiple active directories and more than one directory applies group memberships to at least one user.
- For example, if directory-1 contains user-a in group-x, and directory-2 contains user-a in group-y, then Stash 3.4 will apply non-aggregating membership. If group-y has admin provileges then user-a would have their privileges escalated in this case if aggregating membership was applied instead when upgrading to Stash 3.4.

Any changes made by the update task will be logged.

A Stash admin can change the membership scheme used by Stash using the following commands:

- To change to *aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d
'{"membershipAggregationEnabled":true}' -u <username>:<password>
<base-url>/rest/crowd/latest/application
```

- To change to non-*aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d
'{"membershipAggregationEnabled":false}' -u <username>:<password>
<base-url>/rest/crowd/latest/application
```

Please note that changing the aggregation scheme can affect the authorisation permissions for your Stash users, and how update operations are performed. Read more about using Stash with multiple use directories.

Known issues

P	Key	Summary	Status
↑	STASH-5501	Outbound Proxy is bypassed when connecting to external Crowd Directory	OPEN
↑	STASH-5511	Plain Delegated LDAP configuration mistankenly uses hidden forms	OPEN
↓	STASH-5528	Unable to generate REST API java bindings from wadl file	OPEN
↓	STASH-5506	Error 404 for Stash Internal Directory users that are duplicated in Active Directory which are disabled	OPEN
↓	STASH-5442	Stash source distribution does not contain embedded-crowd-admin-plugin	OPEN
↓	STASH-5492	View JIRA Issue - Small Design Issue	OPEN
↓	STASH-5429	pullrequests/diff rest call returns incomplete JSON response	OPEN
↓	STASH-5475	The attached file in pull request cannot be accessed (Permission Denied) by other Stash Users	OPEN
↓	STASH-5490	Language sometimes not consistent	OPEN
↓	STASH-7044	Page titles for Audit Log and Access Keys don't handle XML Entities	OPEN

10 issues

Stash 3.3 update notes

Please also see:

- the general update steps section above.
- the Stash 3.3 release notes.
- the Stash Supported platforms page. Note that Stash 3.3 does not support Git 2.0.2 or 2.0.3.
- the End of support announcements for Stash.

Known issues

P	Key	Summary	Status

↑	STASH-5501	Outbound Proxy is bypassed when connecting to external Crowd Directory	OPEN
↑	STASH-5218	Potential deadlock when waiting for db connections to drain during backup	OPEN
↓	STASH-7083	Adding JIRA User Directory With URL Context Path Containing "services" Fails	OPEN
↓	STASH-5441	@ (at symbol) within Markdown backticks should not be interpreted as @mention	OPEN
↓	STASH-5313	Updating 2LO and 2LOi flags on incoming authentication does not update corresponding outgoing authentication on opposite site of link and vice versa	OPEN
↓	STASH-5290	Pull Request overview incorrectly displaying the number of tasks	OPEN
↓	STASH-5345	Permissions for deleted users have broken links	OPEN
↓	STASH-5224	Expanding diff views down does not show end of file on first click	OPEN

8 issues

Stash 3.2 update notes

> Upgrading to Stash 3.2 or later from Stash 3.1 or earlier is irreversible – please see the Home directory migration section below.

Please also see:

- the general update steps section above.
- the Stash 3.2 release notes.
- the Stash Supported platforms page.
- the End of support announcements for Stash.

Note that:

- Stash 3.2 does not support Git 2.0.2 or 2.0.3.

Home directory migration

When you update to Stash 3.2 or later, an update task migrates directories in your Stash home directory to new locations. This is irreversible – once you update to Stash 3.2 or later you can not revert to Stash 3.1 or earlier, because of changes to the Stash home directory format.

For most installations, Stash 3.2 is able to perform these moves automatically and transparently. However, in the rare instance where Stash 3.2 can't perform the update automatically, please refer to Upgrading your Stash home directory for Stash 3.2 manually.

Undocumented home directory overrides are no longer supported

Before Stash 3.2 it was possible to override the location of the following subfolders in the Stash home directory, using undocumented environment variables or system properties:

- `export`
- `bin`
- `caches`
- `config`
- `data`
- `lib`
- `lib/native`
- `log`
- `plugins`
- `tmp`

In Stash 3.2 only the `tmp` subfolder can be overridden in this way. Attempting to override the others will fail on startup. For more information, please see Stash fails to start - UnsupportedDirectoryOverrideException .

Stash analytics

Stash 3.2, and later, collects user event data, unless this is disabled by an administrator. You will see outgoing network traffic to amazonaws.com. See Collecting analytics for Stash.

Known issues

P	Key	Summary	Status
↑	STASH-5218	Potential deadlock when waiting for db connections to drain during backup	OPEN
↑	STASH-5178	Branchname with german umlaut	OPEN
↓	STASH-7410	Stash Installer will not work with Windows Server 2003 R2 32-bit	OPEN
↓	STASH-5527	Stash diff view doesn't show when file changed from being symlink	OPEN
↓	STASH-5203	Repositories nav menu z-index is behind UPM filter dropdown z-index	OPEN
↓	STASH-5170	Avatar dialog misleading on small-ish displays	OPEN
↓	STASH-5113	BranchSelector does not propagate value in PullRequestCreation	OPEN
↓	STASH-5100	Windows installer defines STASH_HOME with duplicate path separators (\) in setenv.bat	OPEN

8 issues

Stash 3.1 update notes

Please also see:

- the general update steps section above.
- the Stash 3.1 release notes.
- the Stash Supported platforms page.

Note that:

- Stash does not support Git 2.0.2 or 2.0.3.

Known issues

P	Key	Summary	Status
↓	STASH-6927	Query error handle blocks restful table	OPEN
↓	STASH-5292	Anchor links in markdown include a slash that causes them to work inconsistently	OPEN
↓	STASH-5114	Last parenthesis in URL does not work when it is last character	OPEN
↓	STASH-5027	Highlighting CMake files	OPEN

4 issues

Stash 3.0 update notes

Stash no longer supports Java 6
Stash 3.0 requires at least Java 7, and supports Java 8.

Please see the End of support announcements for Stash.

Your plugins may not be compatible with Stash 3.0
The interfaces in the Stash API for plugin developers that *were deprecated* in Stash 2.11 and earlier *have been removed* in Stash 3.0. This means that, unless they have been updated to work with Stash 3.0, existing Stash add-ons (or plugins) that use these interfaces *will not work with Stash 3.0*.

Please see the section below about Stash add-on incompatibilities for more details.

Please also see:

- the general update steps section above.

- the Stash 3.0 release notes.
- the Stash Supported platforms page.

Note that:

- Stash now has installers for the Linux, Mac OS X and Windows platforms.
- Stash no longer supports Internet Explorer 8, as previoiusly announced.
- Stash does not support the Apache HTTP Server `mod_auth_basic` module.
- Stash does not support Git 2.0.2 or 2.0.3.

Stash add-on incompatibilities

Unless they have been updated to work with Stash 3.0, existing Stash add-ons (or plugins) that use the API interfaces that have been removed in Stash 3.0 *will not work*.

Fresh installs of Stash 3.0 shouldn't encounter any problems. The Stash 'Manage add-ons' page (in the admin area) should only display add-ons from the Marketplace that have been marked as compatible with Stash 3.0. Incompatible add-ons won't be available in the list.

However, if you are upgrading from Stash 2.x to Stash 3.0, you should be aware that some existing installed add-ons may be incompatible with Stash 3.0. After upgrading, you should go to **Admin** > **Manage add-ons**, look for messages of this form, and follow the advice to update:

If no newer version is available, the add-on must be disabled.

Custom add-ons

Please note that your custom locally-developed plugins may be affected by the API removals in Stash 3.0. You will need to update your custom plugins if you want those to work with Stash 3.0. See the Stash API changelog for details of the deprecated APIs.

Third-party add-ons

You'll need to check on Atlassian Marketplace for the compatibility status of any 3rd-party add-ons that you use.

Third-party add-on developers have been given an Early Access Program (EAP) build of Stash 3.0 in advance of release, and many have already updated their add-ons to be compatible. Add-ons must be explicitly marked by the publisher as compatible with Stash 3.0 for them to appear in 'Manage add-ons' page in Stash. ***This is NOT automatic as was the case with previous minor releases such as 2.10 and 2.11.*** Atlassian can not support issues involving third party Add-ons that are incompatible with Stash 3.0; such support cases must be directed to the third-party publisher of the add-on. See Managing add-ons.

Atlassian add-ons

All of the Atlassian add-ons for Stash that are available from the Atlassian Marketplace have been updated to be compatible with Stash 3.0. If you use any of these in your Stash installation you'll need to update them to the Stash 3.0 compatible version.

Add-on	JAR file name	Stash 3.0 compatible version
Custom Navigation Plugin	custom-navigation-plugin	2.0.3
Stash Archive Plugin	stash-archive	1.3.0
Repository git operations plugin	stash-git-ops-plugin	1.2.1
Stash Auto Unapprove Plugin	stash-auto-unapprove-plugin	1.1

Stash Protect Unmerged Branch Hook	stash-protect-unmerged-branch-hook	1.1
Stash Reviewer Suggester	stash-suggest-reviewers	1.2
Stash Web Post Hooks Plugin	stash-web-post-receive-hooks-plugin	1.1.0
Realtime Editor for Stash	stash-editor-plugin	1.0.6

Known issues

P	Key	Summary	Status
↓	STASH-7184	Project avatar resource vulnerable to XSRF	OPEN
↓	STASH-4875	Stash DIY - update to backup progress has unexpected results	OPEN
↓	STASH-4788	Redirect loop going to /plugins/servlet URL that you don't have permission to access	OPEN
↓	STASH-4787	Space doesn't submit the comment form.	OPEN

4 issues

Stash 2.12 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash. In particular, note that support for Java 6 is deprecated. Stash 3.0+ will require Java 7.
- the Stash 2.12 release notes.

Note that:

- Stash does not yet support Java 8.
- Stash 2.12 does not support Git 1.8.4.3
- Stash does not support the Apache HTTP Server `mod_auth_basic` module.

See Supported platforms.

Known issues

P	Key	Summary	Status
↑	STASH-7297	In a pull request, the methode contentService.streamFile(Repository, String, String, TypeAwareOutputSupplier) throws NoSuchPathException if the file was moved in the destination branch	OPEN
↓	STASH-7390	java.io.UTFDataFormatException error when hook rejection message is > 65535 characters and contains non-ASCII characters	OPEN
↓	STASH-4852	Viewing Markdown file with anchor tag for internal reference cause file not to render	OPEN
↓	STASH-4735	Emphasis directly after parentheses broken	OPEN
↓	STASH-4607	SSH submodule not rendered as link due to different hostname between HTTP and SSH URL	OPEN
↓	STASH-4573	Text becomes blurry in Chrome 33 on Windows 7, while using zoom	OPEN
↓	STASH-4533	Comment on repository settings entry for "Default branch" are confusing	OPEN

7 issues

Stash 2.11 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash.
- the Stash 2.11 release notes.

Note that Stash does not support Git 1.8.4.3, nor does Stash support Java 8 yet. See Supported platforms.

Known issues

P	Key	Summary	Status
↓	STASH-5189	Home/End/PageUp/PageDown don't work in Pull request	OPEN
↓	STASH-4662	Handle unreachable Application Links gracefully when their unreachability is non-critical	OPEN
↓	STASH-4607	SSH submodule not rendered as link due to different hostname between HTTP and SSH URL	OPEN
↓	STASH-4584	create branch from issue - chars replacing missing "" char	OPEN
↓	STASH-4505	Preserve styling in Jira task description when looking at the task from Stash	OPEN

5 issues

Stash 2.10 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash.
- the Stash 2.10 release notes.

Note that Stash does not support Git 1.8.4.3. See Supported platforms.

Known issues

P	Key	Summary	Status
↑	STASH-7073	Embedded Crowd's transaction setting clash with Stash's transaction settings	OPEN
↓	STASH-5187	After deleting a Branch via Pull Request merge, all links containing a branch are dead	OPEN
↓	STASH-5084	Empty repo instructions not shown to anonymous users	OPEN
↓	STASH-4575	My list of open PRs needing review will intermittently not fully load	OPEN
↓	STASH-4389	Inline display of svg files	OPEN
↓	STASH-4254	Unknown error when clicking on JIRA link in a pull request	OPEN

6 issues

Stash 2.9 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash.
- the Stash 2.9 release notes.

Note that Stash does not support Git 1.8.4.3. See Supported platforms.

Pull Request Ref Optimization

When you first start Stash after upgrading to Stash 2.9 a repository update task runs that optimizes the pull request refs for all repositories managed by Stash. It's important that you do not interrupt this update process. You can track the progress of this in the Stash logs. See

STASH-3469 - Pull request references in the Git repository are never removed after merge or decline CLOSED .

Backup Client Update Required

Version 1.0.3 of the Stash Backup Client is required to back up Stash 2.9.

Known issues

P	Key	Summary	Status
↓	STASH-4691	Renaming a shadowed user incorrectly renames the associated stash user	OPEN

1 issue

Stash 2.8 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash.
- the Stash 2.8 release notes.

Known issues

P	Key	Summary	Status
↑	STASH-3884	Non-ASCII values used as branch model prefixes/branch names don't work in MSSQL	OPEN
↓	STASH-4302	False-positive issue key recognition in whole word	OPEN
↓	STASH-4033	Clicking "back" after viewing commit details sometimes lists commits with incorrect order	OPEN
↓	STASH-3982	Diff of files is impossible to read when Ctrl+M is used as line separator	OPEN
↓	STASH-3909	Creating a branch containing double quotes in the name will fail on Windows	OPEN

5 issues

Stash 2.7 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash.

Repository System Information Plugin is now deprecated

The functionality of the repository system information plugin has now been moved into core Stash. The plugin will still work for Stash 2.x versions but is redundant as of Stash 2.7.

MySQL default isolation level

Stash 2.7.x uses READ_COMMITTED instead of the MySQL default isolation level (REPEATABLE_READ). This can result in exceptions when installing or upgrading to 2.7.x, if binary logging is enabled in your MySQL server. More details and a fix can be found in this KB article.

Known issues

P	Key	Summary	Status
↑	STASH-3868	"does not exist at revision" error when browsing files with umlaut characters	OPEN
↓	STASH-4356	Notification not sent when adding mention to comment when editing	OPEN
↓	STASH-3990	Mismatches in SQL Server database names cause inscrutable migration errors	OPEN

3 issues

Stash 2.6 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash.

Known issues

P	Key	Summary	Status

No issues found

Stash 2.5 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash.

Known issues

P	Key	Summary	Status
↓	STASH-4417	RepositoryService#countByProject does not take into account repositories that are publicly accesible	OPEN
↓	STASH-3279	URL surrounded by parentheses and followed by punctuation includes trailing parenthesis in URL	OPEN

2 issues

Limited support for JIRA 4.4.x and earlier

JIRA 4.3+ allows for showing commits associated with issues in JIRA. However, viewing issues within Stash is not supported for JIRA 4.4.x and earlier. See JIRA compatibility for details.

Stash 2.4 update notes

Please also see:

- the update steps section above.
- the End of support announcements for Stash.

Known issues

P	Key	Summary	Status
↓	STASH-3818	Inline-HTML: Element-Attribute-ID is Not Rendered	OPEN

1 issue

Stash 2.3 update notes

Please also see the update steps section above.

When upgrading to Stash 2.3 you also need to update the SCM Cache plugin, due to recent Stash API changes.

Known issues

P	Key	Summary	Status
↓	STASH-3308	Git command authentication failure causing the user to enter captcha before maxmimum attempts are met.	OPEN
↓	STASH-3279	URL surrounded by parentheses and followed by punctuation includes trailing parenthesis in URL	OPEN

2 issues

Stash 2.2 update notes

Please see the update steps section above.

Known issues

P	Key	Summary	Status
↓	STASH-3295	Comment Date format changes from relative to absolute dates unexpectedly	OPEN

1 issue

Stash 2.1 update notes

Please also see the update steps section above.

Known issues

P	Key	Summary	Status
↑	STASH-3187	Concurrent commits force Stash to create wrong comments in Pull Requests	OPEN
↓	STASH-3308	Git command authentication failure causing the user to enter captcha before maxmimum attempts are met.	OPEN

2 issues

Install location for third-party libraries

As of Stash 2.1 you can install third-party libraries and jar files, such as the MySQL JDBC driver, into `<Stash home directory>/lib`. This has the advantage that files in this location are not overwritten, and lost, when you update Stash.

Microsoft SQL Server JDBC driver

Stash 2.1 now uses the Microsoft SQL Server JDBC driver to access Microsoft SQL Server, instead of using the jTDS driver. In most cases, Stash will automatically swap to using the Microsoft driver on update and **no**

configuration is required.

If Stash was configured to use Microsoft SQL Server by manually entering a JDBC URL, please refer to this guide.

Stash 2.0 update notes

This section provides specific notes for upgrading to Stash 2.0. See also the update steps section above.

Tomcat

For Stash 2.0, Tomcat has been updated from version 6 to 7. As part of that update, the `server.xml` file has changed. If you have customised `server.xml` (for example, for `port`, `path` or `hostname`), you can not simply copy your own version across to the updated Stash; you must reapply your customisations to the `server.xml` file for the new version of Stash.

If you were running Stash as a Windows service and are upgrading from 1.x to 2.x you will need to reinstall the Stash service to make it use Tomcat 7.

To uninstall the Stash service you need to execute following commands from <STASH DISTRIBUTION DIR>\bin:

```
> net stop <service name>
> service.bat uninstall <service name>
```

You can call this command without the service name if you installed the Stash service with a default name.

After the service is uninstalled you can proceed with the update steps and Running Stash as a Windows service instructions to configure Stash 2.x running as a service.

Perl

Stash 2.0 requires Perl for its branch permission functionality. If Perl is unavailable, Stash 2.0 will not start.

On Windows machines, Perl will only have been installed by the Git installer if the correct install option was chosen. See Installing and upgrading Git.

Existing Git hooks

In order to support Branch Permissions, Stash 2.0 moves existing hooks in the `pre-receive` and `post-receive` folders under `<STASH_HOME>/data/repositories/NNNN/hooks` (where NNN is the internal repository id) to `.../hooks/pre-receive.d/10_custom` or `.../hooks/post-receive.d/10_custom`. Consequently, custom hooks that use relative path names (e.g. "./foo.sh" or "../dir/foo.sh") will be broken by the update to Stash 2.0.

Deprecation of Internet Explorer 8

Support for Internet Explorer 8 is deprecated from the release of Stash 2.0. The official end-of-support date is yet to be determined. See Supported platforms for details.

Known issues

P	Key	Summary	Status
↑	STASH-2862	Non-ASCII values used in branch permissions don't work in MSSQL	OPEN
↓	STASH-3161	Developer Toolbox > Atlassian Stash REST Plugin missing parameters	OPEN

2 issues

Checking for known issues and troubleshooting

If something is not working correctly after you have completed the steps above to update your Stash installation, please check for known Stash issues and try troubleshooting your update as described below:

- **Check for known issues**. Known issues can be seen in the STASH project on our issue tracker.
- **Stash Knowledge Base.** Sometimes we find out about a problem with the latest version of Stash after we have released the software. In such cases we publish information in the Stash Knowledge Base.
- If you encounter a problem during the update and cannot solve it, please create a support ticket and one of our support engineers will help you.

End of support announcements for Stash

End of support matrix for Stash

The table below summarises the end of support announcements for recent Stash releases:

Platform/functionality	Announcement date	Stash end of support
Deprecation of Internet Explorer 10	5 May 2015	Q4 2015
Deprecation of Internet Explorer 9	13 January 2015	From Stash 3.10
Deprecation of Java 7	25 November 2014	From Stash 4.0
Deprecation of Git versions earlier than v1.8	25 November 2014	From Stash 4.0
Deprecation of Tomcat 7	10 September 2014	From Stash 4.0

Why is Atlassian ending support for these platforms?
Atlassian is committed to delivering improvements and bug fixes as fast as possible. We are also committed to providing world class support for all the platforms our customers run our software on. However, as new versions of databases, web browsers etc. are released, the cost of supporting multiple platforms grows exponentially, making it harder to provide the level of support our customers have come to expect from us. Therefore, we no longer support platform versions marked as end-of-life by the vendor, or very old versions that are no longer widely used.

End of support announcements on this page (most recent announcements first):

- Deprecation of Internet Explorer 10 (announced 5 May 2015)
- Deprecation of Internet Explorer 9 (announced 13 January 2015)
- Deprecation of Java 7 (announced 25 November 2014)
- Deprecation of Git versions earlier than v1.8 (announced 25 November 2014)
- Deprecation of Tomcat 7 (announced 10 September 2014)
- Deprecation of Internet Explorer 8 (announced 22 July 2013)
- Deprecation of Java 6 (announced 9 May 2013)

Deprecation of Internet Explorer 10 (announced 5 May 2015)

During Q4 2015, Stash will no longer support Internet Explorer 10, and will only support Internet Explorer 11 and above. See Supported platforms.

Deprecation of Internet Explorer 9 (announced 13 January 2015)

In version 3.10, Stash will no longer support Internet Explorer 9, and will only support Internet Explorer 10 and above. Stash 3.10 is expected to be released around mid-2015. See Supported platforms.

Deprecation of Java 7 (announced 25 November 2014)

In version 4.0, Stash will no longer support Java 7, and will only support Java 8 and above. Stash 4.0 is expected to be released around mid-2015. See Supported platforms.

Deprecation of Git versions earlier than v1.8 (announced 25 November 2014)

In version 4.0, Stash will only support Git 1.8, and later versions (with the exceptions noted in Supported platforms), on the server. Stash 4.0 is expected to be released around mid-2015.

Deprecation of Tomcat 7 (announced 10 September 2014)

In version 4.0, Stash will no longer support Tomcat 7, and will only support Tomcat 8 and above. Stash 4.0 is expected to be released around mid-2015. See Supported platforms.

Deprecation of Internet Explorer 8 (announced 22 July 2013)

In version 3.0, Stash will no longer support Internet Explorer 8, and will only support Internet Explorer 9 and above. Stash 3.0 is now expected to be released mid-2014. See Supported platforms.

Deprecation of Java 6 (announced 9 May 2013)

In version 3.0, Stash will no longer support Java 6.0, and will only support Java 7.0 and above. Stash 3.0 is now expected to be released mid-2013. See Supported platforms.

Stash 3.9 release notes

14 May 2015

If you are upgrading from an earlier version of Stash, check the Stash upgrade guide. See also the End of support announcements for Stash and the changes listed in the API changelog.

The Stash 3.9 changelog is at the bottom of this page.

More improvements handling unlicensed and unauthorized users

We continue to improve how Stash handles unlicensed and unauthorized users pulled in from your external user directories.

Now, when granting access to users in projects, repositories, and branches you'll only see authorized, licensed Stash users – unlicensed and unauthorized users aren't listed, so you don't have to waste time working that out for yourself.

Installer improvements

We reduced the number of steps to get Stash in the hands of users with Stash 3.9.1. Now, generating an evaluation license and accepting the customer agreement is streamlined during the install process, to get you to the good stuff faster.

Change log

This section will contain information about the Stash 3.9 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, check the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.9.x releases, and are ordered by votes received.

22 May 2015 - Stash 3.9.2

T	Key	Summary
	STASH-2970	SNI not supported
	STASH-4832	Current version of AppLinks library does not URL encode username as part OAuth request
	STASH-7412	SoyTofuException When evaluating "to_json($repository)" after upgrade or install of 3.9.1
	STASH-4855	Mailto: links in linked JIRA issues are prepended with JIRA host URI
	STASH-7429	Provide fallback .default-server.xml in case $STASH_HOME/shared/server.xml is missing or unreadable
	STASH-7403	Exception on App Links admin screen after upgrading to 3.9.1 with a configured Trusted App
	STASH-7380	Project decorator is without sidebar for Stash 3+

7 issues

14 May 2015 - Stash 3.9.1

T	Key	Summary
	STASH-7105	SSH holds onto some sessions.
	STASH-4514	"Email a link to the user to set their password" checkbox and password should not be available if user is being created into a LDAP Delegated scenario
	STASH-2532	No indication to admin when granting a non-stash user a project permission
	STASH-7132	Disable weak SSH Ciphers
	STASH-7119	Preserve group data upon loss of connection to external user directory
	STASH-5238	API allows to change avatar for ServiceUser's but with no effect
	STASH-4478	Allow using left/right arrow to move side by side diff left/right
	STASH-5339	Pull request inbox sometimes not showing more than 10 results
	STASH-7371	Crowd directories which are uncached containing group memberships with a space in the group/username will cause a unique constraint or index violation; UK_MEM_DIR_PARENT_CHILD table: CWD_MEMBERSHIP when Stash is using MySQL
	STASH-7268	Remove the hotkey for Approve Pull Request
	STASH-7261	Updating repository hook settings creates new rows instead of updating existing ones
	STASH-6964	Stash should not attempt to authenticate to Crowd with a blank username
	STASH-7302	French footer translation is not good
	STASH-7081	Only 2 concurrent connections allowed to external Crowd server
	STASH-6979	Unrelated cluster nodes should not log a stacktrace
	STASH-7251	postgres version not properly detected
	STASH-7281	Diffs Never Load for Deleted Files that Have More than 10,000 Lines
	STASH-3908	Creating a branch in Stash with a prefix that matches an existing branch will fail
	STASH-7234	Raise an event when a build status is published
	STASH-7156	Atlassian Analytics Fills up the Event Queue in Stash

Showing 20 out of 22 issues

Stash 3.8 release notes

1 April 2015

If you are upgrading from an earlier version of Stash, check the Stash upgrade guide. See also the End of support announcements for Stash and the changes listed in the API changelog.

The Stash 3.8 changelog is at the bottom of this page.

Completely provision Stash automatically

Previously, even when installing Stash automatically, you still had to visit the Stash Setup Wizard in the browser to set the admin account, the Base URL and the external database connection, and to add the Stash license.

Now, by specifying all those settings in the Stash config properties file, you can completely install and set up a Stash instance automatically, without manual intervention. Stash will read the file, and apply those settings automatically after launching.

Read more about automated setup for Stash.

Monitor Stash performance with JMX counters

Stash now exposes instrumentation to provide better monitoring of Stash's performance. The range of metrics includes the number of projects and repositories, the number of Git pushes and pulls, and various thread pool metrics. These are all exposed as JMX counters, allowing you to use a JMX client, such as JConsole, to monitor Stash.

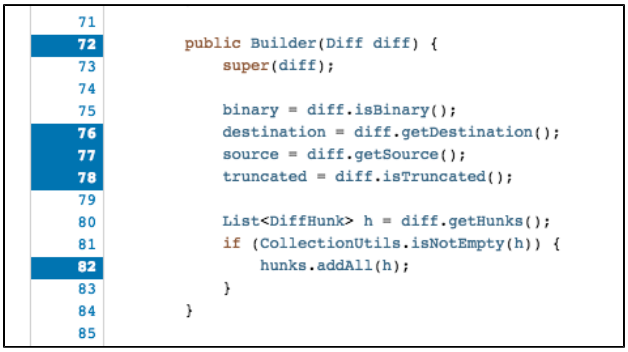
Deep linking into source file lines

We've improved how you can link to source files in Stash. Now you can Shift-click to select a range of lines, and Command-click (or Control-click) to add separate lines to the selection, then copy the updated URL for sharing with others.

Alternatively, you can modify the URL hash directly – for this selection it would be:

```
<your.stash.com/path/to/source>#72,76-78,82
```

Note that Internet Explorer doesn't support Ctrl-clicking, unfortunately.



Cleaner handling of unlicensed and unauthorized users

We've tidied up how Stash handles unlicensed and unauthorized users pulled in from your external user directories.

Now, when you're picking reviewers for a pull request you'll only see those users who actually have permission to read the repository – unlicensed and unauthorized users aren't listed, so you don't have to waste time working that out for yourself.

We've also cleaned up @mentions in Stash – users without a Stash license are now *never* displayed in the @mentions picker anywhere in Stash. As always, notifications are only sent to those who have permission to read the repository.

Small improvements

Tomcat's `server.xml` file has been relocated

In Stash 3.8, the `server.xml` configuration file has been moved to `<Stash home directory>/shared/server.xml`. If you have customized this file, you will have to copy it into this new location when upgrading to Stash 3.8. But once the configuration file is in your Stash home directory, you should not need to change it on future upgrades. See the upgrade notes for Stash 3.8.

Stash Data Center node IDs

You can now assign human readable node names to Stash Data Center that are consistent for the lifetime of a node, and stable across restarts. See Adding cluster nodes to Stash Data Center

Better support for `go get`

For developers hosting Go libraries in Stash, you can now use the `go get` command *without* specifying a VCS suffix.

Change log

This section will contain information about the Stash 3.8 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, check the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.8.x releases, and are ordered by votes received.

5 May 2015 - Stash 3.8.1

T	Key	Summary
	STASH-6964	Stash should not attempt to authenticate to Crowd with a blank username
	STASH-7337	Throttling of hosting operations over SSH is broken

2 issues

1 April 2015 - Stash 3.8.0

T	Key	Summary
	STASH-3584	Allow only licensed and authorised users to be reviewer
	STASH-6828	Automatic user synchronization stops if manual synchronization is performed
	STASH-5291	Make ordering of pre-receive hooks configurable
	STASH-5380	Add support for go get
	STASH-2840	Addition of Stash JMX Counters
	STASH-7195	JIRA 6.4 as a Crowd Server does not work with Stash 3.4.3 or higher
	STASH-7026	Limit @ mention list to only licensed users
	STASH-7106	Remember-me authentication sometimes doesn't work
	STASH-7063	Link to diff from comment in pull request overview should go to line
	STASH-7193	PullRequestMergedEvent is no longer asynchronous
	STASH-4190	expanding diff context on pullrequest overview shows only the unmodified lines
	STASH-7217	Incorrect nested home directory detection in installer

STASH-5208	OOME in ProfilingTimerBean
STASH-7001	Document REST API for moving repo
STASH-7053	Windows service not defining -XX:MaxPermSize is causing Stash to get stuck on startup
STASH-5488	Enable "Remember my login on this computer" by default
STASH-7110	NumberFormatException when parsing JIRA issue number
STASH-7129	Email notifications: the message 'and more commits' is missing when there is more than 5 commits for batched notifications
STASH-2477	Deep linking into a range of lines of source
STASH-7025	Inconsistent pull request attachment URLs

Showing 20 out of 24 issues

Stash 3.7 release notes

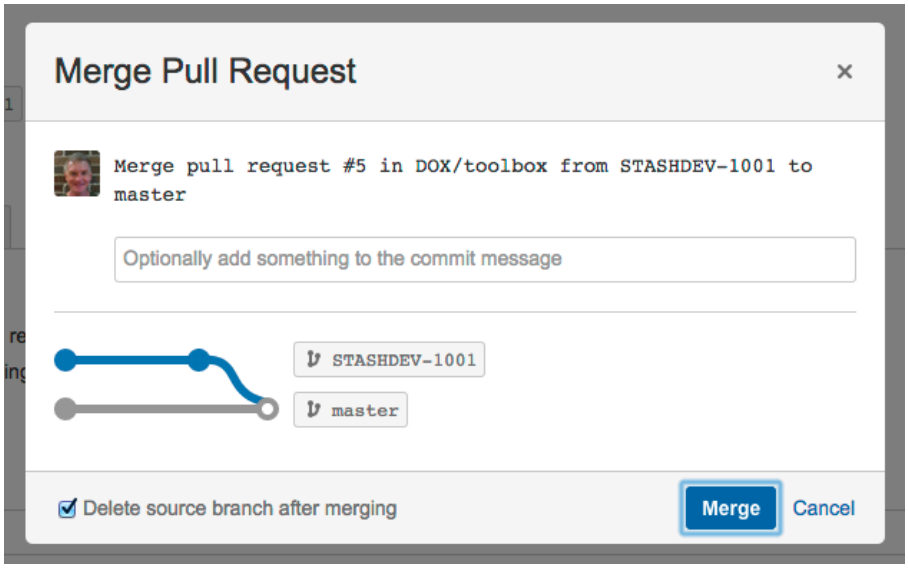
24 February 2015

If you are upgrading from an earlier version of Stash, check the Stash upgrade guide. See also the End of support announcements for Stash and the changes listed in the API changelog.

The Stash 3.7 changelog is at the bottom of this page.

Customize the commit message when merging

With Stash 3.7 you can now add additional information to the pull request merge commit message. This allows you to provide a more descriptive change history in your repositories.



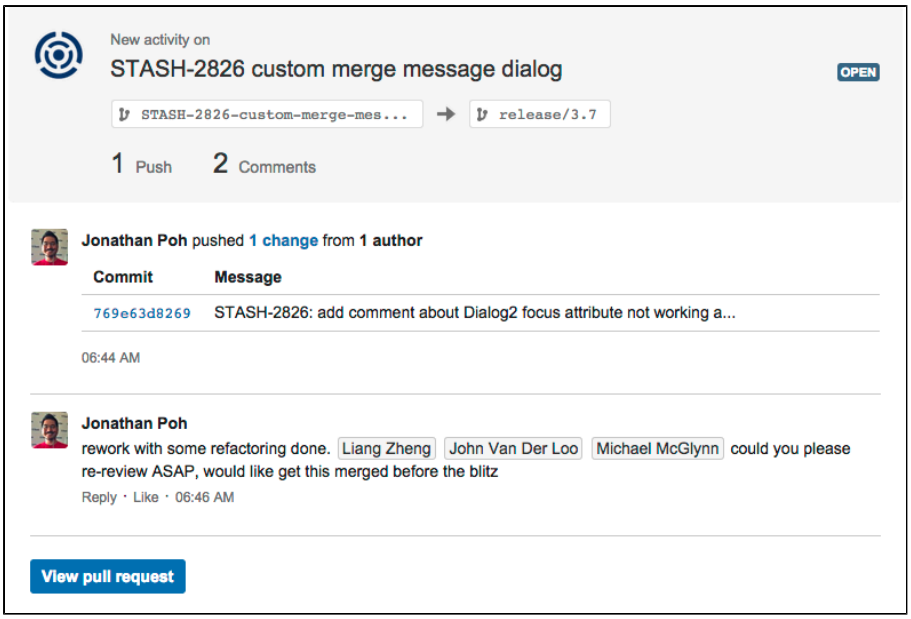
The text you add appears between the subject line and the log lines that Stash and Git generate.

Read more about pull requests in Stash...

Like and reply to comments... directly from email notifications

We've made the Stash email notifications for your team's activity around pull requests more useful. As well as being able to go directly to a colleague's comment or jump to the pull request in Stash, you can:

- Like a comment directly from the email – you get taken to the comment in Stash that you've just liked.
- Reply to a comment – you get taken to a comment dialog where you can just start typing.



Furthermore, comment and pull request authors get notified when their comments get liked.

Improved install and upgrade experience

We've reworked the Stash installer to clarify the differences between the install and upgrade paths. In particular, when upgrading Stash, you're asked for the location of your existing Stash home directory.

Change log

This section will contain information about the Stash 3.7 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, check the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.7.x releases, and are ordered by votes received.

5 May 2015 - Stash 3.7.4

T	Key	Summary
	STASH-7337	Throttling of hosting operations over SSH is broken

1 issue

23 April 2015 - Stash 3.7.3

T	Key	Summary
	STASH-7217	Incorrect nested home directory detection in installer
	STASH-7157	Installer fails to launch Stash (via script, not service) if JAVA_HOME isn't defined
	STASH-6964	Stash should not attempt to authenticate to Crowd with a blank username

3 issues

26 March 2015 - Stash 3.7.2

T	Key	Summary
	STASH-7195	JIRA 6.4 as a Crowd Server does not work with Stash 3.4.3 or higher
	STASH-7193	PullRequestMergedEvent is no longer asynchronous
	STASH-7142	hazelcast.group.name/password in distributions is not a placeholder
	STASH-5505	"Git not found" i18n message fails to load correctly during setup

4 issues

11 March 2015 - Stash 3.7.1

T	Key	Summary
	STASH-4128	REST API to create and delete tags
	STASH-7106	Remember-me authentication sometimes doesn't work
	STASH-7053	Windows service not defining -XX:MaxPermSize is causing Stash to get stuck on startup

3 issues

24 February 2015 - Stash 3.7.0

T	Key	Summary
	STASH-2826	A user can edit the commit message when merging a pull request
	STASH-4117	Provide authentication dates on the administration GUI
	STASH-3536	Format commit messages in the commit list the same as git log --oneline
	STASH-3424	Disable "Change password" field from admin and user page when delegated authentication is used
	STASH-7058	Cannot remove description of PR
	STASH-2744	Pull Request race condition between push and attributed activity author
	STASH-4413	Unique key constraints not working for Embedded Crowd table cwd_membership
	STASH-4270	No hooks are triggered when user creates Git branch via Stash UI
	STASH-3515	Commit messages should be monospaced
	STASH-7052	"User not permitted message" when viewing a pull request
	STASH-7047	Users list in administration fails with 500 status after upgrade to Stash 3.5
	STASH-6839	Stash migration to Postgres stops
	STASH-3352	Detect that Stash is unable to execute its hooks scripts and do something about it
	STASH-6866	"Diff to previous" in source view errors on binary files
	STASH-5449	Change our messaging on the installer on 'Select Stash Home' screen
	STASH-5252	Error when selecting Stash on installer using Start menu folder screen during upgrade
	STASH-4269	No hooks are triggered on pull request source branch deletion
	STASH-6961	Add support for PostgreSQL 9.4
	STASH-5465	Backup client should disable third party plugins before starting backup
	STASH-7084	DevSummary doesn't handle JIRA keys with thousands of commits associated with it

Showing 20 out of 40 issues

Stash 3.6 release notes

13 January 2015

With today's release of Stash 3.6, we've resolved over 30 public issues, made a range of improvements, and given the Stash experience a good polish.

If you are upgrading from an earlier version of Stash, check the Stash upgrade guide. There are no changes for Stash 3.6 in the API changelog.

The Stash 3.6 changelog is at the bottom of this page.

Create a branch starting from a tag

Stash now allows you to create a new branch directly from a tag that you select in the 'Create branch' screen. Read more about using branches in Stash...

MariaDB is now supported

Stash now supports MariaDB 5.5. See Connecting Stash to MySQL for more information.

SMTPS is now supported

Stash admins can now configure Stash to use SMTPS when sending email notifications. See Setting up your mail server.

Accessibility in Stash

We've made Stash more accessible for all individuals. To learn more, go to https://www.atlassian.com/accessibility.

Improved syntax highlighting for markdown

Stash now applies syntax highlighting to fenced code blocks in the markdown in comments, READMEs and pull request descriptions. See Markdown syntax guide.

Deprecated support for Internet Explorer 9

Support for IE 9 is deprecated, and will be removed in Stash 4.0. See End of support announcements for Stash.

Change log

This section will contain information about the Stash 3.6 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, check the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.6.x releases.

27 January 2015 - Stash 3.6.1

T	Key	Summary
✓	STASH-6960	Reduce overly-aggressive error logging for Git HTTP hosting
●	STASH-7125	Comment tooltip sticking when using keyboard shortcuts to go to next file
●	STASH-6972	An error occurred '/usr/bin/git rev-list --format=%H%x02%P%x02%aN%x02%aE%x02%at%n%B%n%x03END%x04 --no-walk

		8a1b904f0fa31dd32305c9c219dcf03cc11239a5 8a1b904f0fa31dd32305c9c219dcf03cc11239a5^@ --' exited with code 141
	STASH-6959	RepositoryService.findByOrigin doesn't order the returned forks

4 issues

13 January 2015 - Stash 3.6.0

T	Key	Summary
	STASH-6954	Block the creation of a user if sending a password-reset email would be ineffective
	STASH-5480	golang support: 'go get' fails to pull repository over ssh
	STASH-4012	Add support for MariaDB to Stash
	STASH-3782	Syntax highlighting for fenced code blocks in markdown files
	STASH-6946	Spurious new WARNING: Problem with directory [/opt/atlassian/stash/3.5.1/bin/lib], exists: [false], isDirectory: [false], canRead: [false] when stopping Stash
	STASH-6941	Application Navigator cannot be triggered using keyboard
	STASH-6912	WARNING: Problem with directory [/sites/atlassian/stash/3.5.0/bin/lib], exists: [false], isDirectory: [false], canRead: [false]
	STASH-6902	File browser is missing text focus outline when using keyboard navigation
	STASH-6901	Missing 'skip navigation' links
	STASH-6900	table headers missing scope attribute
	STASH-6895	Add accessibility text to icon-only buttons
	STASH-6894	IMG tags missing alt attribute
	STASH-6885	Invalid (non-ASCII) character in setenv.sh
	STASH-6875	scm-tags plugin leaves stale indexes on shutdown
	STASH-6868	Default executor.max.threads may be too small for larger instances
	STASH-6830	If a merge check throws an exception, the merge is still allowed to proceed
	STASH-6819	BranchInformationService is not available for plugins
	STASH-6773	Avatar picker doesn't work in IE9
	STASH-6753	Meta tag "activeTab" not working anymore for user profiles in Stash 3.5.0
	STASH-5538	Right clicking "open in new tab" on a different commit from a file's history drop-down results in a new tab with the *same* commit being opened.

Showing 20 out of 31 issues

Stash 3.5 release notes

25 November 2014

Introducing Stash 3.5

Today we're proud to announce the release of Stash 3.5.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

Please also check the End of support announcements for Stash. Add-on developers will be interested in the API changelog.

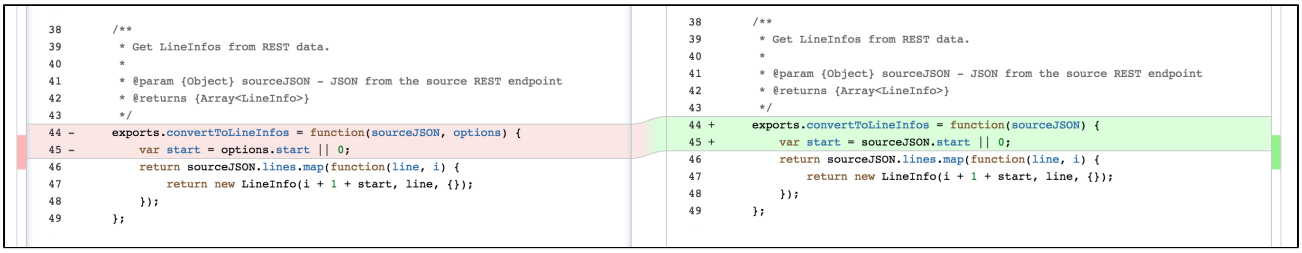
The Stash 3.5 changelog is at the bottom of this page.

Easily browse code in side-by-side diffs and the Source view

With Stash 3.5 you now get syntax highlighting in side-by-side diffs (although not the unified diff view) for pull requests and commits, and in the Source view, so the code is much easier to read.

We've taken the opportunity to add a range of new language mappings as well, including for Erlang, Perl, Python, Ruby and Puppet.

Read more about syntax highlighting in Stash...



Note that the highlighting engine in Stash has been changed from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is now deprecated, and will be removed in Stash 4.0. See Syntax highlight changes for information about how to migrate any custom mappings for HighlightJS that you may have made.

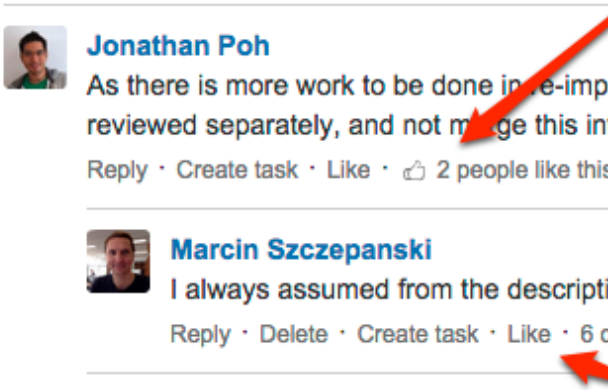
Read more about pull requests in Stash...

Amplify review feedback with comment likes

In another step towards better team collaboration, we've added comment likes. This might sound like a gimmick, but we think it's got real value. We've found that liking a comment is a simple yet strong way for pull request participants to *amplify review feedback* – it's a quick way of saying "also consider this person's feedback".

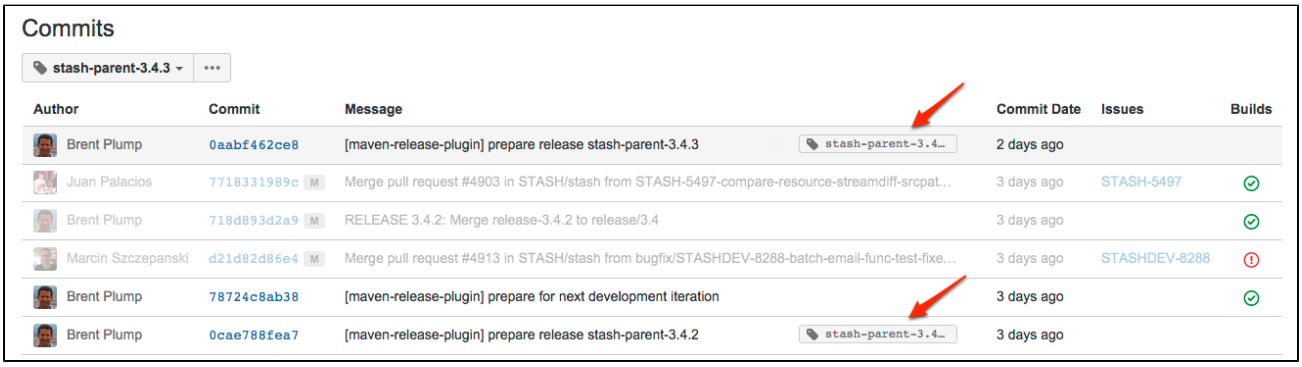
Whether it's a compliment or a suggestion, you can show your support without having to write a "+1" or "I agree" comment.

Read more about discussing pull requests in Stash...



Quickly see work done by using tags

Tags have long been used to mark important points in a repository's history, such as a milestone or a release. To help you make use of tags, Stash now displays them in the Commits list:



You can quickly see the work done since you tagged a commit, and easily find that commit by its tag.

Small improvements

Deprecated support for Java 7

Support for Java 7 is deprecated, and will be removed in Stash 4.0. See End of support announcements for Stash.

Deprecated support for older Git versions

Support for versions of Git earlier than v1.8 on the server is deprecated and will be removed in Stash 4.0. See End of support announcements for Stash.

Change log

This section will contain information about the Stash 3.5 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.5.x releases.

18 December 2014 - Stash 3.5.1

T	Key	Summary
	STASH-3238	High availability/failover
	STASH-2639	Distribute load over multiple servers
	STASH-6823	email password ignored/rewritten after pressing save button
	STASH-6820	Find highlight is broken
	STASH-6774	Cleaning up users fail if there is a backlog of more than 100 users
	STASH-5522	Accessing the port 5701 creates Hazelcast errors

6 issues

25 November 2014 - Stash 3.5.0

T	Key	Summary
	STASH-4846	Support for Git 2.0.x
	STASH-4840	Support for YAML syntax highlighting
	STASH-4630	Support syntax highlighting in side-by-side diff
	STASH-4076	Add "Like" button on events
	STASH-2795	Show tags in commit listing
	STASH-5530	Pull requests: Add support for handling "submodule" merge conflicts
	STASH-5459	Optimization: Avoid serialization and deserialization of session data by default
	STASH-5458	o.s.s.w.a.s.ChangeSessionIdAuthenticationStrategy Your servlet container did not change the session ID when a new session was created. You will not be adequately protected against session-fixation attacks
	STASH-6928	Installing Stash on OS X on Java 1.6 doesn't place JRE on installation directory
	STASH-5504	Crowd synchronization can fail processing groups
	STASH-5485	Hazelcast webfilter could not be found WARN messages
	STASH-5468	Branching model UI bug - auto-merge toggle can incorrectly be in disabled state
	STASH-5452	Stash OS X installer needs to be signed with a Developer ID
	STASH-5428	Stash fails to connect to User Directory behind a reverse proxy
	STASH-5404	Branch name folder/prefix not removed when changing repo
	STASH-5393	/rest/api/1.0/repos returning duplicate/missing results based on limit used
	STASH-5389	start-stash.sh script preserves environment variables with -m flag causing permission issues
	STASH-5381	Fork sync deletes branches which are in a pull request
	STASH-5372	Mail sever configuration page sends mail server password back in the html
	STASH-5362	Stash email settings fields can be inadvertently be populated by browser with user login details - security issue

Showing 20 out of 32 issues

Stash 3.4 release notes

21 October 2014

Introducing Stash 3.4

Today we're delighted to announce the release of Stash 3.4.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

Please also check the End of support announcements for Stash. Add-on developers will be interested in the API changelog.

The Stash 3.4 changelog is at the bottom of this page.

Batched email notifications

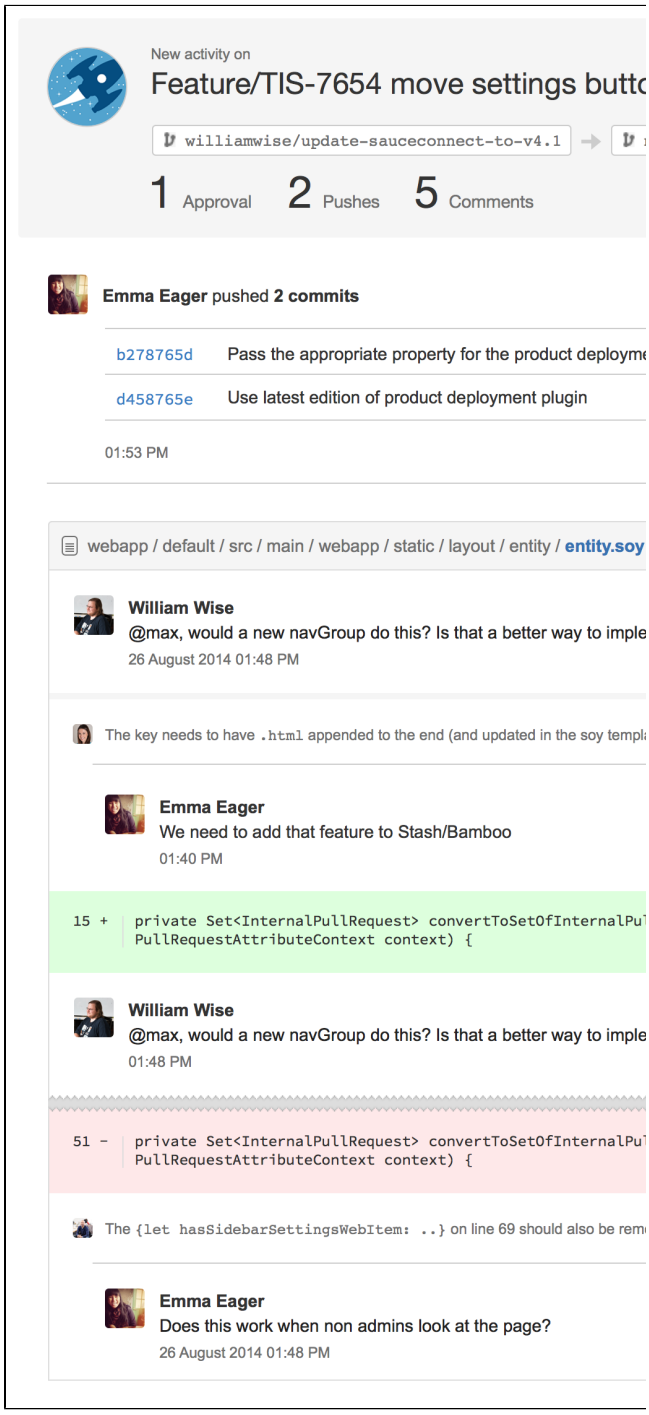
With Stash 3.4, we've taken a scalpel to our notifications engine and added email batching. What's more, we've made the notification emails more informative by putting the most important stuff at the top.

Now, *notifications will be aggregated* for each pull request and then emailed out for each recipient. The batch gets sent if things go quiet for a while (10 mins by default), or when the oldest notification gets 'stale' (30 mins by default), whichever comes first.

We think this is such a great improvement that we've made it the default, but you can change it back, in your personal account settings, if you prefer the rapid-fire pull request notifications.

Stash admins can control the batching behaviour using system properties.

Read more about notifications in Stash...



Small improvements

JMX support

Stash 3.4 introduces new settings in `setenv.bat` and `setenv.sh` to simplify enabling JMX.

When JMX is enabled via `setenv.bat` or `setenv.sh`, administrators can now set `jmx.enabled=true` in `stash-config.properties`. When this is enabled, Stash will publish details about itself and its major libraries, including:

- BoneCP, publishing under `com.jolbox.bonecp`, provides database connection pool statistics
- Hazelcast, publishing under `com.hazelcast`, provides clustering details for Stash Data Center
- Hibernate, publishing under `org.hibernate.core`, provides statistics for entities, queries and cache hits and misses

In addition, for those who wish to monitor Stash's "scm-command" and "scm-hosting" tickets, when `jmx.ena bled=true` is set those counts are published under `com.atlassian.stash` on a Tickets entry. For both

"scm-command" and "scm-hosting" JMX provides details on:

- How many tickets are configured
- How many tickets are available
- The number of requests currently queued for a ticket
- The timestamp at which the *oldest* waiter queued for a ticket
- The timestamp of the most recently rejected ticket

Support for Microsoft Office and OpenOffice/LibreOffice MIME types

We've extended the range of supported MIME types to include the Microsoft Office and OpenOffice document and template types (i.e. `docx`, `dotx`, `pptx`, `potx`, `ppsx`, `xlsx`, `xltx`, `odc`, `otc`, `odb`, `odg`, `otg`, `odf`, `otf`, `odi`, `oti`, `odp`, `otp`, `ods`, `ots`, `odt`, `ott`, `odm`, `oth`).

Aggregated group membership option for multiple user directories

We've added the option to 'blend' group memberships when a user or group is defined in two or more user directories. This is a change to how multiple directories were handled in previous versions of Stash. See the Stash upgrade guide for information about the impact of this change.

Read more about using multiple directories...

Disabling pull request assistive URLs in the console

As of Stash 3.3, the console displays the URL to the 'Create pull request' screen in Stash after you push a new branch (or a branch with no pull requests yet). A Stash admin can disable this behaviour as follows: click **Manage add-ons** in the Stash admin area, filter for 'bundled hooks' and click that, then disable the 'print-branch-links-hook' module.

Change log

This section will contain information about the Stash 3.4 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.4.x releases.

21 December 2014 - Stash 3.4.5

T	Key	Summary
	STASH-6928	Installing Stash on OS X on Java 1.6 doesn't place JRE on installation directory
	STASH-6774	Cleaning up users fail if there is a backlog of more than 100 users
	STASH-5522	Accessing the port 5701 creates Hazelcast errors
	STASH-5504	Crowd synchronization can fail processing groups

4 issues

Stash 3.4.4 was an internal release.

19 November 2014 - Stash 3.4.3

T	Key	Summary
	STASH-5497	Comparing branches/tags does not diff copied/renamed files correctly
	STASH-5485	Hazelcast webfilter could not be found WARN messages
	STASH-5468	Branching model UI bug - auto-merge toggle can incorrectly be in disabled state

	STASH-5428	Stash fails to connect to User Directory behind a reverse proxy

4 issues

Stash 3.4.2 was an internal release.

7 November 2014 - Stash 3.4.1

T	Key	Summary
	STASH-5459	Optimization: Avoid serialization and deserialization of session data by default
	STASH-5458	o.s.s.w.a.s.ChangeSessionIdAuthenticationStrategy Your servlet container did not change the session ID when a new session was created. You will not be adequately protected against session-fixation attacks
	STASH-5399	Increase the default minimum free disk space for scm-cache to prevent the disk from filling up
	STASH-5461	Transactional deadlock when setting global user / group permissions
	STASH-5446	Hunk Map no longer appears on a diff since 3.4.0
	STASH-5440	Clone URLs are generated using user slugs instead of usernames
	STASH-5406	Javascript Errors when viewing inbox
	STASH-5394	Batched e-mails are missing localizations other than English
	STASH-5393	/rest/api/1.0/repos returning duplicate/missing results based on limit used
	STASH-5385	Stash doesn't highlight changes on vertical bar (hunk map) in pull request diff view
	STASH-5381	Fork sync deletes branches which are in a pull request
	STASH-4881	Regression: Middle click to open in new tab doesn't work for branches in the branch selector
	STASH-4544	Refreshing Branch Permissions page after editing permission, re-attempts the change request

13 issues

21 October 2014 - Stash 3.4.0

T	Key	Summary
	STASH-5308	Stash should save a draft description when editing a pull request
	STASH-5013	Add these mime types for Stash
	STASH-2839	Don't send a notification on every pull comment immediately
	STASH-5295	Add Microsoft Office 2007 and OpenOffice MIME Types
	STASH-5457	Incorrect class name in event listener plugin developer docs
	STASH-5378	"Copy User on Login" setting is not saved properly when saving Delegated LDAP settings
	STASH-5350	Strip query params from referer before logging it in access logs.
	STASH-5344	Persistent XSS on a forked repository page
	STASH-5338	When a commit message is the ETX character, commit lists cannot be retrieved.
	STASH-5337	Landing page uses UK version of customize
	STASH-5330	Errors deleting repos in parallel
	STASH-5307	HipChat repository hook icon is broken
	STASH-5293	Persistent xss in comments caused by lacking escaping/encoding in ChangesetMarkupRenderer.java
	STASH-5280	Analytics tries to make downloads from contacts btf-analytics.s3.amazonaws.com although switched off or privacy policy accepted
	STASH-5269	IllegalStateException in DevSummarySupportInfo

STASH-5255	The task count not updated on the PR summary, even after the tasks list is open
STASH-5246	Branches should not auto merge from release branches to the production branch
STASH-5182	Can't cancel backups through web interface
STASH-5181	Hipchat token unauthorized if hipchat server url has a trailing forward slash
STASH-5086	Pull request list - reviewers no longer sorted (random on every page load)

Showing 20 out of 28 issues

Stash 3.3 release notes

10 September 2014

Introducing Stash 3.3

Today we're very pleased to announce the release of Stash 3.3.

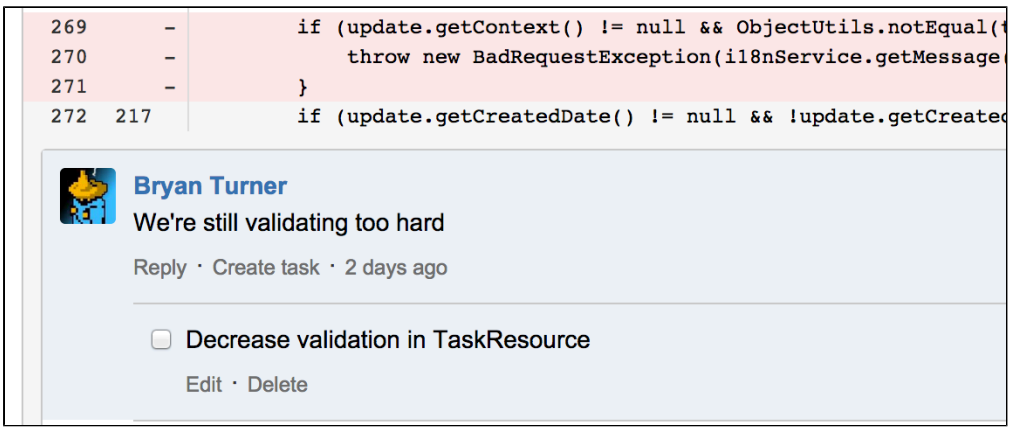
If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

Please also check the End of support announcements for Stash. Add-on developers will be interested in the API changelog.

The Stash 3.3 changelog is at the bottom of this page.

Tasks for pull requests

You can now use tasks in Stash to express and track actions identified during a review. You simply attach tasks to the comments you add to the pull request. We think tasks will make it much easier for developers to see what still needs to be done, and for review owners to keep track of which pull requests need attention before they can be merged.



Read more about tasks for pull requests.

Small improvements

Tomcat 8

Stash 3.3 now bundles Tomcat 8.0.9. Tomcat 8 implements Servlet 3.1, and also switches from using blocking I/O to non-blocking I/O by default. Non-blocking I/O improves Stash's ability to scale.

Note that Stash will still run if deployed to Tomcat 7 until Stash 4, at which time a Servlet 3.1-compliant container, like Tomcat 8, will be required.

Pull request URLs are displayed in the console after pushing

To save you time, the console now displays the URL to the 'Create pull request' screen in Stash after you push a new branch (or a branch with no pull requests yet):

```
rstocker@rstocker ‹ ABC-123-fix-bug › : ~/Projects/stash/repos/rep_1
[0] % git push -u origin HEAD
Password for 'http://admin@localhost:7990':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 263 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
remote: Create pull request for ABC-123-fix-bug:
remote:   http://localhost:7990/stash/projects/PROJECT_1/repos/rep_1/compare/commits?sourceBranch=refs/heads/ABC-123-fix-bug
remote:
To http://admin@localhost:7990/stash/scm/project_1/rep_1.git
 * [new branch]      HEAD -> ABC-123-fix-bug
Branch ABC-123-fix-bug set up to track remote branch ABC-123-fix-bug from origin.
```

You'll find that the branch and repository are already correctly selected in Stash, to make life easier still.

When pull requests already exist for the branch, links to those in Stash are displayed instead.

A Stash admin can disable this behaviour as follows: click **Manage add-ons** in the Stash admin area, filter for 'bundled hooks' and click that, then disable the 'print-branch-links-hook' module.

Change log

This section will contain information about the Stash 3.3 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.3.x releases.

21 December 2014 - Stash 3.3.5

T	Key	Summary
Bug	STASH-6928	Installing Stash on OS X on Java 1.6 doesn't place JRE on installation directory
Bug	STASH-6774	Cleaning up users fail if there is a backlog of more than 100 users
Bug	STASH-5522	Accessing the port 5701 creates Hazelcast errors
Bug	STASH-5497	Comparing branches/tags does not diff copied/renamed files correctly
Bug	STASH-5485	Hazelcast webfilter could not be found WARN messages
Bug	STASH-5468	Branching model UI bug - auto-merge toggle can incorrectly be in disabled state

6 issues

Stash 3.3.4 was an internal release.

5 November 2014 - Stash 3.3.3

T	Key	Summary
Improvement	STASH-5459	Optimization: Avoid serialization and deserialization of session data by default
Improvement	STASH-5458	o.s.s.w.a.s.ChangeSessionIdAuthenticationStrategy Your servlet container did not change the session ID when a new session was created. You will not be adequately protected against session-fixation attacks
Improvement	STASH-5399	Increase the default minimum free disk space for scm-cache to prevent the disk from filling up
Bug	STASH-5461	Transactional deadlock when setting global user / group permissions

	STASH-5393	/rest/api/1.0/repos returning duplicate/missing results based on limit used
	STASH-5381	Fork sync deletes branches which are in a pull request
	STASH-5344	Persistent XSS on a forked repository page
	STASH-5338	When a commit message is the ETX character, commit lists cannot be retrieved.
	STASH-5330	Errors deleting repos in parallel
	STASH-5181	Hipchat token unauthorized if hipchat server url has a trailing forward slash
	STASH-4680	User unable to sync LDAP/Active Directory after user rename if username already exists in another directory

11 issues

9 October 2014 - Stash 3.3.2

T	Key	Summary
	STASH-5307	HipChat repository hook icon is broken
	STASH-5293	Persistent xss in comments caused by lacking escaping/encoding in ChangesetMarkupRenderer.java

2 issues

24 September 2014 - Stash 3.3.1

T	Key	Summary
	STASH-5487	Opening task list requires all comments to be loaded
	STASH-5280	Analytics tries to make downloads from contacts btf-analytics.s3.amazonaws.com although switched off or privacy policy accepted
	STASH-5269	IllegalStateException in DevSummarySupportInfo
	STASH-5246	Branches should not auto merge from release branches to the production branch
	STASH-5234	Fork syncing is enabled, but getting error message
	STASH-5227	Analytics broken in Stash 3.3
	STASH-5221	Missing German translations for user/group permissions
	STASH-5216	Tasks list dialog is empty
	STASH-5182	Can't cancel backups through web interface
	STASH-4701	Performance empty pull request rescope activities is very poor on MySQL

10 issues

10 September 2014 - Stash 3.3.0

T	Key	Summary
	STASH-5082	"service atlstash status" is not supported
	STASH-5000	Automatically merge hotfixes targeting my production branch to my develop branch as well
	STASH-4865	Show link for creating/opening pull request when pushing from console
	STASH-4767	Reposition the repository settings button for easier findability
	STASH-4693	Stash should remember side-by-side diff preference
	STASH-3781	Task list for pull requests
	STASH-5197	Deleting users in parallel makes the license count go stale

	STASH-5159	"fatal: git upload-pack: not our ref" errors
	STASH-5154	Stash - ClassCastException in newly introducted analytics
	STASH-5150	Include stash-config.properties in support zip
	STASH-5148	Account creation notification is localized to the creators locale, instead of the servers default locale
	STASH-5123	Error 500 when adding invalid SSH key
	STASH-5083	The CAPTCHA service found itself in an awkward situation
	STASH-5076	Missing servlet tag in dev docs
	STASH-5041	IsPersonalProjectCondition/IsPersonalRepositoryCondition doesn't work from a plugin
	STASH-5007	Bruteforce Attack via Applinks Servlet
	STASH-4983	Commits disappear from commits list if clicking in back button after checking a commit.
	STASH-4980	Delegated LDAP Group Object Filter changes require restart to take effect, and are not validated
	STASH-4472	Unable to start Stash as git rev-list fails rescoping pull requests
	STASH-3158	Apostrophes in commit messages not rendered correctly in pull request emails.

20 issues

Stash 3.2 release notes

30 July 2014

Introducing Stash 3.2

Today, we're very pleased to announce the release of Stash 3.2.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

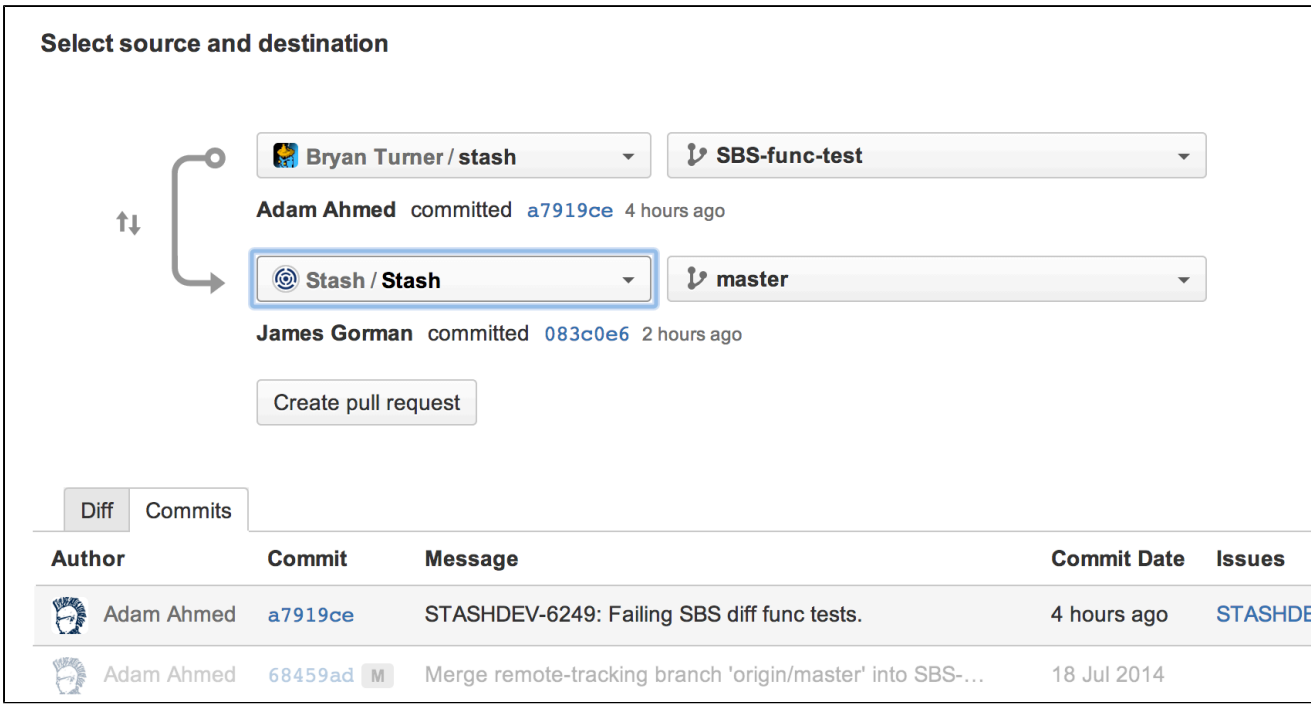
Please also check the End of support announcements for Stash.

Add-on developers will be interested in the API changelog.

The Stash 3.2 changelog is at the bottom of this page.

Improved workflow for creating pull requests

We've streamlined the pull request creation workflow in Stash so you can more easily preview your changes. Clicking **Create pull request** in the sidebar, or choosing **Compare** from the Actions menu (on the Source, Commits or Branches pages) lets you choose and compare branches, all on the same page:



Read more about using pull requests in Stash.

Improved navigation in diffs

In Stash 3.0 we added keyboard shortcuts, N (next) and P (previous), to move between hunks in a diff. In Stash 3.2 you can now use keyboard shortcuts to move between comments in a diff too. Now it's easy to jump between threads, just use Shift+N (next) and Shift+P (previous). Remember, you can press ? at any time to see all Stash keyboard shortcuts.

Small improvements

Improved experience for new users

We've added a landing page that Stash displays to a new user when they log in for the first time. The landing page provides a short introduction to the key features in Stash – this helps ensure new people joining your developer team have a basic understanding of what Stash can do for them, and saves administrators having to explain the basics every time. The page can be recalled later by choosing **Welcome to Stash** from the Help menu:

Product analytics disclosure

In the spirit of openness, we disclose that, as of version 3.2 Stash will collect user event data unless disabled by an administrator, per Atlassian's Privacy Policy. Analytics help us to determine the frequency of feature usage within the product, all towards improving Stash.

See Collecting analytics for Stash.

Change log

This section will contain information about the Stash 3.2 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.2.x releases.

5 November 2014 - Stash 3.2.7

T	Key	Summary
✓	STASH-5459	Optimization: Avoid serialization and deserialization of session data by default
✓	STASH-5458	o.s.s.w.a.s.ChangeSessionIdAuthenticationStrategy Your servlet container did not change the session ID when a new session was created. You will not be adequately protected against session-fixation attacks
●	STASH-5461	Transactional deadlock when setting global user / group permissions
●	STASH-5393	/rest/api/1.0/repos returning duplicate/missing results based on limit used
●	STASH-5344	Persistent XSS on a forked repository page
●	STASH-5338	When a commit message is the ETX character, commit lists cannot be retrieved.
●	STASH-5330	Errors deleting repos in parallel
●	STASH-5181	Hipchat token unauthorized if hipchat server url has a trailing forward slash
●	STASH-4680	User unable to sync LDAP/Active Directory after user rename if username already exists in another directory

9 issues

Note that Stash 3.2.6 was an internal release.

8 October 2014 - Stash 3.2.5

T	Key	Summary
●	STASH-5307	HipChat repository hook icon is broken
●	STASH-5293	Persistent xss in comments caused by lacking escaping/encoding in ChangesetMarkupRenderer.java

	STASH-5182	Can't cancel backups through web interface
	STASH-5155	LDAP Synchronization Time Refresh Broken due to JavaScript error
	STASH-4701	Performance empty pull request rescope activities is very poor on MySQL
	STASH-4472	Unable to start Stash as git rev-list fails rescoping pull requests

6 issues

26 August 2014 - Stash 3.2.4

T	Key	Summary
	STASH-5161	Backup client places stash-config.properties in STASH_HOME not STASH_SHARED_HOME for Stash 3.2+
	STASH-5158	Contents of stash-config.properties not restored by the backup client
	STASH-5138	Upgrading to UPM in Stash 3.2 results in com.atlassian.upm.api.util.Option$3 cannot be cast to com.atlassian.upm.api.util.Option

3 issues

Note that Stash 3.2.3 was an internal release.

15 August 2014 - Stash 3.2.2

T	Key	Summary
	STASH-5080	Setting page.max.changes doesn't work for diff tab of branch compare
	STASH-4835	Calling cancel(KeyedMessage) on a PullRequestOpenRequestedEvent does not display the KeyedMessage in the UI
	STASH-4517	Empty activity deletion deadlocks on MySQL
	STASH-4126	MergeException: New changes were pushed to master in PROJECT/repo while the merge was being performed

4 issues

Note that Stash 3.2.1 was an internal release.

29 July 2014 - Stash 3.2.0

T	Key	Summary
	STASH-4636	As an admin, I would like to configure my HipChat URL in HipChat Hook
	STASH-4031	Notify FishEye of new commits
	STASH-3620	Previous and next comment keyboard shortcuts
	STASH-3290	Allow Diff view during pull request creation
	STASH-6933	"Hazelcast instance is not active!" during shutdown
	STASH-5126	Missing unique constraint on id_sequence table
	STASH-5055	Pull request from fork to upstream repo failed
	STASH-5052	Commit messages are wrong when using Git 2.0.2 and 2.0.3
	STASH-5021	Persistent XSS through License Metadata
	STASH-5015	Reflected XSS in Stash Commit Warning Messages
	STASH-5002	Installing Stash as a service fails to set JAVA_HOME
	STASH-4990	"Reject force push" plugin doesn't block tag moves

STASH-4985	Download file links truncate filenames containing spaces in Firefox
STASH-4984	STASH Diff tool does not show changes for .F90 files
STASH-4974	Image Diffs do not offer Blend or Split
STASH-4966	Stash returns a 500 error through the API if you try to add an SSH access key to a repo while ssh access keys are disabled
STASH-4961	Creating branch in JIRA leads to invalid ref name in STASH.
STASH-4954	JS State API returns empty object in place of undefined.
STASH-4952	Spacing after tip
STASH-4924	Jira issue parsing in Stash commit

Showing 20 out of 25 issues

Stash 3.1 release notes

24 June 2014

Introducing Stash 3.1

Today we're pleased to announce the release of Stash 3.1.

Stash 3.1 brings nice enhancements to the pull request experience (code search, comment attachments, transcoding), and we're taking this opportunity to remind you about the Stash installer (available since Stash 3.0.4).

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

Please also check the End of support announcements for Stash.

Add-on developers will be interested in the API changelog.

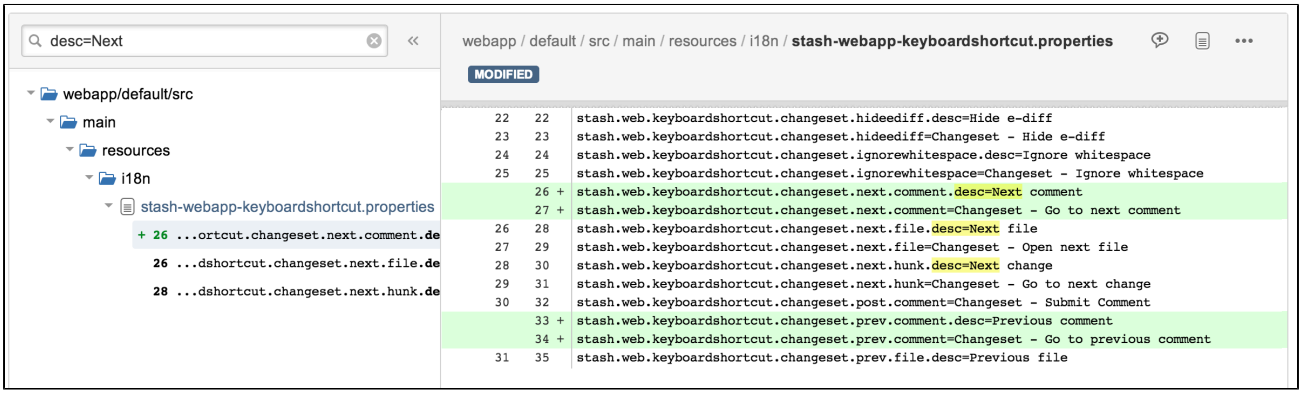
The Stash 3.1 changelog is at the bottom of this page.

Code search in pull request diffs

We've added the ability to search across all the files in a pull request, when viewing a pull request diff.

The search is applied to the diff and surrounding context lines. The search results are highlighted, as you'd expect, and you'll notice that the file tree on the left is updated to display the files and matching lines returned by the search.

Use the 'f' keyboard shortcut to quickly access code search when viewing a diff in a pull request. The usual 'j' (next) and 'k' (previous) keyboard shortcuts let you move between results. 'Esc' cancels the search.



Attach images to pull request comments

You can now drag and drop images, and other file types, in to your pull request comments and descriptions, so it's much easier for your team to discuss how that new UI should look. Alternatively, use the attachment button when editing the comment to upload a file from your file system, or use markup to add a resource, say from the web. Click on an image to see it at full size.

Stash administrators can use Stash config properties to control availability of this feature (`feature.attachments`) and to set the file size limit (`attachment.upload.max.size` – the default is 10 MB).

Stash installer

A Stash installer is now available (since Stash 3.0.4) for the Linux, Mac OS X and Windows operating systems. The installer can be downloaded as usual from http://www.atlassian.com/software/stash/download.

The installer will install a supported version of Java, if necessary, and on Linux and Windows systems, it provides the option to install Stash as a service. See Getting started.

The installer can be run in GUI, console or unattended modes. See Running the Stash installer for details.

Small improvements

Microsoft SQLServer

SQL Server 2014 is now supported. See Supported platforms.

Oracle

Oracle 12c is now supported. See Supported platforms .

Transcoding is now supported for diff views

Diffs in Stash now support encodings other than UTF-8, for example EUC-JP, GB18030, UTF-16 and UTF-32. This means that any file that displays correctly in the source view will now display nicely in a diff view. See Using diff transcoding in Stash.

See also STASH-3179 - Convert non-UTF8 files to UTF8 before generating the diff view CLOSED .

Change log

This section will contain information about the Stash 3.1 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.1.x releases.

5 November 2014 - Stash 3.1.7

T	Key	Summary
✓	STASH-5458	o.s.s.w.a.s.ChangeSessionIdAuthenticationStrategy Your servlet container did not change the session ID when a new session was created. You will not be adequately protected against session-fixation attacks
●	STASH-5344	Persistent XSS on a forked repository page
●	STASH-5338	When a commit message is the ETX character, commit lists cannot be retrieved.
●	STASH-4680	User unable to sync LDAP/Active Directory after user rename if username already exists in another directory

4 issues

Note that Stash 3.1.6 was an internal release.

7 October 2014 - Stash 3.1.5

T	Key	Summary
●	STASH-5293	Persistent xss in comments caused by lacking escaping/encoding in ChangesetMarkupRenderer.java
●	STASH-5062	When viewing the source tab of an issue in JIRA, it claims that stash returned an error
●	STASH-4701	Performance empty pull request rescope activities is very poor on MySQL
●	STASH-4472	Unable to start Stash as git rev-list fails rescoping pull requests
●	STASH-3831	padding/margin around mentions lozenges

5 issues

4 August 2014 - Stash 3.1.4

T	Key	Summary
●	STASH-5052	Commit messages are wrong when using Git 2.0.2 and 2.0.3
●	STASH-5021	Persistent XSS through License Metadata
●	STASH-5015	Reflected XSS in Stash Commit Warning Messages
●	STASH-5002	Installing Stash as a service fails to set JAVA_HOME
●	STASH-4984	STASH Diff tool does not show changes for .F90 files
●	STASH-4835	Calling cancel(KeyedMessage) on a PullRequestOpenRequestedEvent does not display the KeyedMessage in the UI

	STASH-4517	Empty activity deletion deadlocks on MySQL
	STASH-4126	MergeException: New changes were pushed to master in PROJECT/repo while the merge was being performed

8 issues

11 July 2014 - Stash 3.1.3

T	Key	Summary
	STASH-4966	Stash returns a 500 error through the API if you try to add an SSH access key to a repo while ssh access keys are disabled
	STASH-4932	Memory leak in ssh plugin

2 issues

Note that Stash 3.1.2 was an internal release.

1 July 2014 - Stash 3.1.1

T	Key	Summary
	STASH-4983	Commits disappear from commits list if clicking in back button after checking a commit.
	STASH-4974	Image Diffs do not offer Blend or Split
	STASH-4776	sshd sometimes fails key_verify

3 issues

24 June 2014 - Stash 3.1.0

T	Key	Summary
	STASH-4515	Show HTTP as the default when the user hasn't uploaded an SSH key
	STASH-3326	Upload images in pull request comment
	STASH-3101	Increase the number of projects initially shown on the project list from 25 to 100
	STASH-3082	markdown: support displaying images in Stash repo
	STASH-2841	Trigger bamboo builds when there are new commits on associated branches
	STASH-4810	Upgrade to Tomcat 7.0.54
	STASH-4892	Custom Navigation Plugin incompatible with Stash 3
	STASH-4891	Atlassian Development Toolbox incompatible with Stash 3
	STASH-4890	REST API Browser plugin incompatible to Stash 3
	STASH-4880	undefined displayed in ahead/behind tooltip instead of the name of the branch
	STASH-4804	Application Navigator Administration sets incorrect charset resulting in garbage when viewing in Japanese
	STASH-4790	Navbar "Learn more" keeps on appearing with IE 11
	STASH-4762	Relative links in base README.md incorrect
	STASH-4734	CSRF/XSRF vulnerability in UserResource.java [uploadAvatar, line 161]
	STASH-4670	Stash binds Ctrl-Shift-P which is a builtin Firefox keyboard shortcut
	STASH-4474	Changing groupname casing in external user management causes intermittent loss of group membership
	STASH-4400	Opening issue links in a new tab does not open the issue correctly in JIRA
	STASH-3635	Html in PR comments not encoded

18 issues

Stash 3.0 release notes

20 May 2014

Introducing Stash 3.0

Today we're delighted to announce Stash 3.0, the first new major release of Stash in over a year.

Stash 3.0 brings an overhaul of the product navigation, a new way to compare branches, internationalization, and a public JavaScript API for plugin developers. Note in particular that Java 6 is no longer supported, and that previously deprecated APIs have been removed (with possible consequences for plugin compatibility) – add-on developers will be interested in the API changelog.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

Please also check the End of support announcements for Stash.

The Stash 3.0 changelog is at the bottom of this page.

Webcam captured avatars

With Stash 2.12 we introduced custom avatar images, allowing you to upload an image to your profile. Now, with Stash 3.0, we've made it possible to use your webcam to capture an avatar image directly from your **Acc ount settings** in Stash. Looking good!

Sidebar redesign

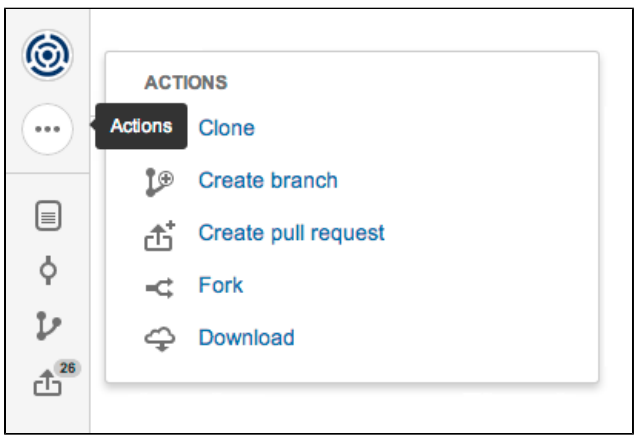
We've undertaken a major rework of the navigation for Stash 3.0.

Now, the Actions and Navigation items all live vertically in the lefthand sidebar, where it is much easier to find them. Note that **Files** has been renamed as **Source**.

.

Click anywhere in the sidebar, or press [, to collapse or expand it.

When the repositories sidebar is collapsed, the Actions items become available from the menu:



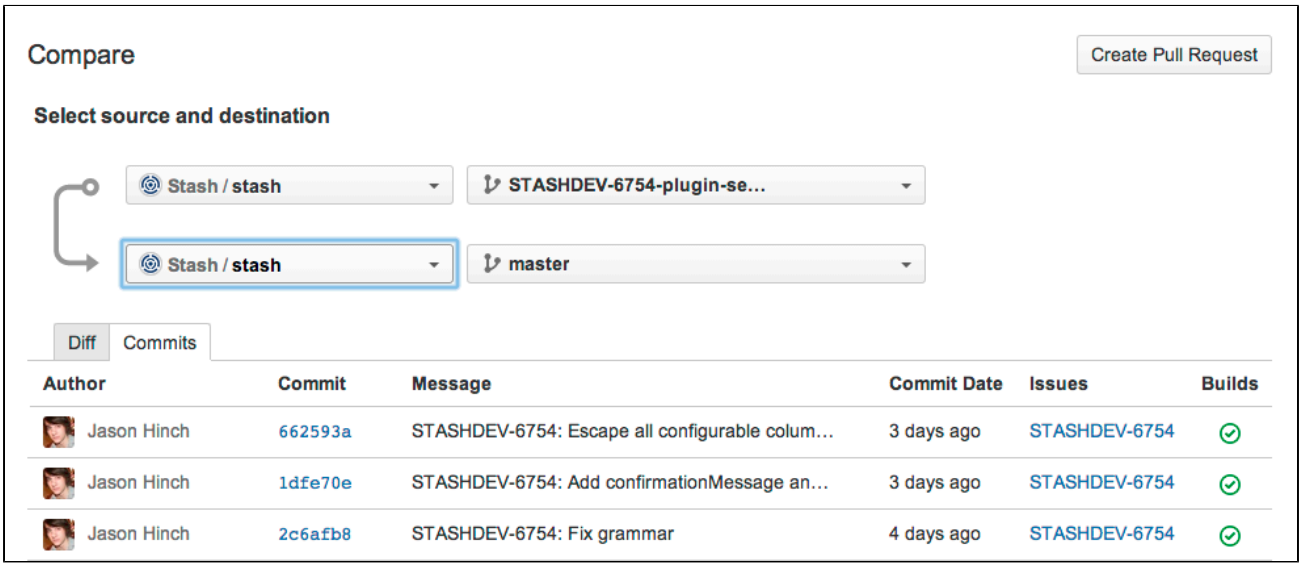
Branch comparisons

Now you can compare the changes between two branches, even before you start creating a pull request.

Simply click on a branch name in the **Branches** tab list or choose **Compare** from the branch actions menu on the Source, Commits or Branches screen. Then, use the project and branch pickers to choose the branches (that you have permission to see) to compare.

- The **Diff** tab shows the changes in the source branch that are not yet in the destination branch.
- The **Commits** tab shows commits on the source branch that have not yet been merged to the destination branch.

These views are very similar to those for pull requests in Stash, including the option for displaying side-by-side or unified diffs:



Furthermore, when you begin creation of a pull request from the Compare screen (using the **Create pull request** action in the sidebar), the branches that you are comparing are automatically used to populate the source and destination branches for the pull request.

Read more about pull requests in Stash...

Add-on and plugin compatibility in Stash 3.0

The interfaces in the Stash API for plugin developers that were deprecated in the Stash 2.11 and earlier releases have been removed in Stash 3.0. This means that, unless they have been updated to work with Stash 3.0, existing Stash add-ons (plugins) that use these interfaces **will not work with Stash 3.0** .

In particular, please note that your custom local plugins may be affected by these API removals when you upgrade to Stash 3.0. You will need to update your custom plugins if you want those to work with Stash 3.0.

See the Stash upgrade guide for details...

New Stash JavaScript API documentation

We've published documentation for the JavaScript API for use in building your Stash plugin's UI and browser behavior. We know that plenty of developers were already using these utilities, so we're pleased to provide support for them through our compatibility policy. And yes, we're going to be expanding this API over the next few releases.

Stash internationalization

Stash is now locale-aware and will be displayed in your preferred language (French, German or Japanese language packs are the first available).

Stash administrators can specify the default language at install time (in the Stash Setup Wizard), and can install additional language packs from the Atlassian Marketplace as these become available. See Managing add-ons.

You can set your preferred language in your Stash account settings – this overrides the default Stash

language setting, but depends on the available language packs that are installed.

Stash selects the display language for each user session by finding the first match from the installed language packs with first, the user's language preference in their Stash account settings, then the locale set in their browser, the default Stash language setting, the Java or operating system locale, or finally US English, in that priority order.

Support for Java 6 removed

As previously announced, support for Java 6 has been removed. Stash 3.0 requires at least Java 7.

Small improvements

Java 8 supported

Stash 3.0 adds support for Java 8.

Internet Explorer 11 supported

Stash 3.0 adds support for IE 11.

Support for Internet Explorer 8 removed

As previously announced, support for IE8 has been removed in Stash 3.0.

Change log

This section will contain information about the Stash 3.0 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 3.0.x releases.

7 October 2014 - Stash 3.0.8

T	Key	Summary
	STASH-5293	Persistent xss in comments caused by lacking escaping/encoding in ChangesetMarkupRenderer.java
	STASH-4858	Showing "Changed files" within a pull request diff is not working
	STASH-4701	Performance empty pull request rescope activities is very poor on MySQL
	STASH-4472	Unable to start Stash as git rev-list fails rescoping pull requests

4 issues

4 August 2014 - Stash 3.0.7

T	Key	Summary
	STASH-5052	Commit messages are wrong when using Git 2.0.2 and 2.0.3
	STASH-5021	Persistent XSS through License Metadata
	STASH-5015	Reflected XSS in Stash Commit Warning Messages
	STASH-5002	Installing Stash as a service fails to set JAVA_HOME
	STASH-4835	Calling cancel(KeyedMessage) on a PullRequestOpenRequestedEvent does not display the KeyedMessage in the UI
	STASH-4517	Empty activity deletion deadlocks on MySQL

	STASH-4126	MergeException: New changes were pushed to master in PROJECT/repo while the merge was being performed

7 issues

30 June 2014 - Stash 3.0.6

T	Key	Summary
	STASH-4974	Image Diffs do not offer Blend or Split
	STASH-4880	undefined displayed in ahead/behind tooltip instead of the name of the branch
	STASH-4776	sshd sometimes fails key_verify

3 issues

19 June 2014 - Stash 3.0.5

T	Key	Summary
	STASH-3082	markdown: support displaying images in Stash repo
	STASH-4810	Upgrade to Tomcat 7.0.54
	STASH-4804	Application Navigator Administration sets incorrect charset resulting in garbage when viewing in Japanese
	STASH-4790	Navbar "Learn more" keeps on appearing with IE 11
	STASH-4762	Relative links in base README.md incorrect
	STASH-4648	Switching default permission from "write" to "read" is broken
	STASH-4400	Opening issue links in a new tab does not open the issue correctly in JIRA

7 issues

3 June 2014 - Stash 3.0.4

T	Key	Summary
	STASH-3181	Create a Linux installer
	STASH-4777	Branch Compare: the target branch's repository is ignored
	STASH-4765	Redirect after login doesn't work with a hash in the URL
	STASH-4671	Switching to side-by-side diff continuously redirects anonymous users to login form

4 issues

Note that Stash 3.0.2 and Stash 3.0.3 were internal releases.

20 May 2014 - Stash 3.0.1

T	Key	Summary
	STASH-4572	Add a way to switch back from local avatar to gravatar
	STASH-4494	Add support for JRE 8
	STASH-4427	Side by side diff view: Keyboard shortcuts for jumping between diffs
	STASH-4421	Add Support for Internet Explorer 11
	STASH-4322	Settings page of repository loading for a long time on large repos

	STASH-3276	Compare and Diff for Branches
	STASH-4715	Basic authentication in the web UI leads to XSRF errors
	STASH-3582	Cancelable Permission Grant/Revoke events
	STASH-4789	'T' shortcut for doesn't highlight my own PRs like it used to
	STASH-4700	ContextualFormFragment#validate is never called for fork repository dialog
	STASH-4696	Memory leak when using a persistent SSH connections
	STASH-4694	OOME on bad Markdown (due to ```)
	STASH-4663	PermissionAdminService API does not grant permissions properly when used in a project permission revoked event
	STASH-4641	Heavy use of Trusted Apps appears to be causing products to slow
	STASH-4571	directives in stash-config.properties with trailing spaces at the end of the directive line are not parsed
	STASH-4559	The refsync plugin is not shutting down its thread pool
	STASH-4539	Navigating to the account page when the user is not logged in redirects to the project list
	STASH-4530	bash-only syntax used in setenv.sh
	STASH-4499	Diff scrolling glitches
	STASH-4217	SSH idle timeout is too large resulting in OutOfMemoryError when clients fail to terminate the session

Showing 20 out of 21 issues

Stash 2.12 release notes

25 March 2014

Introducing Stash 2.12

Today we're very pleased to announce Stash 2.12, which brings custom avatars, hunk maps for side-by-side diffs, enhanced SSH access keys, a new REST API for backing up and restoring Stash, and improvements to markdown rendering.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

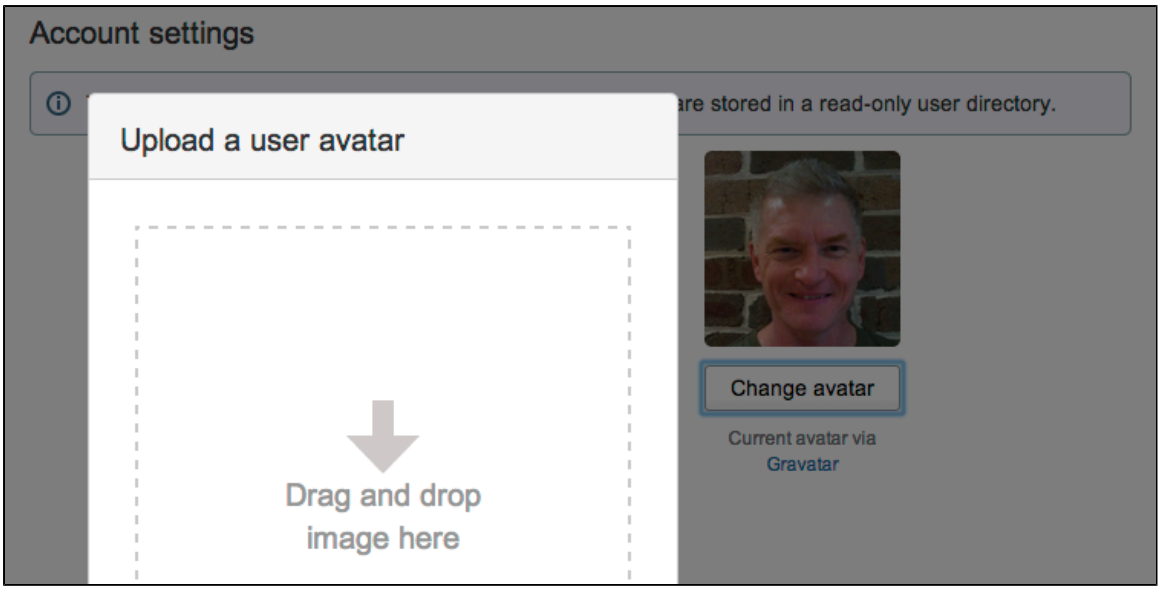
Please also check the End of support announcements for Stash.

Add-on developers will be interested in the API changelog.

The Stash 2.12 changelog is at the bottom of this page.

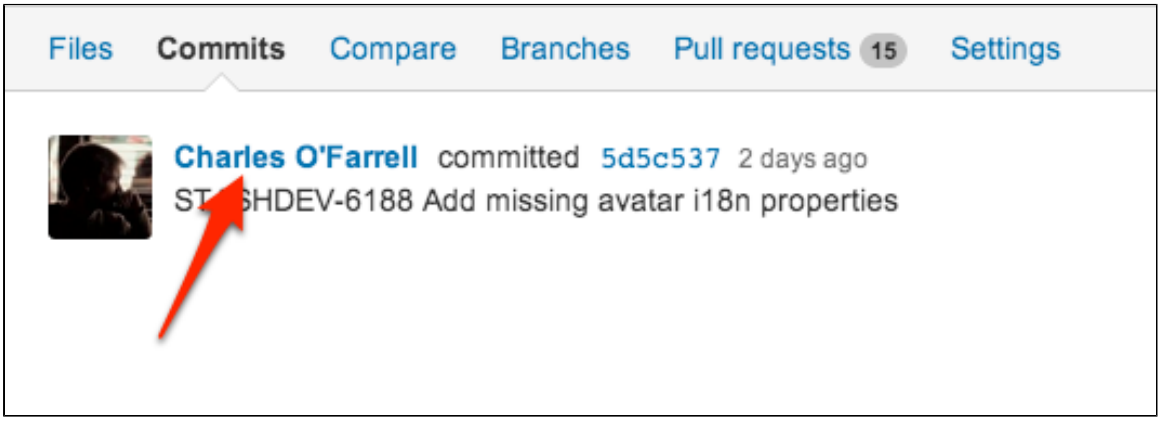
Custom avatar images

You can now upload a custom avatar image to your profile, which will replace your Gravatar image wherever that is used in Stash. Do this on the **Account settings** tab in your personal profile.



User name linking

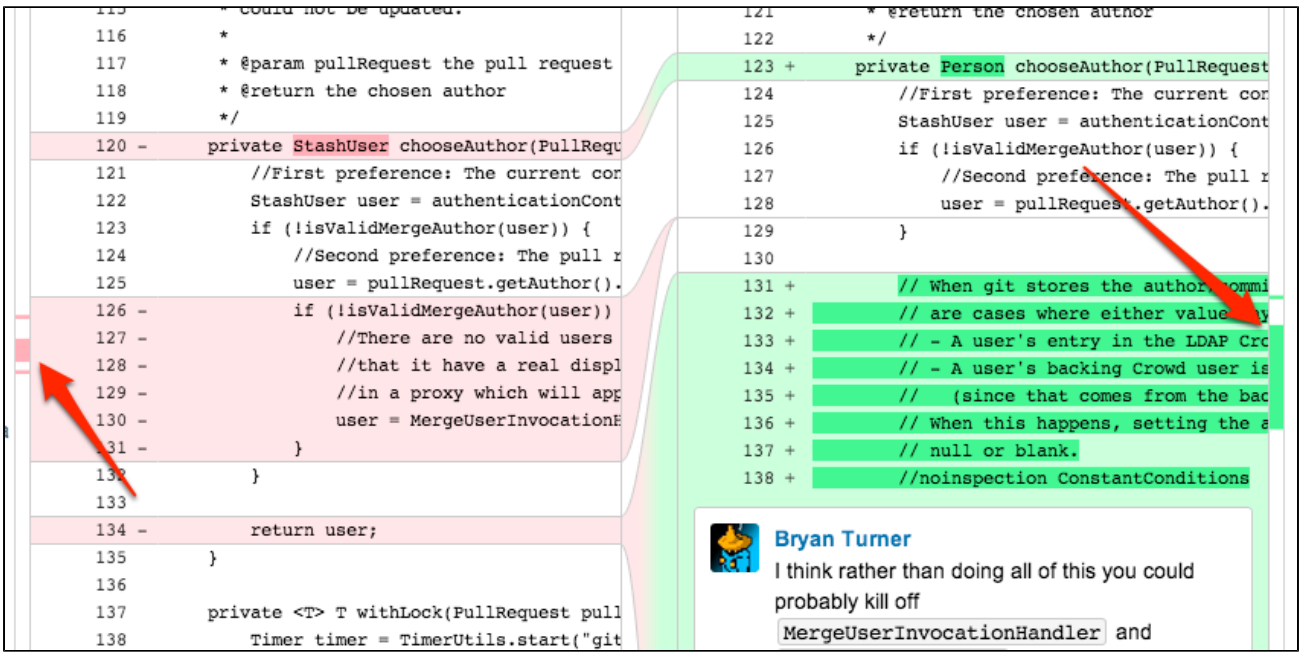
When you're logged in to Stash you'll notice that wherever commit author names are displayed (for example, on the commits list, and the blame view) they are now linked to the profile page of the author (if the author is also a Stash user). Authors are matched to Stash users via the author's email address, and the user's name in their Stash account is displayed instead of their author name.



Hunk maps

We've improved how you can use side-by-side diffs, first introduced with Stash 2.11.

Now we've added a 'hunk map' to side-by-side diffs, to make it easier for you to find where a file has been changed. Simply click on the hunk map in the left- or right-hand margin to scroll the view to that location:

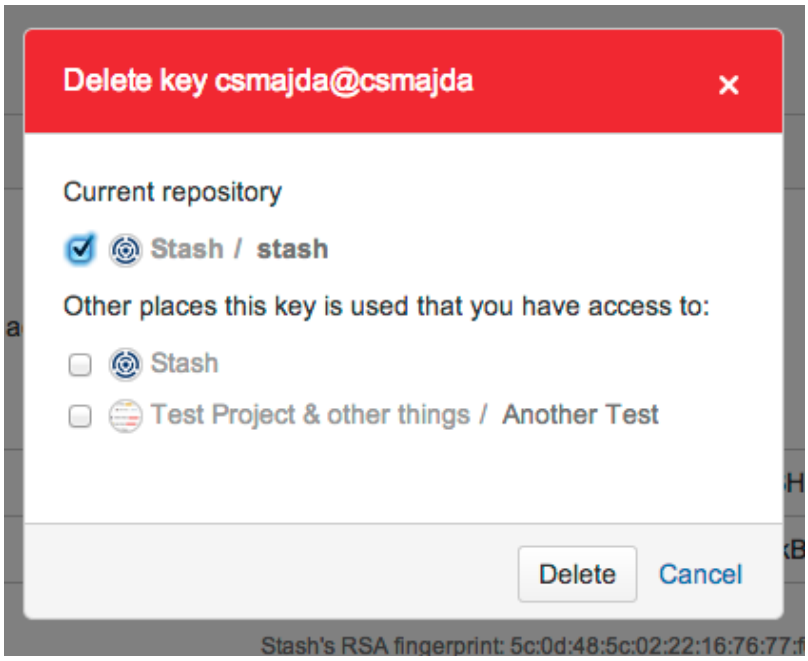


Access keys

We've extended how you can use SSH keys to secure Git operations for Stash repositories.

Firstly, system SSH access keys, which allow you to secure the Git operations performed by other systems on Stash repositories, can now be either *Read-only* or *Read / Write*. Now you can use operations like `git push` or `git merge` if you want the other system to merge successful feature branch builds to the default branch in the Stash repository, or so that deployments can be tagged.

Furthermore, we've provided greater visibility into where access keys are being used in Stash. When you want to delete a key, you now see a list of the other places where the key is being used, and can delete the key from those places as well:



The keys displayed in the list depend on your permissions to see those projects and repositories.

Read more about SSH access keys for system use...

DIY Backup

We've added to the backup and restore strategies available for use with Stash.

The new Stash DIY Backup is an alternative to the Stash Backup Client, and provides a REST API that allows you to:

- significantly reduce the downtime needed to create a consistent backup,
- use the appropriate industry-standard tools to back up your Stash file system and external database back end,
- automate the Stash back up process.

As an indication of the downtime that can be expected when using Stash DIY Backup, in our testing and internal use we have seen Stash downtimes of 7–8 minutes with repositories totalling 6 GB in size when

using the Backup Client, compared with downtimes of less than a minute when using the new DIY Backup with the same repositories.

Read more about a fully worked solution in Using Stash DIY Backup...

Small improvements

Toggle comment display

Use the Shift-C keyboard shortcut or the diff options menu to toggle comment display, either in a unified diff or a side-by-side diff. The icon in the left margin shows that comments are hidden:
.

Markdown rendering in source view

Markdown files are now rendered in the source view. The output handles images, links (including relative links) and table syntax.
.

Markdown table syntax in comments

Table syntax is now supported in comments and markdown files. For more details, see the markdown syntax guide.

Change log

This section will contain information about the Stash 2.12 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 2.12.x releases.

4 August 2014 - Stash 2.12.6

T	Key	Summary
	STASH-5021	Persistent XSS through License Metadata
	STASH-5015	Reflected XSS in Stash Commit Warning Messages

2 issues

18 June 2014 - Stash 2.12.5

T	Key	Summary
	STASH-4810	Upgrade to Tomcat 7.0.54
	STASH-4671	Switching to side-by-side diff continuously redirects anonymous users to login form

2 issues

22 May 2014 - Stash 2.12.4

T	Key	Summary
	STASH-4700	ContextualFormFragment#validate is never called for fork repository dialog

	STASH-4698	Stash not returning JIRA summary data when running JDK 1.6
	STASH-4696	Memory leak when using a persistent SSH connections
	STASH-4694	OOME on bad Markdown (due to ```)
	STASH-4640	page.max.directory.children does not show correct number of files on page load
	STASH-4217	SSH idle timeout is too large resulting in OutOfMemoryError when clients fail to terminate the session

6 issues

12 May 2014 - Stash 2.12.3

T	Key	Summary
	STASH-4683	"Remember-me" users forced to log in too often

1 issue

1 May 2014 - Stash 2.12.2

T	Key	Summary
	STASH-4322	Settings page of repository loading for a long time on large repos
	STASH-4571	directives in stash-config.properties with trailing spaces at the end of the directive line are not parsed
	STASH-4563	Typo in SSH Key Documentation
	STASH-4548	Markdown pages with inline markdown links and empty parentheses load indefinitely
	STASH-4499	Diff scrolling glitches

5 issues

2 April 2014 - Stash 2.12.1

T	Key	Summary
	STASH-4523	JiraIssueService.getIssuesForChangesets() fails when querying with a large number of changesets

1 issue

25 March 2014 - Stash 2.12.0

T	Key	Summary
	STASH-4422	Stash markdown relative links should work on branches
	STASH-4305	have stash recognize relative url's for git submodules
	STASH-3456	Toggle off comments in Pull Request Diff tab
	STASH-3395	Render .md files as markdown
	STASH-3235	Link committers to users
	STASH-3232	Add Table Support to Markdown
	STASH-2760	Allow Non-Gravatar Profile Picture
	STASH-4934	PullRequestRescopeListener throws PreAuthenticationFailedException on startup, causing startup failure
	STASH-4659	Stash performance degrades over time with Java 1.7.0_51

STASH-4502	Application Links -> Display URL Property not used for browser client links
STASH-4488	Commit graph cache incorrectly propagates exception
STASH-4454	JIRA issues dialog: JIRA keys with banned characters do not show
STASH-4450	Allow toggling of the authority part in clone URLs on RepositoryCloneLinksRequest
STASH-4432	Links to comments in emails no longer jump to the comment
STASH-4410	When cycling through files in the diff view quickly using j/k, the window scrollbar disappears and it's not possible to scroll the diff
STASH-4407	Diff scrolling 'jumps' near the top of the page.
STASH-4334	Projects with MVC as their key broken on some pages
STASH-4327	Using Gravatar leaks internal information to 3rd party
STASH-4250	REST json response does not match the configured Base URL
STASH-4228	Licensed user count cache should be repopulated in the background to avoid blocking request threads

Showing 20 out of 24 issues

Stash 2.11 release notes

25 February 2014

Introducing Stash 2.11

Today we're excited to announce Stash 2.11, which brings added support for code review workflows with side-by-side diffs and new ways to discuss code changes.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

Please also check the End of support announcements for Stash.

The Stash 2.11 changelog is at the bottom of this page.

Commit comments

Now you can get immediate feedback on changes in a particular commit, even if they're not part of a pull request. Commit discussions allow you to request feedback early on in the development cycle – not only to ensure you are on the right path, but also to speed up subsequent pull requests because the bulk of the changes have already been reviewed. Commit comments also let developers point out general issues in code or discuss ideas for improving code quality in future work.

You can comment directly on a line of code right in the diff, just as you've been able to do with diffs in pull requests. Just hover over the line of code, click the icon at the left (arrowed below), and enter your comment; @mention another Stash user in the comment, to get their attention.

.

As the author of the commit, or when you make a comment, you're automatically made a watcher, and get notified when others comment. You can watch, or unwatch, the commit at any time, of course.

File comments for pull requests and commits

Until now you've only been able to comment on particular lines of code in files under discussion in pull requests. But what if you wanted to comment about the whole file, or about a file without text, such as an image? Now you can!

Whether you're looking at a pull request or a file in a commit, just click the header icon (arrowed below, or use the 'M' keyboard shortcut) to add a comment about the whole file. You can do this in both a unified diff or a side-by-side diff (also new in this release - see below).

As with general or in-line comments, file comments on a pull request appear in the activity stream, so it's easy for others to see, and you can @mention other Stash users to really get their attention.

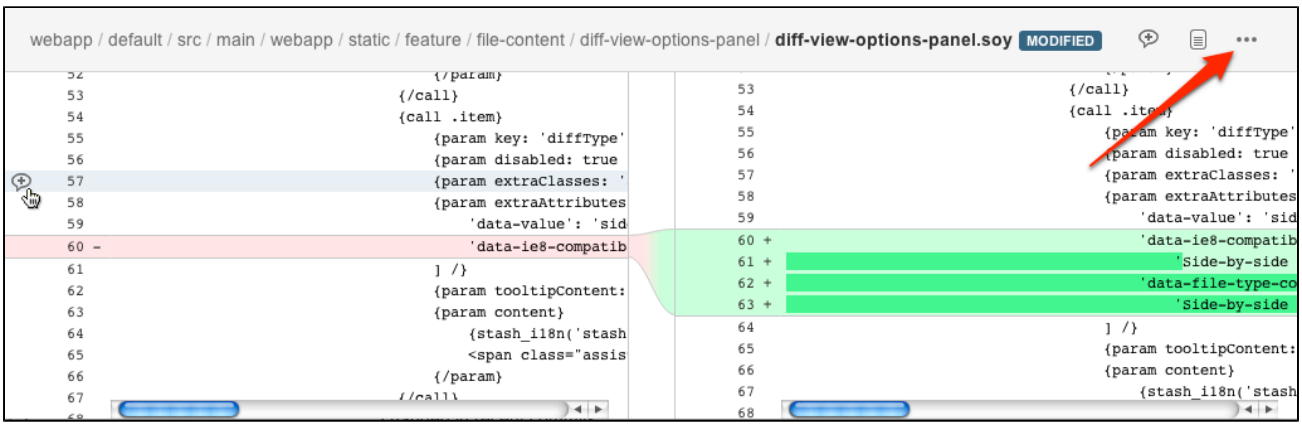
.

Read more about pull requests in Stash...

Side-by-side diffs

We added a new side-by-side view for when you're looking at diffs in both pull requests and commits – we think you'll find that this really improves your experience of using Stash for code reviews, and for viewing code changes in general.

Toggle between the unified diff and side-by-side diff views using the menu arrowed below. The side-by-side view is not available for added or removed files.



For a pull request, the lefthand panel shows the code in the target branch and the righthand panel shows what the result of merging will be. For a commit, the lefthand panel shows the previous revision and the righthand panel show the current revision.

As mentioned above, file comments and per-line comments are both available for the side-by-side view.

Small improvements

MySQL 5.6.16+ is now supported

Stash now supports versions of MySQL 5.6 from MySQL 5.6.16 on. See Supported platforms.

Lockout recovery process

We've made some changes to help admins who accidentally lock themselves out of Stash. See Lockout recovery process.

Change log

This section will contain information about the Stash 2.11 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 2.11.x releases. Note that Stash 2.11.0 and 2.11.1 were internal releases.

4 Aug 2014 - Stash 2.11.9

T	Key	Summary
	STASH-4671	Switching to side-by-side diff continuously redirects anonymous users to login form

1 issue

22 May 2014 - Stash 2.11.8

T	Key	Summary
	STASH-4696	Memory leak when using a persistent SSH connections
	STASH-4659	Stash performance degrades over time with Java 1.7.0_51
	STASH-4640	page.max.directory.children does not show correct number of files on page load
	STASH-4217	SSH idle timeout is too large resulting in OutOfMemoryError when clients fail to terminate the session

4 issues

1 May 2014 - Stash 2.11.7

T	Key	Summary
	STASH-4322	Settings page of repository loading for a long time on large repos
	STASH-4571	directives in stash-config.properties with trailing spaces at the end of the directive line are not parsed
	STASH-4499	Diff scrolling glitches

3 issues

1 April 2014 - Stash 2.11.6

T	Key	Summary
	STASH-4526	Backup Client 1.2.0 cannot restore

1 issue

26 March 2014 - Stash 2.11.5

T	Key	Summary
	STASH-4519	Migration/Setup: "A database error has occurred" masks useful failure details
	STASH-4518	SQL Server: Deleting repositories with many comments fails

2 issues

17 March 2014 - Stash 2.11.4

T	Key	Summary
	STASH-4503	Page doesn't scroll to the focused comment
	STASH-4488	Commit graph cache incorrectly propagates exception
	STASH-4432	Links to comments in emails no longer jump to the comment
	STASH-4410	When cycling through files in the diff view quickly using j/k, the window scrollbar disappears and it's not possible to scroll the diff

	STASH-4407	Diff scrolling 'jumps' near the top of the page.

5 issues

5 March 2014 - Stash 2.11.3

T	Key	Summary
	STASH-4453	Hibernate Session OutOfMemoryError AND ORA-01000
	STASH-4452	500 errors on PR page due to markdown renderer
	STASH-4420	BranchModelService is not exported to OSGI
	STASH-4411	Commit links uses relative paths in emails

4 issues

25 February 2014 - Stash 2.11.2

T	Key	Summary
	STASH-4296	Add support for SQL Server 2014
	STASH-4256	Support project keys with hyphens
	STASH-2981	Link to commits within Markdown
	STASH-2756	File Comments on Pull Requests
	STASH-2511	Comments on Commits
	STASH-2492	Include side-by-side diffs
	STASH-4396	Increase license service cache expiration
	STASH-3164	Add MySQL 5.6 to test matrix
	STASH-4372	Some invalid licenses show 500 errors when saving
	STASH-4362	Security vulnerability in apache commons fileupload
	STASH-4329	Changeset parent selector does not update trigger
	STASH-4299	404 when navigating to Commits (nav item) after merging a pull request with the delete branch option checked
	STASH-4293	NPE in DefaultRescopeProcessor
	STASH-4287	JIRA keys not being indexed for individual repositories
	STASH-4280	Error transitioning JIRA issues containing a Number custom field
	STASH-4245	Branches containing '|' breaks parts of Stash
	STASH-4043	Branch List ahead/behind count incorrect
	STASH-4005	Clones over SSH fail for large repositories
	STASH-3717	Users can be locked out of Stash if external user directory is moved to another IP

19 issues

Stash 2.10 release notes

17 December 2013

Introducing Stash 2.10

Today we're pleased to announce Stash 2.10, which brings a wide range of small improvements. We've also

added considerable polish to the Stash experience – these changes are listed in the 2.10.0 changelog at the bottom of this page.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

Please also check the End of support announcements for Stash.

Small improvements

We've made the following notable improvements:

- There is now a Branch Utils REST API that provides REST resources for managing branches in Stash repositories.
- The SSH clone URL is now shown for admins, when a project or repository access key is available.
- The permission screens for projects and repositories have been tidied.
- The URL now updates correctly when you switch revisions via the file history when viewing the 'Diff to previous' tab.
- Stash has been updated to use AUI 5.3.

Change log

This section will contain information about the Stash 2.10 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 2.10 releases.

22 May 2014 – Stash 2.10.5

T	Key	Summary
	STASH-4696	Memory leak when using a persistent SSH connections
	STASH-4640	page.max.directory.children does not show correct number of files on page load
	STASH-4217	SSH idle timeout is too large resulting in OutOfMemoryError when clients fail to terminate the session

3 issues

1 April 2014 – Stash 2.10.4

T	Key	Summary

No issues found

27 March 2014 – Stash 2.10.3

T	Key	Summary
	STASH-4519	Migration/Setup: "A database error has occurred" masks useful failure details
	STASH-4518	SQL Server: Deleting repositories with many comments fails

2 issues

14 February 2014 – Stash 2.10.2

T	Key	Summary
	STASH-4326	Clicking "Decline" declines and then merges a pull request

1 issue

31 December 2013 – Stash 2.10.1

T	Key	Summary
	STASH-4242	Stash shutdown is stalling
	STASH-4239	Scrolling on Pull Request activity page broken after second page

2 issues

17 December 2013 – Stash 2.10.0

T	Key	Summary
	STASH-3267	Ability to see related pull requests from within JIRA
	STASH-4326	Clicking "Decline" declines and then merges a pull request
	STASH-4239	Scrolling on Pull Request activity page broken after second page
	STASH-4215	Branch listing: Builds header not hidden when filtered
	STASH-4154	LDAP synchronization fails when user rename contains case only changes
	STASH-4131	Error calculating comment drift on startup
	STASH-4121	Default user avatars do not load
	STASH-4116	Pull requests cannot be merged in repositories with extremely short slugs
	STASH-4115	IE8: missing conflict marker in difftree
	STASH-4112	2.9.1 upgrade fails when SSH keys exist with labels greater than 255 characters
	STASH-4111	SSH keys corrupted after upgrade to 2.9.1 when key contains newline characters
	STASH-4107	Paging in access keys doesn't work
	STASH-4106	Access logs do not log which access key used the system
	STASH-4104	Branch create: js error in IE8. _ bind on an object
	STASH-4100	Repository move not recorded in audit log
	STASH-4052	Changing the base url should require system administration privileges
	STASH-4002	Browse Repository page not rendering correctly in IE11
	STASH-3918	Stash adds class "aui" to "table"-tags when inside markdown backtick code block
	STASH-3866	SSH clone URL generation should take the request URL into account
	STASH-3859	<dl> block not rendered by Stash in README

Showing 20 out of 23 issues

Stash 2.9 release notes

19 November 2013

Introducing Stash 2.9

Today we're excited to announce Stash 2.9, which makes managing your branches, pull requests and access to repositories much easier.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

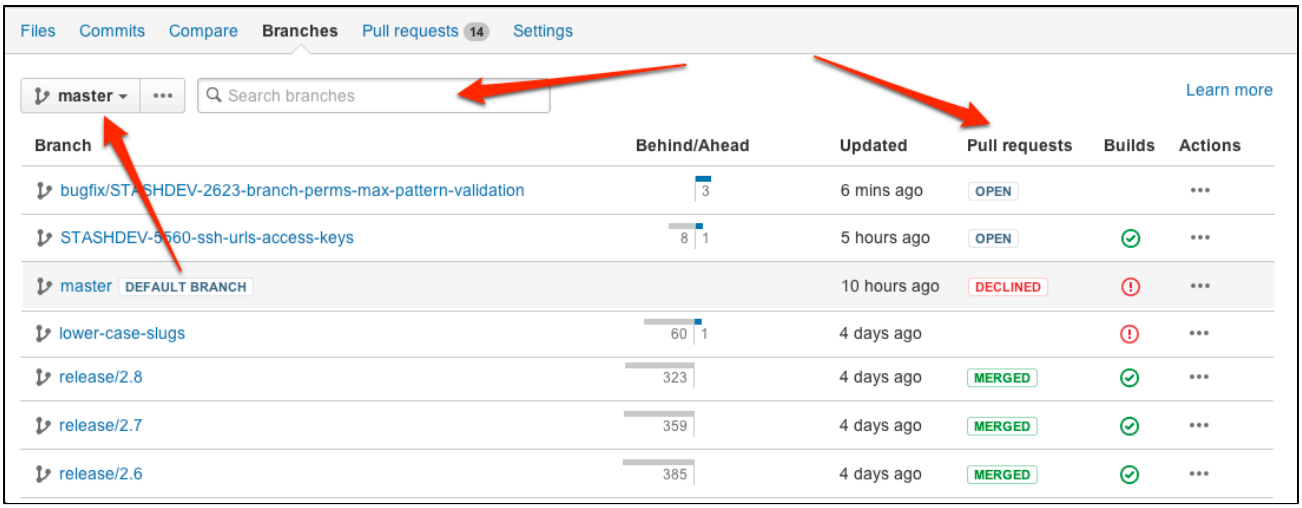
Please also check the End of support announcements for Stash.

The Stash 2.9 changelog is at the bottom of this page.

Branch listing improvements

We've made the branches listing a much more useful overview of the state of your repository:

- The new **Pull requests** status helps you to track the review and merge work that still needs to be done.
- The **Behind/Ahead** column helps you identify work in progress as well as stale branches, and now you can choose the base branch for the comparison.
- The new search makes it easy to find brances when you have a ton of them. Even better, if you're using the Stash branch model, you can filter by branch type simply by searching for the prefix – for example, search for "feature/" to see all your feature branches.
- As before, the **Builds** column allows you to see the build status of branches at a glance.
- The **Actions** menus have altered tasks, such as creating a pull request, and checking out the branch in Atlassian SourceTree.
- There are new navigation shortcuts – see them in **Keyboard shortcuts** under the Stash Help menu.



Read more about the branches listing...

SSH access keys for projects and repositories

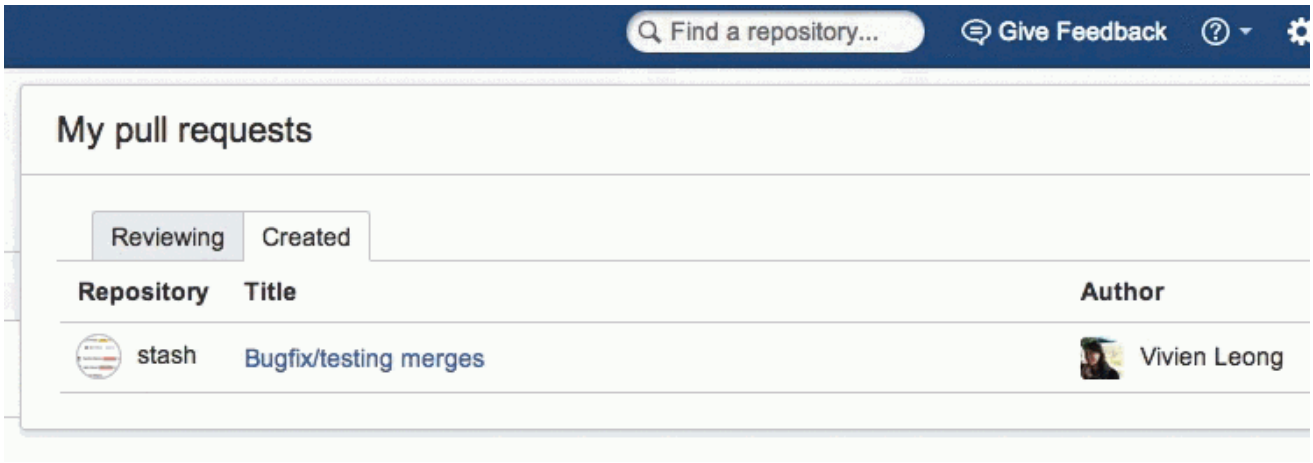
Stash now provides a simple way for other systems to perform read-only Git operations on repositories managed in Stash. This allows systems such as your build server to authenticate with Stash to checkout and test source code, without having to store user credentials on another system, and without requiring a specific user account.

These new access keys complement the personal account keys that have been available since Stash 1.1 – but they work for repositories rather than user accounts.

Read more about access keys ...

Pull request inbox

Our pull request inbox has always provided easy access to the pull requests that need your review. In Stash 2.9, we've improved the inbox so that it also shows your own open pull requests. Toggle between these two sets directly from your inbox, and keep track of all your open pull requests seamlessly.

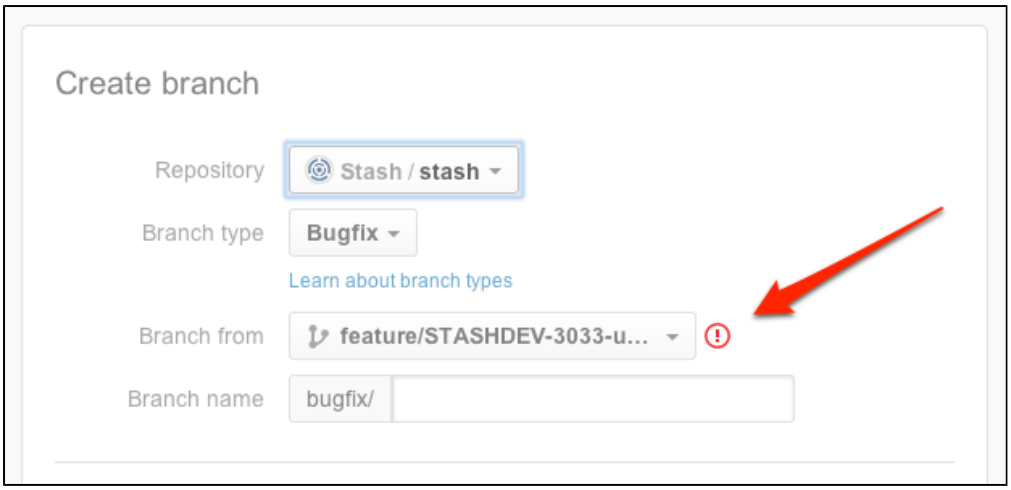


Read more about pull requests in Stash...

Small improvements

Build status during branch creation

When you're creating a new branch, Stash displays the current build status for the source branch – is the branch good enough to branch from? *Read more about build status ...*



Support for PostgreSQL 9.3

See Supported platforms.

Extra keyboard shortcuts

We've added new keyboard shortcuts to help you navigate around the branch listing screen quickly – choose **Keyboard shortcuts** from the Stash Help menu to see these. Read more about the branch listing screen.

Extra merge strategy option

We've added the `squash-ff-only` option. As with `squash`, it collapses all the incoming commits into a single commit directly to the target branch, never creating a merge, but it does so *only* if the source branch is fast-forward. If not, a `MergeException` will be thrown. See Stash config properties.

Support for changing usernames

You can change the username for a user account that is hosted in Stash's internal user directory. If a user is renamed in your LDAP directory, Stash will pick that up in the next synchronisation, or when the user next logs in. All content and attributes associated with the original username will be automatically associated with the new username, including pull requests, comments and permissions. See the Crowd 2.7 release notes.

Change log

This section will contain information about the Stash 2.9 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 2.9 releases.

14 February 2014 – Stash 2.9.5

T	Key	Summary
	STASH-4326	Clicking "Decline" declines and then merges a pull request
	STASH-4239	Scrolling on Pull Request activity page broken after second page

2 issues

28 November 2013 – Stash 2.9.4

T	Key	Summary
	STASH-4154	LDAP synchronization fails when user rename contains case only changes

1 issue

26 November 2013 – Stash 2.9.3

T	Key	Summary
	STASH-4215	Branch listing: Builds header not hidden when filtered
	STASH-4121	Default user avatars do not load
	STASH-4116	Pull requests cannot be merged in repositories with extremely short slugs
	STASH-4115	IE8: missing conflict marker in difftree
	STASH-4100	Repository move not recorded in audit log
	STASH-3918	Stash adds class "aui" to "table"-tags when inside markdown backtick code block
	STASH-3557	Stash favicon looks bad on non white backgrounds

7 issues

20 November 2013 – Stash 2.9.2

T	Key	Summary
	STASH-4112	2.9.1 upgrade fails when SSH keys exist with labels greater than 255 characters
	STASH-4111	SSH keys corrupted after upgrade to 2.9.1 when key contains newline characters

2 issues

19 November 2013 – Stash 2.9.1

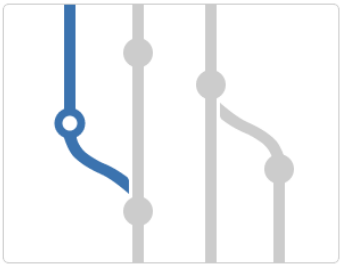
T	Key	Summary
	STASH-3360	List of Owned Pull Requests
	STASH-2821	add support for deployment ssh keys
	STASH-3998	More logging in DefaultCrowdSSoService when fetching cookie config fails
	STASH-4075	Pull Request Settings are visible and editable to non-admin users
	STASH-4025	Stash Backup Client fails if the backup will is greater than 8GB
	STASH-4021	SSH SCM Cache causes " c.a.s.i.s.g.p.ssh.GitSshScmRequest upload-pack did not complete. It will be assumed the command failed" errors
	STASH-4018	Duplicate entries on the project list
	STASH-4011	File Search screen should not show README file
	STASH-4010	Upgrade Trusted Apps to fix signature issues
	STASH-3991	Stash dependency leaking into plugin dependencies
	STASH-3977	JIRA Feature Discovery behaves badly when switching tabs
	STASH-3961	Ref syncing raises RepositoryRefsChangedEvents with bad RefChanges
	STASH-3956	Ref syncing fails when notes are pushed

	STASH-3949	Source view fails with "Resetting to invalid mark" on larger source files
	STASH-3945	TrustedApplicationsAuthListener.authenticationFailure produces NullPointerExceptions
	STASH-3943	ORA-01795: maximum number of expressions in a list is 1000
	STASH-3930	The BackupController.java startBackup method is vulnerable to csrf
	STASH-3915	The MaintenanceController.java lock method is vulnerable to csrf
	STASH-3914	BranchSelectorField doesn't work when appended to an intermediate non-DOM element with jQuery
	STASH-3907	Stash includes a Clover license in the public POM file
	STASH-3871	Stash 2.7 fails to start with MySQL when binary logging is enabled
	STASH-3824	SSH operations hang when using PuTTy or TortoiseGit
	STASH-3535	PagedTable JS component spinner hiding race conditions

23 issues

Stash 2.8 release notes

1 October 2013



Introducing Stash 2.8 – Git branch workflows

Today we're excited to announce Stash 2.8, which makes it easy for your team to use a branching workflow for your Git development process.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

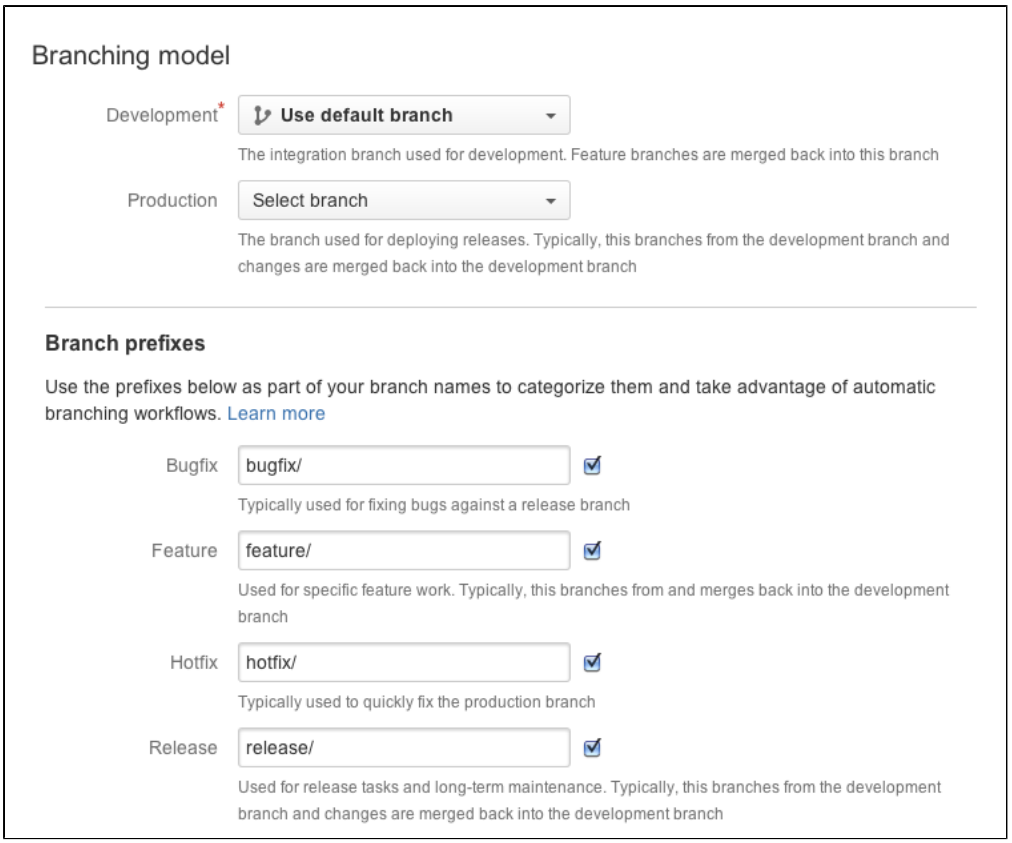
Please also check the End of support announcements for Stash.

The Stash 2.8 changelog is at the bottom of this page.

Stash branching model

Stash 2.8 introduces a 'branching model' that provides key advantages for streamlining your Git development workflow:

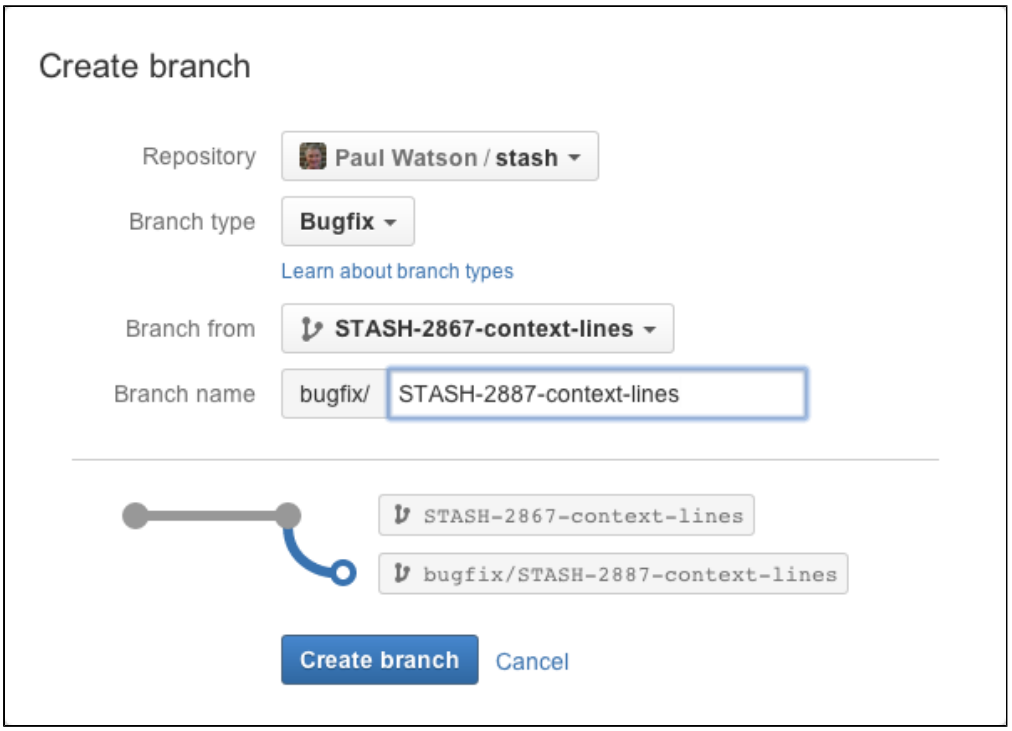
- Developers can make consistent naming decisions when creating branches in Stash, guided by the naming conventions specified in the branching model.
- Reduce the need for manual maintenance of release branches, because consistent branch naming allows Stash to automatically cascade merges to those branch types.



Read more about the new branching model in Stash.

Branch creation from within Stash

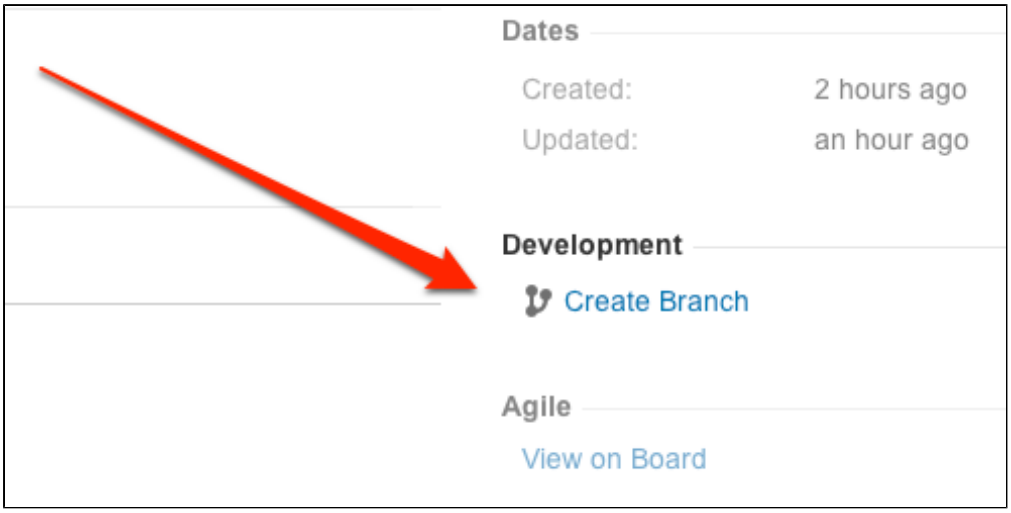
You can now create new branches in your Git repositories directly from the Stash web UI. When a branching model is configured, Stash guides you into making consistent choices for branch names:



Read more about creating branches in Stash.

Branch creation from within JIRA and JIRA Agile

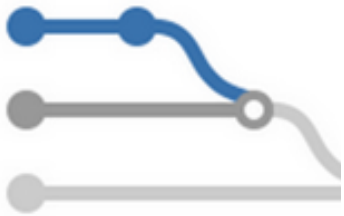
As part of our push to provide tighter integration between Stash and JIRA, you can now start creating a branch from within a JIRA issue, whether you are in JIRA or JIRA Agile. This gives you a faster workflow from picking an issue to starting coding:



Stash will suggest the branch type and branch name, based on the JIRA issue type and summary – you can change these, of course.

Your Admin needs to have set up linking with JIRA before you'll see this. Read more about creating branches from JIRA.

Automated merges



When a branching model is configured in Stash that defines a 'release' branch for a repository, Stash can automatically merge changes to newer release branches. Cascading merges can reduce the manual maintenance for branches of that type.

Automated merging must be enabled for each repository, and is subject to a number of conditions. Stash applies an ordering algorithm, based on patterns in the branch names, to determine the 'newer' branches.

Read more about automatic branch merging in Stash.

Branch listing pages

To complement branch workflows, Stash now provides an overview of all branches in a repository. The **Behind/Ahead** information shows by how many commits a branch has diverged from the default branch (for example, `master`) for the repository, and can help you to identify work in progress as well as stale branches. The **Actions** menus provide branch tasks such as viewing the branches source files, creating a pull request, creating a new branch, and checking out the branch in Atlassian SourceTree.

.

Read more about the branches listing for a repository in Stash.

Move Git repositories between Stash projects

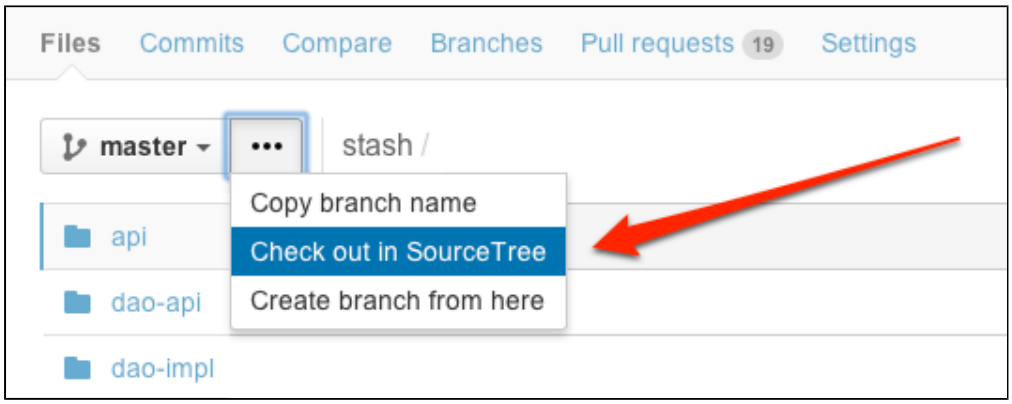
You can now move a repository from one Stash project to another, if you have project administrator permissions for both projects.

To move a repository, go to **Settings** > **Repository details** for the repository you wish to move, and click **Move repository**:

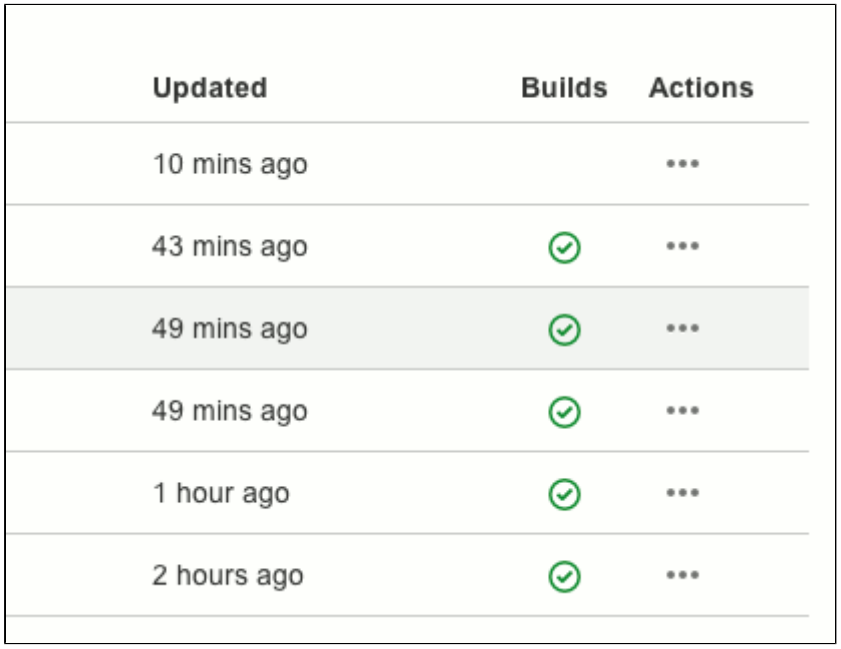
Improved integration with Atlassian SourceTree

You've been able to clone a repository from Stash using Atlassian SourceTree for a while. Now there are more ways that you can use Stash and SourceTree together, to help you use Git faster and more easily. Choose **Check out in SourceTree** from the Actions menu when you are:

- Viewing files or commits in a repository:



- Viewing branches in a repository:



Small improvements

Stash backup client is now production-ready

The Stash backup client introduced with Stash 2.7.0 is now supported for use in production environments. See Data recovery and backups.

See the branches that a commit is visible on

When browsing a commit you can now see all the branches that the current commit is visible on. This makes it easier to determine if a particular change has been merged to the master branch, for example, or not:

SSH support in the SCM Cache Plugin

The SCM Cache Plugin now supports SSH. This allows your CI build server to securely access the Stash server over SSH. See Scaling Stash for Continuous Integration performance for configuration information.

Auto-expand directories with only one sub-directory

In the files listing for a repository, Stash will now automatically expand a directory that contains only one sub-directory. This can save time where there is a series of nested single directories – Stash will continue expanding directories until it finds two or more sub-directories.

Change log

This section will contain information about the Stash 2.8 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are the highlights of all those that have been resolved for the Stash 2.8 releases.

10 March 2014 – Stash 2.8.5

T	Key	Summary
	STASH-4452	500 errors on PR page due to markdown renderer
	STASH-4075	Pull Request Settings are visible and editable to non-admin users

2 issues

11 November 2013 – Stash 2.8.4

T	Key	Summary
	STASH-4122	Privilege escalation
	STASH-4062	DB Migration fails with NumberFormatException
	STASH-4010	Upgrade Trusted Apps to fix signature issues

3 issues

5 November 2013 – Stash 2.8.3

T	Key	Summary

	STASH-4026	Switching branches in file view causes internal error with branch names containing a "/"
	STASH-4018	Duplicate entries on the project list
	STASH-4000	Occasional 500 errors from stash - NPE in XSRF token validation
	STASH-3997	Branch permission patterns are case sensitive
	STASH-3991	Stash dependency leaking into plugin dependencies
	STASH-3987	Backups may be generated with out-of-date database data
	STASH-3986	Stash REST API does not consistently provide a nextPageStart on paginated resources
	STASH-3977	JIRA Feature Discovery behaves badly when switching tabs
	STASH-3914	BranchSelectorField doesn't work when appended to an intermediate non-DOM element with jQuery
	STASH-3356	Stash fails to connect to MySQL database if the user password contains special symbols
	STASH-3288	UI broken after inactive user added to pull request
	STASH-3253	Inconsistent behaviour for nextPageStart attribute on commits REST resource when specifying a since parameter

12 issues

15 October 2013 – Stash 2.8.2

T	Key	Summary
	STASH-3973	Hook callback sockets are not closed correctly
	STASH-3961	Ref syncing raises RepositoryRefsChangedEvents with bad RefChanges
	STASH-3956	Ref syncing fails when notes are pushed
	STASH-3952	Ahead and behind information is not visible on the branch page anonymously
	STASH-3949	Source view fails with "Resetting to invalid mark" on larger source files
	STASH-3947	Searching nested group memberships fails for users with a large number of groups
	STASH-3945	TrustedApplicationsAuthListener.authenticationFailure produces NullPointerExceptions
	STASH-3943	ORA-01795: maximum number of expressions in a list is 1000
	STASH-3495	Unsafe redirect

9 issues

2 October 2013 – Stash 2.8.1

T	Key	Summary
	STASH-3922	Cannot set a reviewer when editing a pull request

1 issue

1 October 2013 – Stash 2.8.0

T	Key	Summary
	STASH-3561	Create branch on Stash like on github
	STASH-3347	Delete branch from UI
	STASH-3122	API to hook into pull request merge process.
	STASH-2796	Possibility to move a git repository from one project to another

	STASH-2691	Display branches that particular commit is visible on
	STASH-3888	Updating a trusted application fails with unique constraint violation on uk_trusted_app_restrict
	STASH-3881	stash fails to display reviews/commits where commit message includes <U+2028>
	STASH-3880	Clickjacking could be used to add SSH keys
	STASH-3875	The default project permission is not getting cleared from the cache
	STASH-3837	Pull request merges fail with "Could not get current working directory"
	STASH-3819	ExternalProcessImpl.areOutputPumpsRunning is not thread-safe
	STASH-3793	git rev-list fails on startup
	STASH-3788	Pull request events are not audited
	STASH-3777	Stash cannot connect to other Atlassian applications on non-standard HTTPS port
	STASH-3776	Cross-site request forgery token checking implementation is vulnerable to a timing attack
	STASH-3748	since and until parameters combined within REST API makes jira-key attribute go missing
	STASH-3204	DataIntegrityViolationException when configuring Trusted Application

17 issues

Stash 2.7 release notes

20 August 2013

Introducing Stash 2.7

Today we're pleased to announce Stash 2.7, which introduces a range of JIRA integration improvements, to help optimise your development workflows. We've also included a brand new Stash backup and restore solution.

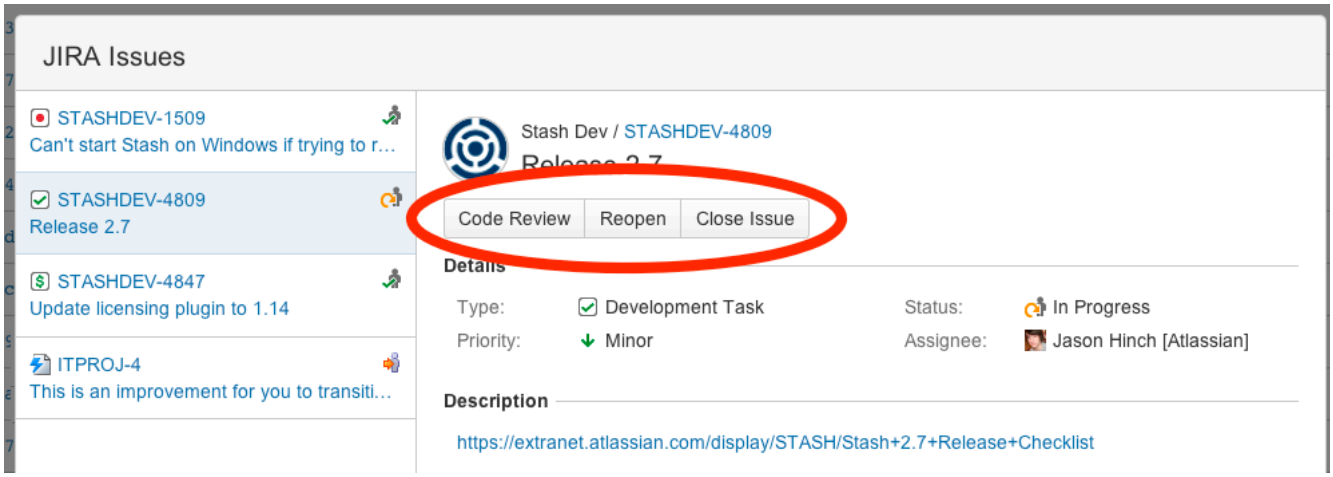
If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

Please also check the End of support announcements for Stash.

The Stash 2.7 changelog is at the bottom of this page.

JIRA issue transitions

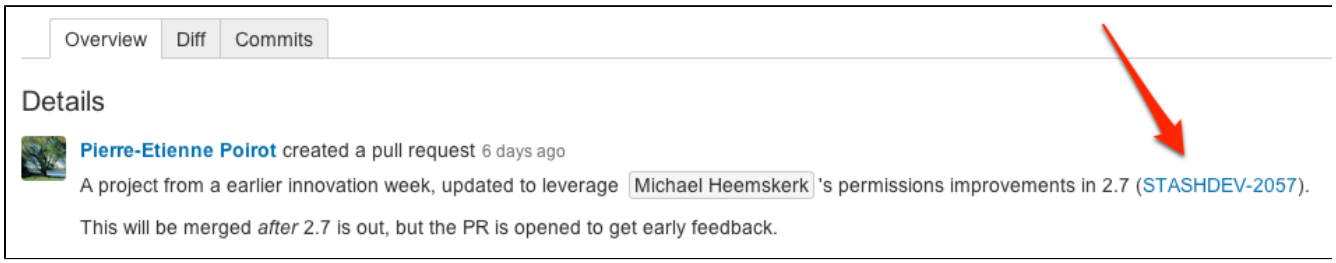
Now you can easily transition a JIRA issue from within Stash. For example, when creating a pull request you may want to transition the issue into review. Click on a linked JIRA issue anywhere in Stash to see a dialog with the available workflow steps:



Your Admin needs to have set up linking with JIRA before you'll see this. Read more about JIRA integration in Stash.

Autolink JIRA issues in markdown

When you mention JIRA a issue key in Stash, for example in a pull request description or comment, the key gets automatically linked:



You can click on the linked key to see details for the issue. Note that issue keys in a URL will link directly to the JIRA instance, as you'd expect. Read more about JIRA integration in Stash.

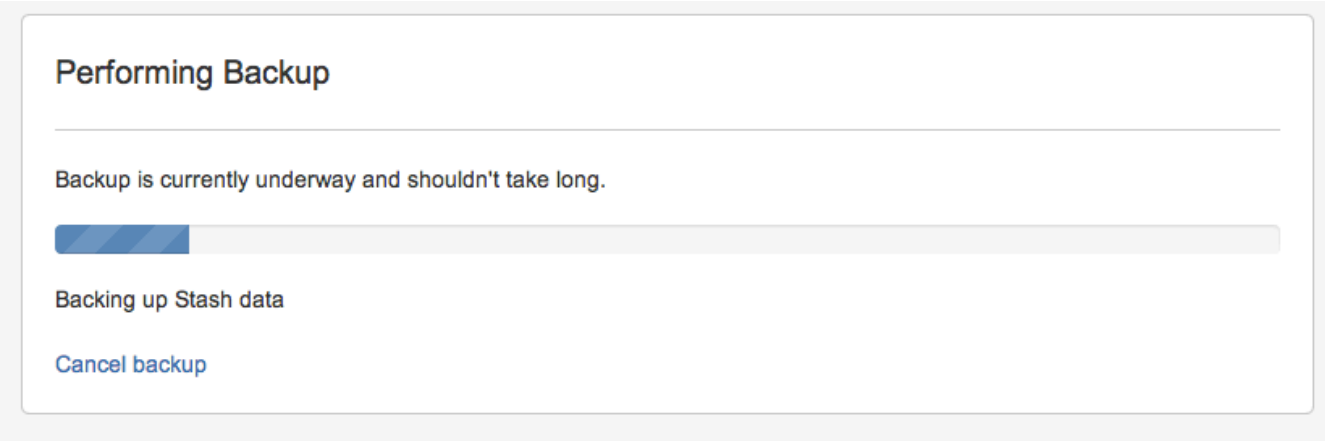
Support for multiple JIRA instances

Stash can now link to more than one JIRA server at a time, so different teams can work with their own projects in different JIRA instances, or a single team can link to issues across multiple JIRA servers.

Read more about JIRA integration in Stash.

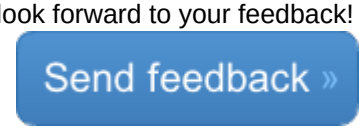
Backup and restore beta

With the Stash 2.7 release we have made available a *beta* release of a backup and restore client for Stash. The Stash backup client locks access to Stash, checks that all Git and database operations have completed, and then performs a concurrent backup of the Stash database, repositories and data. You can download the Stash backup and restore client from the Atlassian Marketplace.



Although we are confident about the performance of this tool, we are interested in hearing how well it meets your requirements in actual use – this is why we are releasing the client as a beta. We look forward to your feedback!

Read more about backing up and restoring your Stash data.



Small improvements

Improved LDAP synchronisation performance

We have greatly improved the performance for LDAP synchronisation. For example, on a test LDAP server with

10,000 users and 10,000 groups, the time for a full synchronisation improved from 64 minutes to 4 minutes.

The Repository System Information plugin is now deprecated

The functionality of the repository system information plugin has now been moved into core Stash. The plugin will still work for Stash 2.x versions but is redundant as of Stash 2.7.

Syntax highlighters are now configurable

You can now update the extensions associated with different syntax highlighters using the `stash-config.properties` file in the Stash home directory. For more details see Configuring syntax highlighting for file extensions.

MySQL default isolation level

Stash 2.7.x uses READ_COMMITTED instead of the MySQL default isolation level (REPEATABLE_READ). This can result in exceptions when installing/upgrading to 2.7.x, if binary logging is enabled in your MySQL server. More details and a fix can be found in this KB article.

Change log

This section will contain information about the Stash 2.7 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are just the highlights of all those that have been resolved for the Stash 2.7 release.

11 November 2013 – Stash 2.7.6

T	Key	Summary
	STASH-4122	Privilege escalation
	STASH-4062	DB Migration fails with NumberFormatException
	STASH-4010	Upgrade Trusted Apps to fix signature issues

3 issues

5 November 2013 – Stash 2.7.5

T	Key	Summary
	STASH-3987	Backups may be generated with out-of-date database data

1 issue

15 October 2013 – Stash 2.7.4

T	Key	Summary
	STASH-3973	Hook callback sockets are not closed correctly

1 issue

15 October 2013 – Stash 2.7.3 (Internal release only)

T	Key	Summary
	STASH-3961	Ref syncing raises RepositoryRefsChangedEvents with bad RefChanges
	STASH-3956	Ref syncing fails when notes are pushed
	STASH-3952	Ahead and behind information is not visible on the branch page anonymously
	STASH-3949	Source view fails with "Resetting to invalid mark" on larger source files
	STASH-3947	Searching nested group memberships fails for users with a large number of groups
	STASH-3943	ORA-01795: maximum number of expressions in a list is 1000
	STASH-3888	Updating a trusted application fails with unique constraint violation on uk_trusted_app_restrict
	STASH-3881	stash fails to display reviews/commits where commit message includes <U+2028>
	STASH-3880	Clickjacking could be used to add SSH keys
	STASH-3875	The default project permission is not getting cleared from the cache
	STASH-3777	Stash cannot connect to other Atlassian applications on non-standard HTTPS port
	STASH-3495	Unsafe redirect
	STASH-3204	DataIntegrityViolationException when configuring Trusted Application

13 issues

19 September 2013 - Stash 2.7.2

T	Key	Summary
	STASH-3843	LazyReference$InitializationException: java.lang.IllegalArgumentException: duplicate key: {username}
	STASH-3838	Public repositories not showing project after first 25

2 issues

29 August 2013 - Stash 2.7.1

T	Key	Summary
	STASH-3807	Show the first few lines of an exception on the console
	STASH-3805	Backup client fails if backup directory starts with same string as Stash home
	STASH-3804	Admin > Users returns empty user list for Oracle database
	STASH-3798	nav-links and ref-syncing are not in the list of bundled plugins
	STASH-3793	git rev-list fails on startup
	STASH-3788	Pull request events are not audited
	STASH-3786	StringIndexOutOfBoundsException: String index out of range: -1

7 issues

20 August 2013 - Stash 2.7.0

T	Key	Summary
	STASH-3733	Bundle Repository System Info Plugin with Stash

	STASH-3594	Syntax Highlighting For Erlang Code
	STASH-2471	Support integration with multiple JIRA servers
	STASH-3634	De-Activated Users In Crowd Still Count Against the User License
	STASH-3720	Document .mailmap feature
	STASH-3698	Create an "End of Support Announcements for Stash" page
	STASH-3744	Stash incorrectly counts licensed users when more than 1000 users are licensed through a single group
	STASH-3671	Potential for XSS in third party code using internal <page-data-provider> module type
	STASH-3651	JIRA issue link not obeying Display URL value in Application Links (JIRA Issues pop-up only)
	STASH-3588	Synchronising a directory fails on MySQL
	STASH-3577	Mouseover tooltip on "Approve" button always tells user to "Approve this pull request"
	STASH-3492	Unauthenticated access to Setup
	STASH-3471	HistoryService lacks documentation
	STASH-3415	Hide upload public SSH key option, when SSH access is disabled
	STASH-3301	About page doesn't respect that I'm logged in
	STASH-2868	When using JIRA authentication, inactive users are shown in Stash

16 issues

Stash 2.6 release notes

22 July 2013

Fork synchronization with Stash 2.6

Today we're pleased to announce Stash 2.6, which introduces automated fork synchronization and audit logging. We've also added a repository search and polished some existing features.

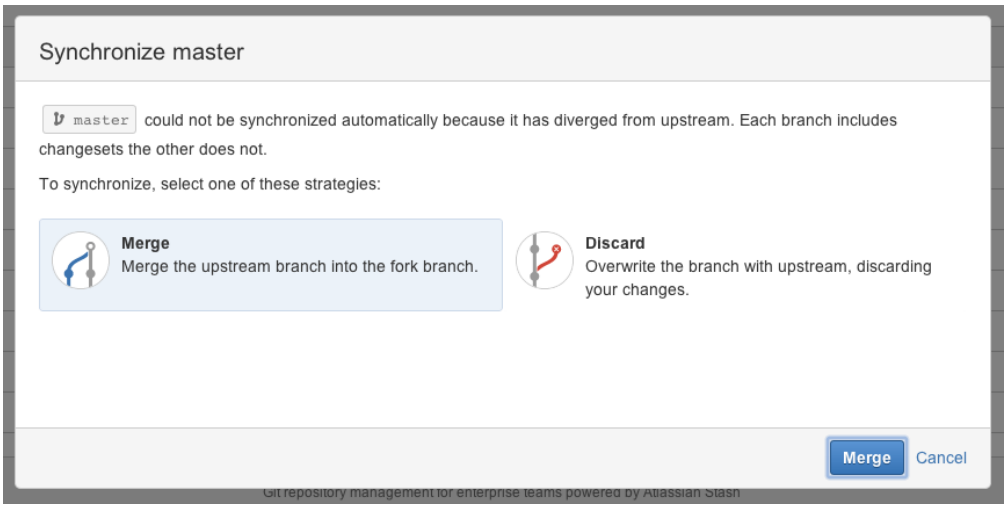
If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The Stash 2.6 changelog is at the bottom of this page.

Fork synchronization

Fork syncing provides a way for your fork in Stash to be kept up-to-date with changes in the upstream repository, thus minimising the effort needed to do this manually. Stash can do this automatically, as long as you haven't modified the branches or tags in the fork.

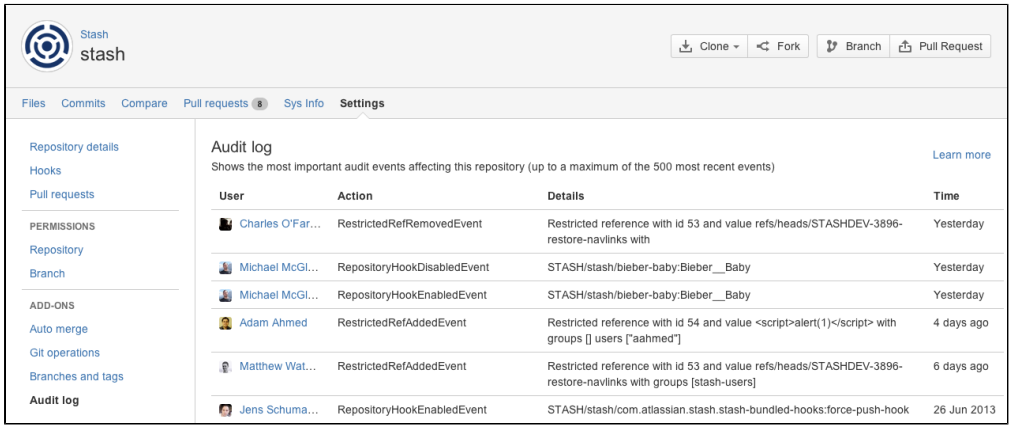
If you have modified branches or tags in the fork, Stash will offer merge options:



Read more about using forks in Stash.

Audit logging

Stash now provides full audit logging. The most important audit events for each repository and project are displayed in Stash on the **Settings** tab (these are only visible to Stash admins and system admins):

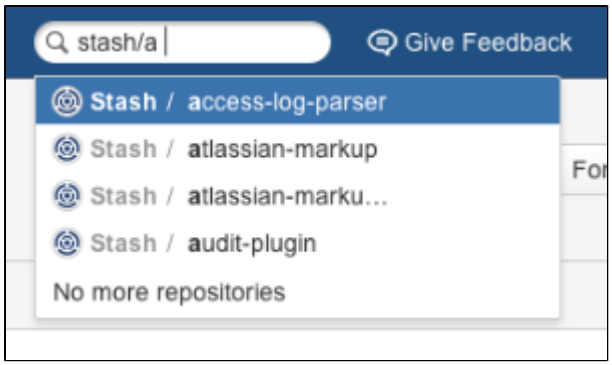


Stash also creates full audit log files that can be found in the `<Stash home directory>/log/audit` directory. The level of logging can be configured via system properties. Note that we recommend an automated backup of log files.

Read more about audit logging in Stash. Plugin developers can specify that their plugin events should be added to the 'Audit log' view, and the full audit log file.

Repository Quicksearch

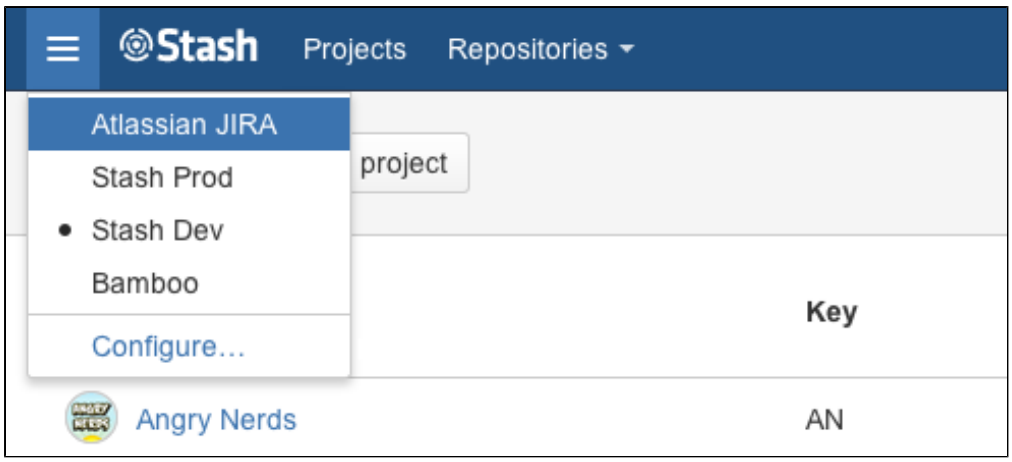
Quicksearch makes it easy to jump to a particular repository, without needing to know the name of the project it's in. Start typing to see a list of suggested repositories. If you type a '/', text before that is matched against project names, and text after is matched against repository names. Note that wildcards and antglobs are *not* supported.



Small improvements

Application navigator

The new application navigator, on the left of the header, allows you to switch to your other applications, such as Atlassian JIRA and Bamboo – or any other web application – all from the Stash header:



Stash administrators can configure which apps appear in the navigator – just click **Configure** in the application navigator, or go to the Stash admin area and click **Application navigator.** See Configuring the application navigator.

List of public repositories

Stash now displays a list of repositories for which anonymous access has been enabled – anonymous and logged-in users can choose **Repositories** > **View all public repositories**. See Allowing public access to code.

Deprecation of Java 6

In version 3.0, Stash will no longer support Java 6.0, and will only support Java 7.0 and above. Stash 3.0 is expected to be released later in 2013. See Supported platforms.

Deprecation of Internet Explorer 8

In version 3.0, Stash will no longer support Internet Explorer 8, and will only support Internet Explorer 9 and above. Stash 3.0 is expected to be released later in 2013. See Supported platforms.

Change log

This section will contain information about the Stash 2.6 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are just the highlights of all those that have been resolved for the Stash 2.6.x releases.

11 November 2013 - Stash 2.6.5 (1 issues)

T	Key	Summary
	STASH-4122	Privilege escalation

1 issue

12 August 2013 - Stash 2.6.4 (7 issues)

T	Key	Summary
	STASH-3952	Ahead and behind information is not visible on the branch page anonymously
	STASH-3888	Updating a trusted application fails with unique constraint violation on uk_trusted_app_restrict
	STASH-3742	Pull request merge fails if there are rename-add or directory-file conflicts
	STASH-3627	Horizontal Scrollbar Hides the Last Line
	STASH-3602	Blame line height is wrong on increased font size in browser
	STASH-3592	Adding buttons into stash.pull-request.toolbar.actions causes wrapping
	STASH-3204	DataIntegrityViolationException when configuring Trusted Application

7 issues

5 August 2013 - Stash 2.6.3 (2 issues)

T	Key	Summary
	STASH-3696	Syntax highlight .pm extensions as Perl
	STASH-3694	LicenseServiceImpl#onGroupMembershipCreatedEvent performs potentially expensive operations causing slow LDAP synchronization

2 issues

26 July 2013 - Stash 2.6.2 (2 issues)

T	Key	Summary
	STASH-3693	Outer sessions are not flushed during crowd synchronization slowing them down significantly for large directories
	STASH-3690	Fork sync issue, creates a merge commit with message that starts with "null"

2 issues

24 July 2013 - Stash 2.6.1 (5 issues)

T	Key	Summary
	STASH-3689	Stash search renders filenames with "null" as "$1"
	STASH-3684	UserCreatedEvent should be transaction aware
	STASH-3681	DefaultCaptchaService#onUserCreated can be slow when synchronising large LDAP directories
	STASH-3676	CallbackLsTreeOutputHandler does not call onEndPage when the git process is interrupted

	STASH-3482	Java mail doesn't work when connecting Microsoft Exchange 2010

5 issues

22 July 2013 - Stash 2.6.0 (15 issues)

T	Key	Summary
	STASH-3652	Provide a way to see all the repositories that are publicly accessible
	STASH-3550	Application Navigator in Stash
	STASH-3425	Sync branches/forks at the server automatically
	STASH-3390	Application Navigator
	STASH-3098	Finding a certain repository
	STASH-3075	Audit logging
	STASH-2983	Ability to Search for a Repository by Name
	STASH-3659	Incorrect rendering of BranchSelector in RepositoryHook config-form
	STASH-3615	PageUtils.newRequest(0, Integer.MAX_VALUE) returns 1 item, but PageUtils.newRequest(0, 2) returns 2
	STASH-3593	Can't use PullRequestService.find method with SecurityService
	STASH-3514	Can't login with Crowd SSO user anymore if Crowd is started a bit later than Stash
	STASH-3506	Stash Footer
	STASH-3123	Support Tool incorrectly signals successful Support Request creation
	STASH-3057	Do a version check of Stash-Home directory before intializing server
	STASH-2465	Stash creates sessions for unauthenticated users

15 issues

Stash 2.5 release notes

12 June 2013

Go public with Stash 2.5

Today we're pleased to announce Stash 2.5, which fosters external participation and let's you enjoy even more streamlined code collaboration with public access to projects and repositories.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The Stash 2.5 changelog is at the bottom of this page.

Public access to projects and repositories

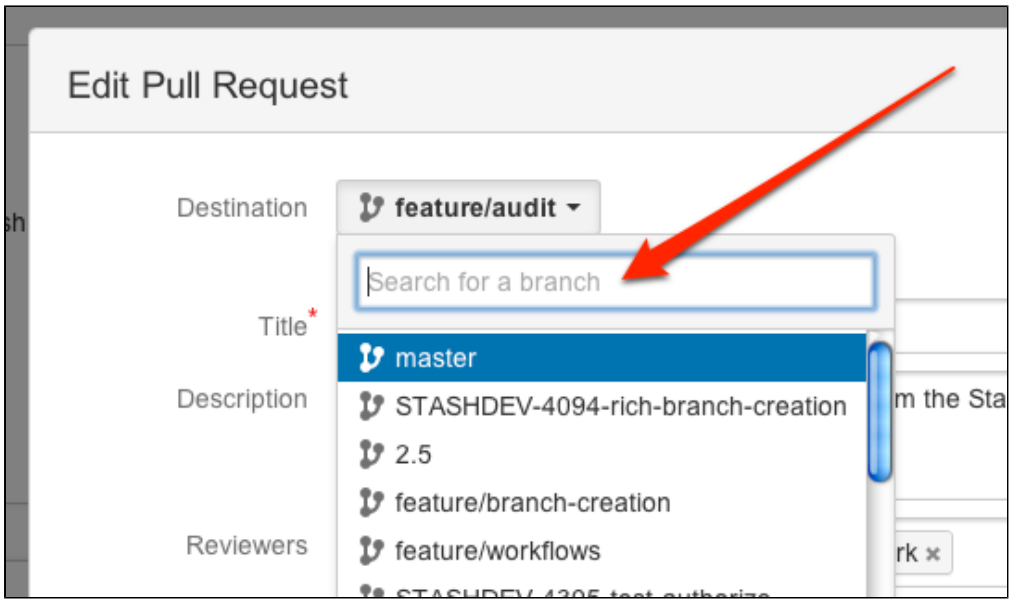
Public access provides anonymous (unauthenticated) users with *browse* access to specific repositories or entire projects in Stash. Anonymous users can see the source code in a repository, and clone the repository. Logged-in users get *read* access to a repository that has public access, so they can fork it and create pull requests. Public access doesn't allow your source code to be changed in any way.

You can configure public access to repositories and projects separately. You can also disable anonymous access by setting a global system property. Please refer to the Stash API changelog for information about how plugin design is affected by public access.

Read more about public access in Stash.

Edit a pull request's destination branch

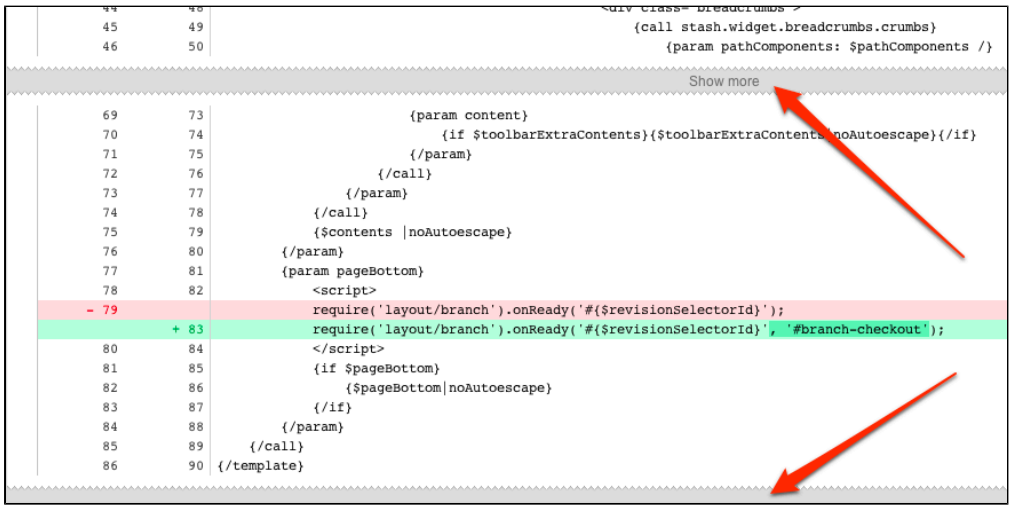
You can now change the destination branch for an active pull request – if you set the wrong destination when creating the PR, you no longer have to delete it and create a new one.



Read more about pull requests in Stash.

Get more context in diffs

We've added a way for you to see more lines of code in the diffs for pull requests and commits. We think this will really help with using pull requests for code reviews. Just click on the grey bars to see more context, up to the whole file if you want. There may also be grey bars at the top or bottom of the diff, if more context is available there.



Note that the expanded lines are displayed as greyed-out, to show that you can't comment on them.

Java OpenJDK is now supported

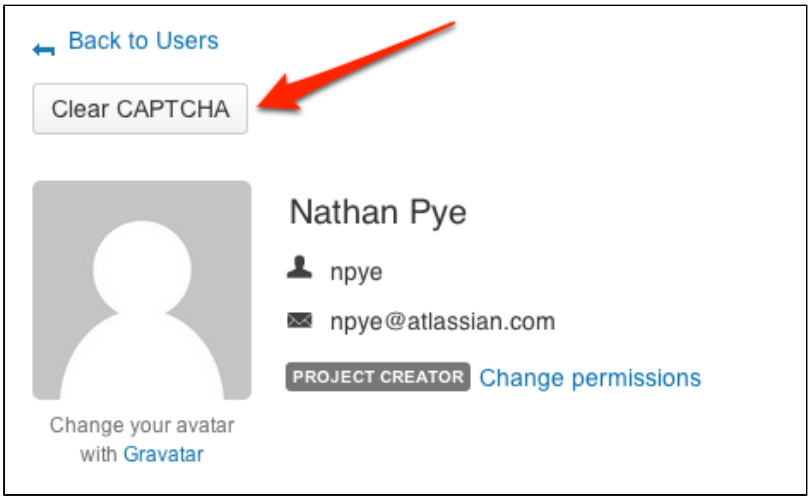
Java OpenJDK is widely used by Linux distributions. We've made your life easier by adding OpenJDK to the list of Java JDKs supported by Stash.

Just a reminder, Java 6 has been deprecated and will not be supported by Stash 3.0 (and later versions), which will be released later in 2013.

See Supported platforms for more information.

Small improvements

- JIRA issues that only appear in forks are now linked to the JIRA server.
- Stash now respects `.mailmap` files in Git repositories. See Git resources for details.
- The SCM Cache plugin is now bundled - see Scaling Stash for Continuous Integration performance.
- We've added a bit more control for CAPTCHA:
  - There is now a system property switch to turn off (and hide) the **Maximum login attempts** control (look under 'Authentication' in the admin area).
  - Administrators are now able to clear a user's CAPTCHA status, from the user's profile:



Change log

This section will contain information about the Stash 2.5 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are just the highlights of all those that have been resolved for the Stash 2.5 release.

11 November 2013 - Stash 2.5.4 (3 issues)

T	Key	Summary
	STASH-4122	Privilege escalation
	STASH-4010	Upgrade Trusted Apps to fix signature issues
	STASH-3952	Ahead and behind information is not visible on the branch page anonymously

3 issues

22 July 2013 - Stash 2.5.3 (1 issues)

T	Key	Summary
	STASH-3681	DefaultCaptchaService#onUserCreated can be slow when synchronising large LDAP directories

1 issue

03 July 2013 - Stash 2.5.2 (6 issues)

T	Key	Summary
	STASH-3622	Elevation of global permission from Administrator to System administrator
	STASH-3604	changing an internal user's password as an "Administrator" fails
	STASH-3599	PermissionService#hasAnyUserPermission incorrectly requires a context user
	STASH-3593	Can't use PullRequestService.find method with SecurityService
	STASH-3590	New footer cuts off "t" explanation tooltip on the Pull Request list
	STASH-3197	README markdown doesn't display if "too large"

6 issues

25 June 2013 - Stash 2.5.1 (5 issues)

T	Key	Summary
	STASH-3579	IE8: Adding reviewers doesn't work on Stash version 2.5.0
	STASH-3578	BAD_SIGNATURE or signature_invalid while creating Stash to JIRA application link
	STASH-3565	Typo in documentation when describing query string parameters to find web fragments
	STASH-3560	Commits added to pull request are not displayed in the correct order
	STASH-3510	<resource name="view" > syntax in client-web-panel does not add web-resource dependency.

5 issues

11 June 2013 - Stash 2.5.0 (16 issues)

T	Key	Summary
	STASH-3408	Allow @Context injection of ContainerRequest in all stash plugins
	STASH-3084	Support .mailmap
	STASH-2934	Change target branch for opened Pull Request
	STASH-2867	As a Stash User, I want to be able to specify the amount of 'context' provided for a diff
	STASH-3476	Documentation: little documentation for overriding stash-default.properties
	STASH-2653	Support OpenJDK7
	STASH-2565	Support anonymous access in Stash
	STASH-3668	Able to create a repository from Source Tree on a Stash project on which i do not have 'admin' access
	STASH-3521	D syntax highlighting is set to use the wrong extension.
	STASH-3513	Stash SSO NPE if Crowd fresh installation does not have SSO domain
	STASH-3439	Unable to find a JSON surrogate for an object of type com.atlassian.stash.internal.web.fragments.WebSectionData
	STASH-3403	JIRA issues not indexed in forks
	STASH-3400	MergeConflictHandler must use `git add -f` to ignore .gitignore
	STASH-3388	Issues with "My Pull Requests" when there are more than nine pull request entries
	STASH-3332	Stash built from the source distribution doesn't work
	STASH-3129	README not displayed if both README & README.md exist

16 issues

Stash 2.4 release notes

6 May 2013

Introducing Stash 2.4 – Forking in the Enterprise

Today we're pleased to announce Stash 2.4, which introduces several key features to help you manage and collaborate on your Git repositories behind the firewall – forking, a workflow popularized in open source development, along with personal repositories and per-repository permissions.

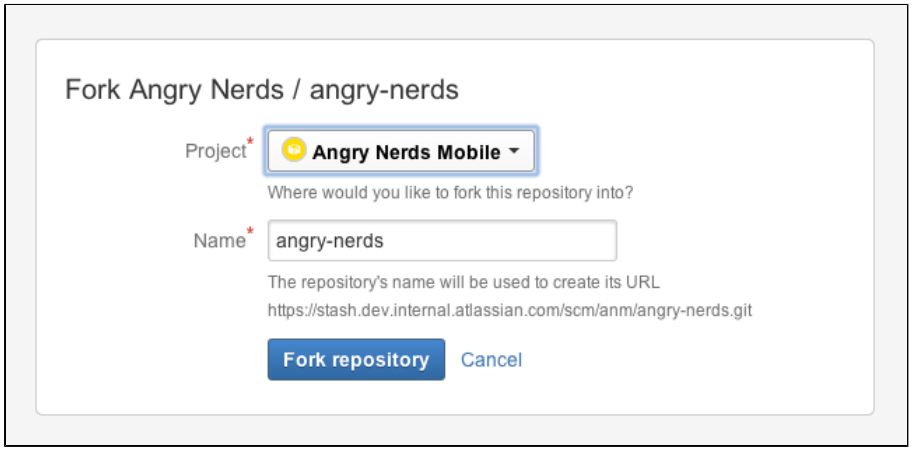
If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The Stash 2.4 changelog is at the bottom of this page.

Forks

Forks in Stash provide developers with a way to contribute code back to a repository for which they do not have write access. This extra control and flexibility of your development process lets you choose workflow scenarios that fit best. You can allow contractors to contribute to a project, or give developers the freedom to spike a project or prototype a feature, without risk to your core repositories.

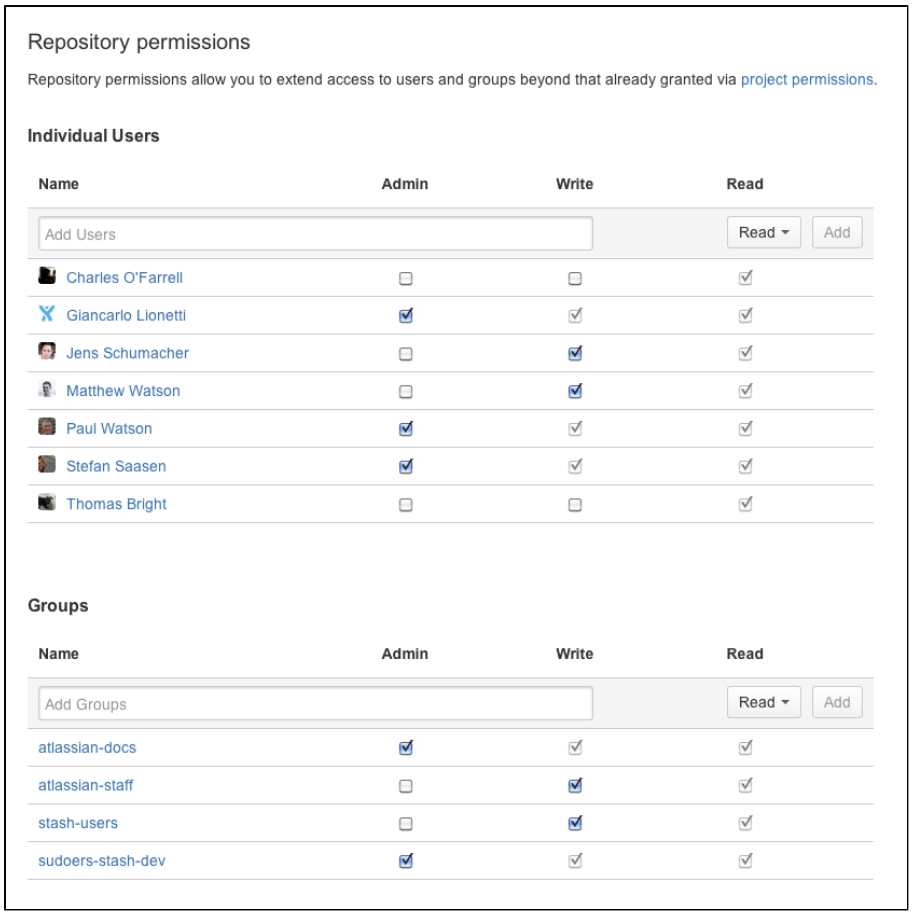
Stash allows developers to fork a repository into any other project in Stash for which they have admin access. They can also create personal forks and then give others access by applying repository permissions.



Read more about using forks in Stash.

Repository permissions

We've added repository permissions to Stash, which along with the already existing project and branch-level permissions, give you flexible yet fine-grained control over access to your repositories.

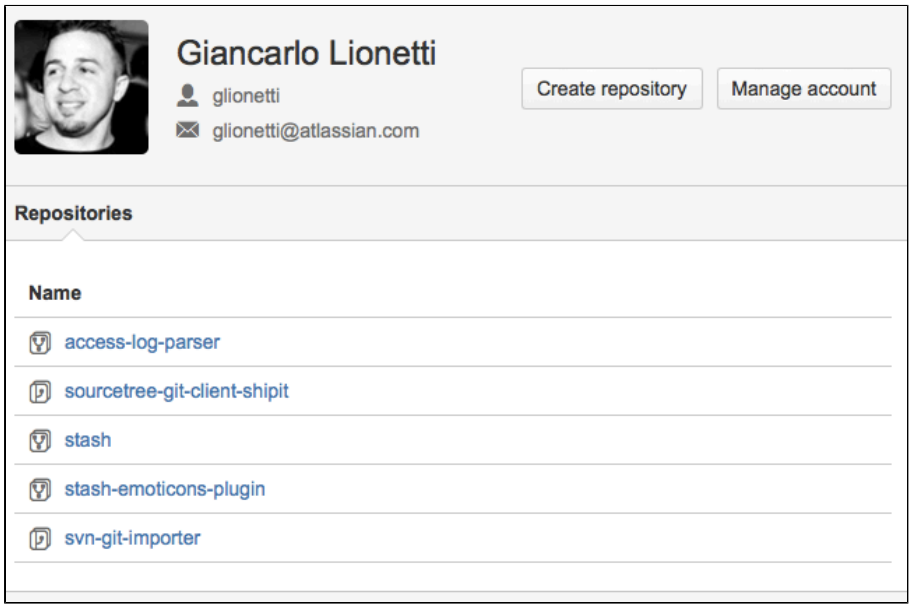


Read more about repository permissions in Stash.

Personal repositories

You can now create personal repositories, unrelated to other projects, that you can use for such purposes as storing private snippets of work, kick-starting your own project, or contributing a bug-fix for a project that you are not a member of. By default, personal repositories are not visible to other Stash users. However, you can use repository permissions to open up access to other individuals and groups.

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



Read more about personal repositories in Stash.

Deprecation of Java 6

This is to give advance notice that Stash 3.0 will not support Java 6.0, and will only support Java 7.0 and above. Stash 3.0 is expected to be released later in 2013.

API changes

Please see the API changelog for any changes made for Stash 2.4.

Small improvements

Managing your Stash account

All the details of your Stash user account are now collected on your Stash account page. Use your account page to manage aspects of your Stash account, including changing your Stash password or avatar, and adding SSH keys.

To go to your account page simply choose **Manage account** from your user menu in the header:

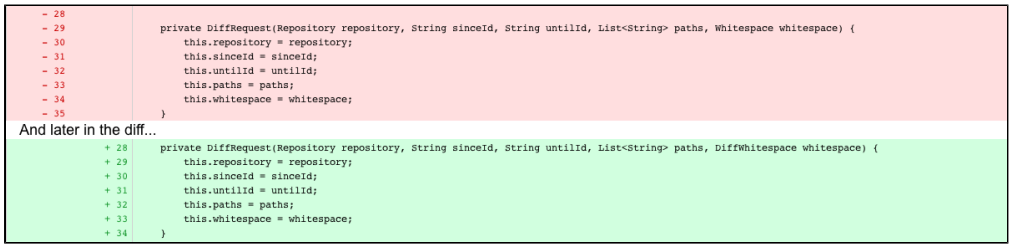
.

Secret whitespace query parameter

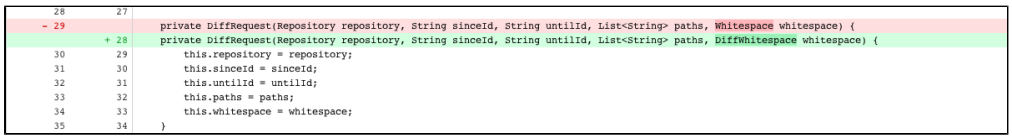
Sometimes a diff is hard to read simply because multiple lines have only had the whitespace changed. To see just the important changes, use the `w=1` (`--ignore-all-space`) query parameter, which is the same as the `git diff` option. For example, the URL could look like:

```
https://stash-your_organisation.com/projects/STASH/repos/your_repo/commits/c573ea24e70c9d21ee?w=1#your_repo_path/DiffRequest.java
```

Before:



After:



Change log

This section will contain information about the Stash 2.4 minor releases as they become available. These releases will be free to all customers with active Stash software maintenance.

If you are upgrading from an earlier version of Stash, please read the Stash upgrade guide.

The issues listed below are just the highlights of all those that have been resolved for the Stash 2.4 release.

21 May 2013 - Stash 2.4.2 (11 issues)

T	Key	Summary
	STASH-3475	Upgrade bundled Tomcat due to security vulnerabilities

	STASH-3463	Invalid "Authorization" headers for basic auth result in 500 errors
	STASH-3460	During directory synchronization, commit errors trigger cascading rollback failures
	STASH-3458	ResourceBundle ClassLoader lock contention under heavy load can soft-lock the system on Java 6
	STASH-3452	Cannot create pull request between same branch in different repos
	STASH-3448	Database errors during setup and migration produce useless messages
	STASH-3447	2.4 RestStashUser not backwards compatible
	STASH-3433	Files view shows an old version of README.md
	STASH-3427	500 error if the user session has expired when opening the merge dialog
	STASH-3393	Forking Git commands fails silently on Solaris when there is not enough memory
	STASH-3096	WARNING: Problem with directory [/home/stash/atlassian-stash-data/lib], exists: [false], isDirectory: [false], canRead: [false]

11 issues

8 May 2013 - Stash 2.4.1 (3 issues)

T	Key	Summary
	STASH-3411	Bad submodule file formatting causing 500 errors
	STASH-3410	ClientWebFragment NPE if an i18n bundle is available but no key given
	STASH-3405	2.4 Links to Stash Docs are pointing to the previous version

3 issues

6 May 2013 - Stash 2.4.0 (19 issues)

T	Key	Summary
	STASH-3354	Add decorator for project settings
	STASH-3249	Make rename/copy thresholds in git diff configurable
	STASH-3118	Do not show login form for logged in users
	STASH-3112	Change Gravatar server URL
	STASH-2816	Provide configurable whitespace settings on the diff view
	STASH-2784	Document specifying a custom JIRA issue key regex
	STASH-2643	Repository Permissions
	STASH-2495	Repository forking
	STASH-3294	Document user directory related limitations
	STASH-3145	Doc Task - Update - Running Stash as a Windows service
	STASH-3045	Document process for running Stash as a 64-bit Windows service
	STASH-3367	Viewing old pull requests sometimes shows "Activity not found"
	STASH-3346	NPE with SSO plugin enabled & Crowd directory set without SSO Domain
	STASH-3285	Profile javascript errors for users with numeric usernames starting with zero
	STASH-3277	Administration menu highlight for current page is broken
	STASH-3190	Merge button should refresh when pull request is approved
	STASH-3108	SQL Server URLs with instance names don't reload correctly
	STASH-2848	Unlicensed users can be added as reviewers if the default permission on the project is set to READ_ONLY

STASH-2602 "Clone in SourceTree" WebUI button uses HTTPS even when SSH is selected

19 issues

Stash security advisories

Finding and reporting a security vulnerability

Atlassian's channel for reporting security issues is detailed in How to Report a Security Issue.

Publication of security advisories

Atlassian's approach to publishing security advisories is detailed in Security Advisory Publishing Policy.

Severity levels

Atlassian's scale for measuring security issues is detailed in Severity Levels for Security Issues.

Our patch policy

Atlassian's approach to releasing patches is detailed in our Security Patch Policy.

Security advisories

- Stash security advisory 2012-09-04
- Stash security advisory 2014-02-26

Stash security advisory 2012-09-04

This advisory discloses a security vulnerability that we have found in Stash and fixed in Stash 1.1.2.

Customers who have downloaded and installed Stash should upgrade their existing Stash installations to fix this vulnerability.

Atlassian is committed to improving product security. The vulnerability listed in this advisory has been discovered by Atlassian, unless noted otherwise. The reporter may also have requested that we do not credit them.

If you have questions or concerns regarding this advisory, please raise a support request at http://support.atlassian.com/.

In this advisory:

- XSS Vulnerability

XSS Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as **High**, according to the scale published in Severity Levels for Security Issues. The scale allows us to rank the severity as critical, high, medium or low.
This is an independent assessment and you should evaluate its applicability to your own IT environment. This vulnerability is **not** of Critical severity.

Description

We have identified and fixed a persistent cross-site scripting (XSS) vulnerability that affects Stash instances, including publicly available instances (that is, Internet-facing servers). XSS vulnerabilities allow an attacker to embed their own JavaScript into a Stash page.

You can read more about XSS attacks at cgisecurity.com, The Web Application Security Consortium and other places on the web.

This vulnerability affects all supported versions of Stash, and has been fixed in Stash 1.1.2. This issue can be tracked here: STASH-2676 - Persistent Cross Site Scripting Vulnerability CLOSED

Risk Mitigation

We strongly recommend upgrading your Stash installation to fix this vulnerability. Please see the 'Fix' section below.

Fix

Upgrade

The vulnerability and fix version are described in the 'Description' section above.

We recommend that you upgrade to the latest version of Stash, if possible. For a full description of the latest version of Stash, see the release notes. You can download the latest version of Stash from the download centre.

Patches are not available for this vulnerability.

Stash security advisory 2014-02-26

This advisory details a critical security vulnerability that we have found in Stash and fixed in a recent versions of Stash.

- **Customers who have downloaded and installed Stash** should upgrade their existing Stash installations to fix this vulnerability.
- **Atlassian OnDemand customers** are not affected because OnDemand does not include Stash.

The vulnerability affects all versions of Stash up to and including 2.5.3, 2.6.4, 2.7.5 and 2.8.3.

It does **not** affect versions 2.5.4, 2.6.5, 2.7.6, 2.8.4, 2.9.X, 2.10.X.

Atlassian is committed to improving product security. We fully support the reporting of vulnerabilities and we appreciate it when people work with us to identify and solve the problem.

If you have questions or concerns regarding this advisory, please raise a support request at http://support.atlassian.com.

User privilege escalation

Severity

Atlassian rates the severity level of this vulnerability as **critical**, according to the scale published in Severity Levels of Security Issues. The scale allows us to rank the severity as critical, high, moderate or low.

This is an independent assessment and you should evaluate its applicability to your own IT environment.

Description

We have identified and fixed a vulnerability in Stash which allowed unauthenticated users to commit actions on behalf of any other authorised user. In order to exploit this vulnerability, an attacker requires access to your Stash web interface.

The Stash server is only vulnerable if it has been configured to be a part of an Application link with Trusted Applications authentication.

The vulnerability affects all supported versions of Stash up to and including 2.5.3, 2.6.4, 2.7.5 and 2.8.3. It has been fixed in the Stash security patch releases 2.5.4, 2.6.5, 2.7.6, 2.8.4. The vulnerability does not affect the Stash 2.9 and 2.10 releases. The issue is tracked in STASH-4122 - Privilege escalation CLOSED .

Risk Mitigation

If you are unable to upgrade or patch your Stash server you can do the following as a **temporary workaround**:

- Block access to your Stash server web interface from untrusted networks, such as the Internet.
- Remove any Application links that use Trusted Applications authentication and re-create them using OAuth.

Fix

This vulnerability can be fixed by upgrading Stash to one of the security patch releases or any release of Stash higher than 2.9.0. If required, there is also a patch available for this vulnerability for all supported versions of Stash. If you have any questions, please raise a support request at http://support.atlassian.com. We recommend upgrading.

The Security Patch Policy describes when and how we release security patches and security upgrades for our products.

Upgrading Stash

Upgrade to one of the Stash patch releases, 2.5.4, 2.6.5, 2.7.6, 2.8.4, which fixes this vulnerability, or one of the unaffected releases, 2.9 or a later version. For a full description of these releases, see the Stash Release Notes.

Patches

Binary patches are not available for this advisory. You need to either install one of the patch releases or apply recommended temporary workarounds.

Git resources

Get Git

See Installing and upgrading Git

On this page:

- Get Git
- Learning Git
- Getting started
- Git cheat sheets and other resources
- Git .mailmap

Learning Git

Git Tutorials and Training

Basic Git commands

Getting started

One "gotcha" when starting with Git is the way in which it pushes branches by default. On older versions of Git, pushing without arguments would push *all* branches that have the same name both locally and remotely. This can result in unexpected behaviour if you have old branches that complain when the remote branch is updated. It can even be quite dangerous if you do a force push and it reverts changes on the server. You can see the current value by running:

```
git config push.default
```

If this value is blank or 'matching', it is our recommendation that you reconfigure it to use 'upstream'.

```
git config --global push.default upstream
```

There has been some discussion around changing the default behaviour of Git.

Git cheat sheets and other resources

http://rogerdudler.github.com/git-guide/

http://byte.kde.org/~zrusin/git/git-cheat-sheet-medium.png

http://nvie.com/posts/a-successful-git-branching-model/

http://zrusin.blogspot.com.au/2007/09/git-cheat-sheet.html

http://ndpsoftware.com/git-cheatsheet.html#loc=workspace;

http://blog.fournova.com/2011/06/git-cheat-sheet/

http://jan-krueger.net/development/git-cheat-sheet-extended-edition

Git .mailmap

The Git `.mailmap` feature is useful locally, and in Stash repositories, to map multiple commit identities to the one Stash user – this can be used to tidy up your Git histories.

The Git documentation for `.mailmap` has configuration details (see the "MAPPING AUTHORS" section).

Basic Git commands

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

For more detail, check out the **Atlassian Git Tutorials** for a visual introduction to Git commands and workflows, including examples.

Git task	Notes	Git commands
Tell Git who you are	Configure the author name and email address to be used with your commits. Note that Git strips some characters (for example trailing periods) from `user.name`.	`git config --global user.name "Sam Smith"` `git config --global user.email sam@example.com`
Create a new local repository		`git init`
Check out a repository	Create a working copy of a local repository:	`git clone /path/to/repository`
	For a remote server, use:	`git clone username@host:/path/to/repository`

Add files	Add one or more files to staging (index):	`git add <filename>`
		
		`git add *`

Commit	Commit changes to head (but not yet to the remote repository):	`git commit -m "Commit message"`
	Commit any files you've added with `git add`, and also commit any files you've changed since then:	`git commit -a`
Push	Send changes to the master branch of your remote repository:	`git push origin master`
Status	List the files you've changed and those you still need to add or commit:	`git status`
Connect to a remote repository	If you haven't connected your local repository to a remote server, add the server to be able to push to it:	`git remote add origin <server>`
	List all currently configured remote repositories:	`git remote -v`
Branches	Create a new branch and switch to it:	`git checkout -b <branchname>`
	Switch from one branch to another:	`git checkout <branchname>`
	List all the branches in your repo, and also tell you what branch you're currently in:	`git branch`
	Delete the feature branch:	`git branch -d <branchname>`
	Push the branch to your remote repository, so others can use it:	`git push origin <branchname>`
	Push all branches to your remote repository:	`git push --all origin`
	Delete a branch on your remote repository:	`git push origin :<branchname>`
Update from the remote repository	Fetch and merge changes on the remote server to your working directory:	`git pull`

	To merge a different branch into your active branch:	`git merge <branchname>`

	View all the merge conflicts:
	View the conflicts against the base file:
	Preview changes, before merging:
		`git diff`
		`git diff --base <filename>`
		`git diff <sourcebranch> <targetbranch>`

	After you have manually resolved any conflicts, you mark the changed file:	`git add <filename>`
Tags	You can use tagging to mark a significant changeset, such as a release:	`git tag 1.0.0 <commitID>`
	CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:	`git log`
	Push all tags to remote repository:	`git push --tags origin`
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.	`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:	`git fetch origin` `git reset --hard origin/master`
Search	Search the working directory for `foo()`:	`git grep "foo()"`

Stash FAQ

On this page:

- The Stash product
  - Q: Why did you create a new product for Git repository management? Couldn't you build this into FishEye?
  - Q: What about Git repository management in FishEye and Crucible?
  - Q: Does FishEye require Stash? Does Stash require FishEye? Can they be used together?
  - Q: Will Stash be available for Atlassian Cloud?
  - Q: What is the difference between Stash Server and Stash Data Center?
- Repositories
  - Q: Does Stash support Mercurial (Hg)? What about other version control systems?
- Integration
  - Q: Does Stash work with JIRA? If so, what version of JIRA do I need to run Stash?
  - Q: Will Stash integrate with any other Atlassian Tools? Crowd? Bitbucket? SourceTree?
  - Licensing
  - Q: Does my Stash license have to match the number of users in my external directory (LDAP, Active Directory, Crowd or JIRA)?
  - Q: The number of users in my instance has exceeded my license count. Will Stash still work properly?
- Data recovery and backups
  - Q: Can I restore the .tar file created by the backup client into a database that is different from my original one (i.e. Oracle -> MySQL, etc.)?
  - Q: I forgot the user/password for my old database schema. How will I perform the restore? How does it work?
  - Q: What is the difference between the parameter pairs "stash.user & stash.password" and " jdbc.user & jdbc.password"?
  - Q: I backed up Stash of a particular version. Can I restore Stash to a newer release version?
- Troubleshooting
  - Q: I'm getting a "broken pipe" error when pushing my commits.

Related pages:

- Stash Data Center FAQ
- Stash Knowledge Base Home
- Support policies

Child pages:

- How do I change the external database password
- Stash home directory
- Raising a request with Atlassian Support
- Support policies
- Building Stash from source
- Contributing to the Stash documentation
- Collecting analytics for Stash

The Stash product

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

A: In FishEye 2.7 we added basic capabilities to host and manage Git repositories within FishEye. However, as we were planning future releases, we realized that the architecture of FishEye, built to index, browse and search across various SCMs, was not adequate for a DVCS repository management tool.

Therefore we have made the decision to build a new product, with a clear focus: hosting and managing Git

repositories. Instead of a "Jack of all trades", we will have two products that are focused on 2 very different tasks:

1. Stash – Host, manage and collaborate on Git repositories, and
2. FishEye – Track, search and browse Subversion, Perforce, Git, Mercurial and CVS repositories in one place.

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

A: *Internally managed* Git repositories were deprecated in FishEye and Crucible 2.8, and support for these was removed for the FishEye and Crucible 3.2 releases. We encourage those interested in Git repository management to check out Stash.

FishEye and Crucible will continue to deliver new features and enhancements to help users browse, search, review and visualize across different Version Control Systems *including Git*, Subversion, Mercurial, Perforce and CVS.

Q: Does FishEye require Stash? Does Stash require FishEye? Can they be used together?

A: FishEye and Stash are two separate standalone products that do not require each other.

If you are using multiple source code management systems (SCM) at your organization it makes sense to use both Fisheye and Stash. While you are managing your Git repositories with Stash, you can use FishEye to browse, search and reference code from other SCMs including Subversion.

Also, if you are using Git, Stash will provide your Git repository management, and FishEye will be a central place to keep track of changes and search for code across your repositories.

Q: Will Stash be available for Atlassian Cloud?

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

Q: What is the difference between Stash Server and Stash Data Center?

Stash Server is a single instance of Stash running on a single machine. It can only handle as much load as a single machine is capable of handling before performance degrades, and if the machine goes down for any reason (for example, hardware failure, network fault, or planned maintenance), then Stash is unavailable to users for the duration of the downtime.

Stash Data Center, on the other hand, looks like a single instance of Stash to users, but under the hood consists of a cluster of multiple machines ("cluster nodes") each running the Stash web application, behind a load balancer. This provides important benefits over Stash Server:

- **Performance at scale:** A cluster of many machines running Stash can handle more load than a single machine.
- **High availability:** If one cluster node goes down, then the remaining cluster node(s) can continue servicing requests so users should see little or no loss of availability.
- **Instant scalability:** You can rapidly provision extra capacity without downtime.

For more information see Stash Data Center and the Stash Data Center FAQ.

Repositories

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 - STASH-2469 - Include Mercurial (Hg) support in Stash OPEN

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 our documentation on 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.

Licensing

Q: Does my Stash license have to match the number of users in my external directory (LDAP, Active Directory, Crowd or JIRA)?

A: No. You can control which users in your external directory have access to Stash, so that the license limit is not exceeded. A user is by definition any account that has permission to log into the Stash application. If you synchronise Stash with an external user directory, you can grant access to Stash to a subset of users, so as to stay below your license limit. The Global permissions page explains in detail how to manage login rights for users and groups in Stash.

Q: The number of users in my instance has exceeded my license count. Will Stash still work properly?

A: As stated in the Global permissions document, any user assigned "Stash User" permission or higher, granted to the individual or via a group, will count towards the license limit. Stash will not allow you to grant the "Stash User" permission if this will exceed the license limit while manually adding users using Stash UI. However, if you happen to exceed the license limit by connecting your Stash instance to a User Directory that contains more users than you license allows you to have, Stash will display a banner with the content below:

> You have more users than your license allows.
>
> Users will not be able to push commits to repositories until you restrict the number of active users to match your license or you upgrade your current license.

To fix that bear in mind that users don't have to be removed from the database in order to reduce their license count. You merely have to make sure that they no longer have access to Stash via the "Stash User" permission through individual or group assignments.

Data recovery and backups

Q: Can I restore the .tar file created by the backup client into a database that is different from my original one (i.e. Oracle -> MySQL, etc.)?

A: Yes you can, as long as you specify all the `jdbc` parameters (`jdbc.override`, `jdbc.driver`, etc.) when running the restore. Please read Restoring Stash into a newly created DB for more details.

Q: I forgot the user/password for my old database schema. How will I perform the restore? How does it work?

A: As described in Restoring Stash into a newly created DB, the restore client will only restore into an empty home directory and an **empty database**. The new database should be configured following the instructions in Connecting Stash to an external database and its sub-page that corresponds to your database type. If you want to use a different type of database or a different user/password, you just need to specify all the `jdbc` parameters (`jdbc.override`, `jdbc.driver`, etc) when running the restore.

Q: What is the difference between the parameter pairs "`stash.user` & `stash.password`" and "`jdbc.user` & `jdbc.password`"?

A: `stash.user` and `stash.password` hold the credentials for a Stash sys admin user. They are only used during the **backup** procedure so that the backup client can lock Stash and instruct Stash to perform the database-agnostic backup. The backup client does not need database credentials because the Stash system performs the database backup.

`jdbc.user` and `jdbc.password` are only used during the **restore** procedure when `jdbc.override` is set to `true`. They are used to connect to the newly-installed database.

Q: I backed up Stash of a particular version. Can I restore Stash to a newer release version?

A: No. You need to use the same Stash binary as the one originally used to back up your instance. Note that using an older Stash binary will result in an error – downgrades are not possible. See Using the Stash Backup Client for details on the backup/restore procedure.

Once you have restored Stash, you can upgrade to a newer version of Stash following the instructions in the Stash upgrade guide.

Troubleshooting

Q: I'm getting a "broken pipe" error when pushing my commits.

A: This error occurs when the amount of data you're trying to push in one go exceeds Git's http post buffer. Just run the following command to increase it to 500MB.

```
git config http.postBuffer 524288000
```

See Git push fails with 'fatal: The remote end hung up unexpectedly'.

How do I change the external database password

You can change the password the Stash uses to connect to an external database, however you don't do this from the Stash Administration area – you must follow the procedure described below.

Related pages:

- Connecting Stash to an external database

To change the password that Stash uses when connecting to an external database:

1. Stop Stash. See Starting and stopping Stash.
2. Get your database administrator to change the password on your database.
3. Go to your Stash home directory.

   Edit the `stash-config.properties` file to change the line that looks like:

   ```
   jdbc.password=MY_PASSWORD
   ```

replacing `MY_PASSWORD` with your new database password.

4. Restart Stash. See Starting and stopping Stash.

Stash home directory

What is the Stash home directory?

> The Stash home directory is created automatically by the Stash installer – see Getting started to install and start using Stash.
>
> The information on this page only applies if you are manually installing or upgrading Stash.

The Stash home directory is where your Stash data is stored. The home directory location is defined either by the `STASH_HOME` environment variable, or in the `STASH_HOME` line of:

- Windows: `<Stash installation directory>\bin\setenv.bat`
- Linux and Mac: `<Stash installation directory>/bin/setenv.sh`

⚠ Stash 3.5 and later versions no longer allow the Stash home directory to be the same directory as, or a subdirectory of, the `<Stash installation directory>`. The Stash home directory, as defined by the `STASH_HOME` variable, must be in a separate location – Stash will fail on startup otherwise. And by the way, you'll need separate Stash home directories if you want to run multiple instances of Stash (when these are not nodes for a Stash Data Center).

⚠ Where possible, you should choose a location for your Stash home directory that will *never* need to be moved. Some home contents are location-sensitive, so moving the home directory may corrupt them. Stash attempts to update contents when it detects that the home directory has moved, but the safest approach is to avoid the issue altogether by leaving the home directory in the same location.

What does the Stash home directory contain?

Your Stash home directory contains the following directories and files:

Path	Description
`caches`	Contains cache and index files. It should be safe for these files to be deleted between application restarts; however, these files must not be modified or deleted while Stash is running.
`shared/config`	Contains application configuration.
`shared/data`	Contains the Git repositories, project avatars, and the embedded HSQL database if an external database is not configured.
`export`	Contains dump files produced during database migrations.
`lib`	As of Stash 2.1, can contain third-party jars such as the MySQL JDBC driver.
`lib/native`	As of Stash 2.1, can contain *native libraries*, such as Tomcat's APR-based native library.
`log`	Contains log files for Stash.
`shared/plugins`	Contains plugin related data (such as externally uploaded plugins) for Stash.
`tmp`	Contains temporary files created by the system. Its contents can safely be deleted when Stash is *not running*.

`shared/stash-config.properties`	Allows configuring various aspects of how Stash behaves, such as its database connection pool size and the location of the Git binary to use. This file will be created automatically during a database migration. It can be created manually otherwise. See Stash config properties for further information.

Setting the Stash home directory

Note that the Stash home directory is created automatically by the Stash installer – see Getting started.

Only when you are installing Stash from an archive file will you need to set the value of `STASH_HOME` yourself, as described in this section.

Click here if setting STASH_HOME on Linux or Mac...

The Stash home directory is where your Stash data is stored.

Create your Stash home directory (without spaces in the name), and then tell Stash where you created it by editing the `<Stash installation directory>/bin/setenv.sh` file – uncomment the `STASH_HOME` line and add the absolute path to your home directory. Here's an example of what that could look like when you're done:

```
#
if [ "x${STASH_HOME}" = "x" ]; then
    export STASH_HOME="/home/username/stash-home"
fi
```

Click here if setting STASH_HOME on Windows...

The Stash home directory is where your Stash data is stored.

Create your Stash home directory, and then tell Stash where you created it by setting a `STASH_HOME` environment variable, as follows.

For Windows 7:

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "`STASH_HOME`" as the **Variable name**, and the absolute path to your Stash home directory as the **Variable value**. Don't use a trailing backslash.

There are a few things to know about setting up the Stash home directory on Windows that will make life easier:

- You *should not* locate your Stash home directory inside the `<Stash installation directory>` — they should be entirely separate locations. If you do put the home directory in the `<Stash installation directory>` it will be overwritten, and lost, when Stash gets upgraded. And, by the way, you can't use the same Stash home directory for multiple instances of Stash.
- Keep the path length to the Stash home directory as short as possible. See Stash always shows incorrect Merge Conflict in PRs for an explanation.
- Don't use spaces in the path to the Stash home directory.

Securing the Stash home directory

The internal database files, the migration dump files and `stash-config.properties` all contain information that may be considered secret (server settings, salted and hashed user passwords, database passwords, etc).

For production use, we strongly recommend that you secure this directory against unauthorised access.

We recommend the following precautions:

- Assign a separate restricted user account on the machine for running Stash (not a root/administrator user)
    - If you wish to run Stash on port 80, use a separate http front end as described in Integrating Stash

with Apache HTTP Server (do not run as root/Administrator if security of the home directory is important to you)
- Ensure that only the user running Stash can access the Stash home directory, and that this user has read, write and execute permissions, by setting file system permissions appropriately for your operating system.

About the repositories

As noted above, `data` contains the Git repositories being managed by Stash, where "managed by Stash" are the operative words. The repositories are for *Stash* to interact with, and they are configured and managed accordingly. They are *not* a mechanism for configuring Stash behaviour. We *strongly* recommend that customers never modify them, nor interact with them directly. They are *intentionally* structured in a way which does not lend itself well to direct interaction.

Being Git repositories, there are certainly standard aspects to how the repositories on disk are stored and how they function. However, the exact way they are configured *can and does* change between Stash releases. Stash makes *no effort* to preserve unexpected configuration changes which have been applied by customers, and such changes may cause failures at runtime or during upgrades. If there is an aspect of Stash's behaviour you wish to configure, please open a feature request on jira.atlassian.com rather than trying to modify the repositories directly.

> Repositories are *location sensitive*. Moving your Stash home directory will result in the system being locked (briefly) on startup while Stash updates the repositories on disk. Assuming the updates are applied successfully, the system will then unlock itself for normal usage.
>
> Where possible, please choose a Stash home location which will not need to be changed later.

Raising a request with Atlassian Support

If you encounter any problems when setting up or using Stash, please let us know — we're here to help!

You may want to search the following first:

- the Atlassian Answers site (the Stash forum), where Atlassian staff and Stash users can answer your questions.
- the Stash Knowledge Base.

If you've found a bug in Stash, or want to request a feature or improvement, raise a ticket in the Stash project of our public issue tracker. Try searching for similar issues - voting for an existing issue is quicker, and avoids duplicates.

If you still need assistance, please raise a support request, either from within Stash or on the Atlassian Support site, as described in the following sections.

Providing as much information as possible about your Stash installation with your initial request will help our Support Engineers to give you a faster and more complete response.

On this page:

- Raising a Support Request from within Stash
- Raising a Support request yourself at Atlassian Support
- Information you should provide

Raising a Support Request from within Stash

This method depends on having a mail server configured for Stash that supports large zip file attachments.

1. Log in to Stash (as a System Administrator) and go to the admin area.
2. Click **Atlassian Support Tools** (under 'Support') then **Support Request**.
3. Provide as much information as possible in the Description, including steps to replicate the problem, and any error messages that are appearing on the console or in the logs. For performance issues, please include profiling logs. See the section below about information you should provide.
4. Click **Send**.

This will produce a zip file containing the information categories selected from the list and will email this to Atlassian Support. You will receive an email advising you of details of the Support Request that was automatically created, and you will receive emailed updates about progress on your issue. You can also see the status of your request directly by visiting the Atlassian Support System.

Raising a Support request yourself at Atlassian Support

1. Log in to Stash (as a System Administrator) and go to the admin area.
2. Click **Atlassian Support Tools** (under 'Support') then **Support Zip**.
3. Select information categories to include in the zip file.
4. Click **Create**.

The zip file is created in the home directory of the Stash server, for example `<STASH_HOME>\export\Stash_support_2013-11-17-20-49-18.zip`.

When you now go to Atlassian Support and create a Support Request, you can attach the Support Zip file to the request.

Please provide as much information as possible in the request, including steps to replicate the problem, and any error messages that are appearing on the console or in the logs. For performance issues, please include profiling logs. See the section below about information you should provide.

Information you should provide

In addition to the logs and configuration information that you can include in the Support Request zip file, the following information can help to give you a faster response:

Environment details

- Stash version
- Java version (for example OpenJDK 1.7.0 JRE)
- Git and Perl versions
- Operating system (for example, Windows 7, Mac OS X 10.6.8)
- Database type (for example, MySQL) and version
- Browsers and versions
- Network topology - is Stash running behind a reverse proxy? Is that secured using HTTPS (SSL)?

Configuration

- Java settings, including JVM_MINIMUM_MEMORY, JVM_MAXIMUM_MEMORY

Logs

You may need to adjust the logging level, or enable profiling in Stash, in order to get more detailed logs. See Stash debug logging.

- Debug logs – Stash debug logs can be found in `<STASH_HOME>/log`.
- Profiling logs – Stash profiling logs can help with analyzing performance issues and can be found in `<STASH_HOME>/log`.

Performance factors

- Number of concurrent Git clones
- Number of users
- The size of the .git directory
- CPU spec, number of cores, whether hyperthreading is enabled
- RAM and cache sizes

Integrations

- Other Atlassian applications (and their versions)
- Which build servers are integrated with Stash, if any?
- Are Application Links configured?

Support policies

Welcome to the support policies index page. Here, you'll find information about how Atlassian Support can help you and how to get in touch with our helpful support engineers. Please choose the relevant page below to find out more.

- Bug fixing policy
- New features policy
- Security Bugfix Policy

To request support from Atlassian, please raise a support issue in our online support system. To do this, visit support.atlassian.com, log in (creating an account if need be) and create an issue under 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 the appropriate project on http://jira.atlassian.com to report bugs for Atlassian products.

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.

New features policy

Summary

- We encourage and display customer comments and votes openly in our issue tracking system, http://jira.atlassian.com.
- We do not publish roadmaps.
- Product Managers review our most popular voted issues on a regular basis.
- We schedule features based on a variety of factors.
- Our Atlassian Bug Fixing Policy is distinct from this process.
- Atlassian provides consistent updates on the top 20 issues.

How to Track what Features are Being Implemented

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

How Atlassian Chooses What to Implement

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

- **Customer contact**: We get the chance to meet customers and hear their successes and challenges at Atlassian Summit, Atlassian Unite, developer conferences, and road shows.
- **Customer interviews**: All product managers at Atlassian do customer interviews. Our interviews are not simply to capture a list of features, but to understand our customers' goals and plans.
- **Community forums**: There are large volumes of posts on answers, of votes and comments on jira.atlassian.com, and of conversations on community forums like groups on LinkedIn.
- **Customer Support**: Our support team provides clear insights into the issues that are challenging for customers, and which are generating the most calls to support
- **Atlassian Experts**: Our Experts provide insights into real-world customer deployments, especially for customers at scale.
- **Evaluator Feedback**: When someone new tries our products, we want to know what they liked and disliked and often reach out to them for more detail.
- **In product feedback**: The JIRA Issue Collectors that we embed our products for evaluators and our Early Access Program give us a constant pulse on how users are experiencing our product.
- **Usage data:** Are customers using the features we have developed?
- **Product strategy:** Our long-term strategic vision for the product.
- Please read our post on Atlassian Answers for a more detailed explanation.

How to Contribute to Feature Development

Influencing Atlassian's release cycle
We encourage our customers to vote on issues that have been raised in our public JIRA instance, http://jira.atlas

sian.com. Please find out if your request already exists - if it does, vote for it. If you do not find it you may wish to create a new one.

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

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

Further reading

See Atlassian Support Offerings for more support-related information.

Security Bugfix Policy

See Security @ Atlassian for more information on our security bugfix policy.

Building Stash from source

This page has moved!

To our Development Hub at https://developer.atlassian.com/display/STASHDEV/Building+from+Source+Code.

But you really wanted to build a plugin anyway, right?

Contributing to the Stash documentation

Would you like to share your Stash hints, tips and techniques with us and with other Stash users? We welcome your contributions.

Blogging your technical tips and guides

Have you written a blog post describing a specific configuration of Stash or a neat trick that you have discovered? Let us know, and we will link to your blog from our documentation.

Contributing documentation in other languages

Have you written a guide to Stash in a language other than English, or translated one of our guides? Let us know, and we will link to your guide from our documentation.

On this page:

- Blogging your technical tips and guides
- Contributing documentation in other languages
- Updating the documentation itself
  - Getting permission to update the documentation
  - Our style guide
  - How we manage community updates

Updating the documentation itself

Have you found a mistake in the documentation, or do you have a small addition that would be so easy to add yourself rather than asking us to do it? You can update the documentation page directly

Getting permission to update the documentation

Please submit the Atlassian Contributor License Agreement.

Our style guide

Please read our short guidelines for authors.

How we manage community updates

Here is a quick guide to how we manage community contributions to our documentation and the copyright that applies to the documentation:

- **Monitoring by technical writers.** The Atlassian technical writers monitor the updates to the documentation spaces, using RSS feeds and watching the spaces. If someone makes an update that needs some attention from us, we will make the necessary changes.
- **Wiki permissions.** We use wiki permissions to determine who can edit the documentation spaces. We ask people to sign the Atlassian Contributor License Agreement (ACLA) and submit it to us. That allows us to verify that the applicant is a real person. Then we give them permission to update the documentation.
- **Copyright.** The Atlassian documentation is published under a Creative Commons CC BY license. Specifically, we use a Creative Commons Attribution 2.5 Australia License. This means that anyone can copy, distribute and adapt our documentation provided they acknowledge the source of the documentation. The CC BY license is shown in the footer of every page, so that anyone who contributes to our documentation knows that their contribution falls under the same copyright.

Collecting analytics for Stash

We are continuously working to make Stash better. Data about how you use Stash helps us do that. We have updated our Privacy Policy so that we may collect usage data automatically unless you disable collection. The data we collect includes information about the systems on which your installation of Stash is operating, the features you use in Stash, and your use of common IT terminology within the product. For more details, see our Privacy Policy, in particular the 'Analytics Information from Downloadable Products' section.

See also our End User Agreement.

How to change data collection settings?

You can opt in to, or out of, data collection at any time. A Stash admin can change the data collection settings by going to **Analytics** (under 'Settings') in the Stash admin area.

How is data collected?

We use the Atlassian Analytics plugin to collect event data in Stash. Analytics logs are stored locally and then periodically uploaded to a secure location.