HelixSwarm

Helix Swarm Guide

2017.3

October 2017

PERFORCE

www.perforce.com

Contents

How to Use this Guide

This guide tells you how to use Helix Swarm, for collaboration and code review for teams using Helix Versioning Engine. It is intended for anyone using Swarm to perform code review tasks with Helix Core.

Feedback

How can we improve this manual? Email us at manual@perforce.com.

Other Helix Core documentation

See https://www.perforce.com/support/self-service-resources/documentation.

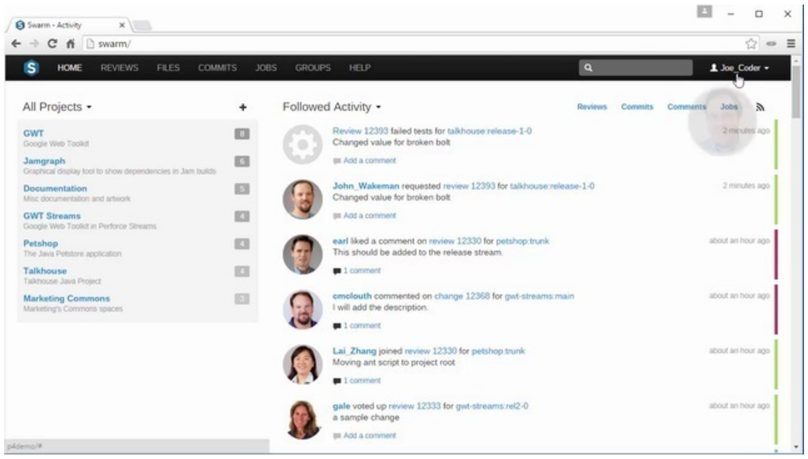
Syntax conventions

Helix documentation uses the following syntax conventions to describe command line syntax.

Notation	Meaning
`literal`	Must be used in the command exactly as shown.
italics	A parameter for which you must supply specific information. For example, for a *serverid* parameter, supply the ID of the server.
[`-f`]	The enclosed elements are optional. Omit the brackets when you compose the command.
...	■ Repeats as much as needed: • `alias-name[[$(arg1)... [$(argn)]]=transformation` ■ Recursive for all directory levels: • `clone perforce:1666 //depot/main/p4... ~/local-repos/main` • `p4 repos -e //gra.../rep...`
element1 | *element2*	Either *element1* or *element2* is required.

1 | Helix Swarm

1. Watch Swarm Overview



GWT	Google Web Toolkit	8
Jamgraph	Graphical display tool to show dependencies in Jam builds	6
Documentation	Misc documentation and artwork	5
GWT Streams	Google Web Toolkit in Perforce Streams	4
Petshop	The Java Petstore application	4
Talkhouse	Talkhouse Java Project	4
Marketing Commons	Marketing's Commons spaces	3

Review 12393 failed tests for talkhouse:release-1-0 — 2 minutes ago
Changed value for broken bolt
Add a comment

John_Wakeman requested review 12393 for talkhouse:release-1-0 — 2 minutes ago
Changed value for broken bolt
Add a comment

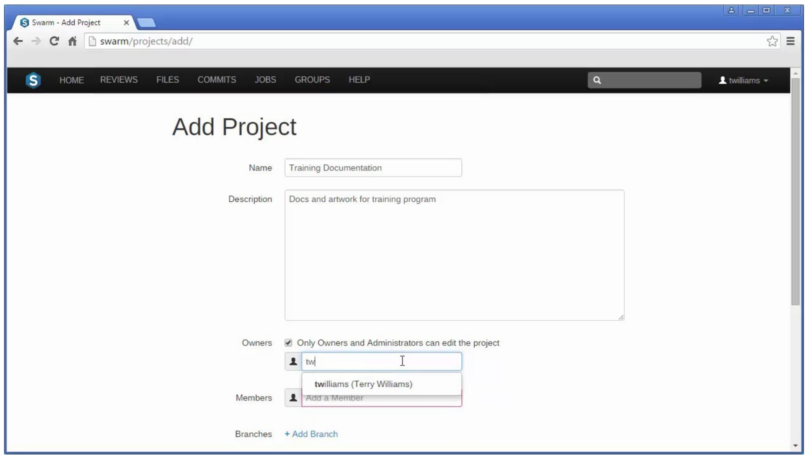
earl liked a comment on review 12330 for petshop:trunk — about an hour ago
This should be added to the release stream.
1 comment

cmclouth commented on change 12368 for gwt-streams:main — about an hour ago
I will add the description.
1 comment

Lai_Zhang joined review 12330 for petshop:trunk — about an hour ago
Moving ant script to project root
1 comment

gale voted up review 12333 for gwt-streams:rel2-0 — about an hour ago
a sample change
Add a comment

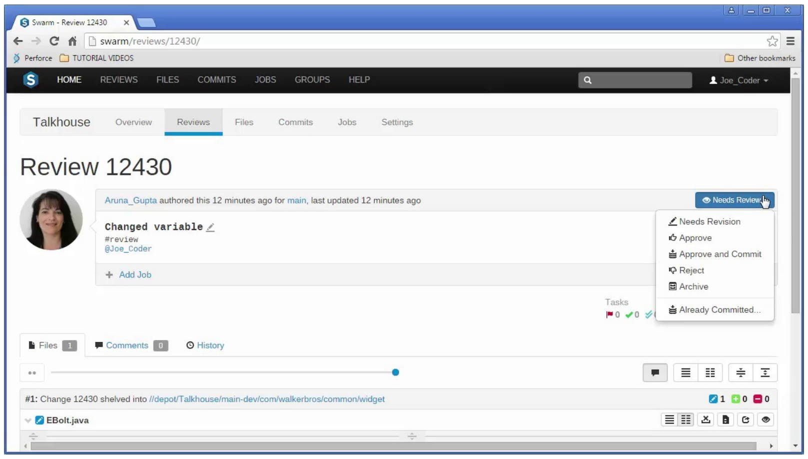
2. Create a Project



Add Project

Name	Training Documentation
Description	Docs and artwork for training program
Owners	✓ Only Owners and Administrators can edit the project
	tw
	twilliams (Terry Williams)
Members	Add a Member
Branches	+ Add Branch

3. Create a Review



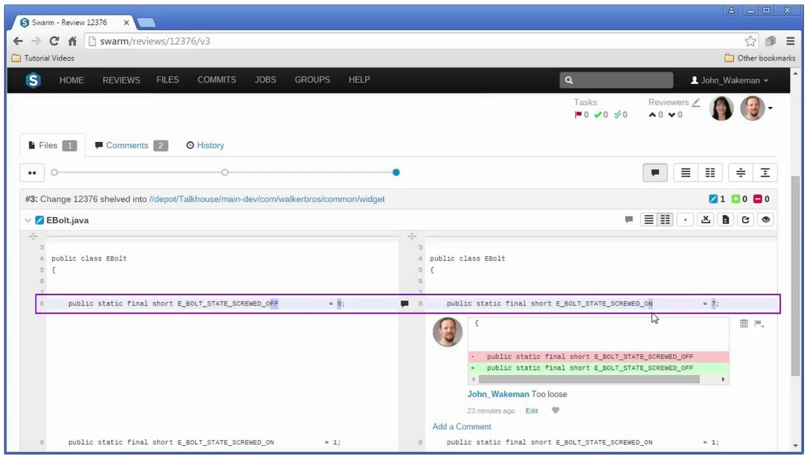
Review 12430

Aruna_Gupta authored this 12 minutes ago for main, last updated 12 minutes ago

```
Changed variable
#review
@Joe_Coder
```

#1: Change 12430 shelved into //depot/Talkhouse/main-dev/com/walkerbros/common/widget

4. Collaborate on Reviews



#3: Change 12376 shelved into //depot/Talkhouse/main-dev/com/walkerbros/common/widget

```
public class EBolt
{

    public static final short E_BOLT_STATE_SCREWED_OFF        = 0;
```

```
public class EBolt
{

    public static final short E_BOLT_STATE_SCREWED_ON         = 7;
```

```
{

-   public static final short E_BOLT_STATE_SCREWED_OFF
+   public static final short E_BOLT_STATE_SCREWED_OFF
```

John_Wakeman Too loose

23 minutes ago · Edit

```
    public static final short E_BOLT_STATE_SCREWED_ON         = 1;
```

Swarm enables collaboration and code review for teams using Helix server, helping ship quality software faster. Review code before or after commit, bring continuous integration into the review, and merge work that passed the review.

This documentation is structured with quickstart topics up front, for anyone experienced with code review but unfamiliar with Swarm, followed by chapters covering Swarm operations including installing and configuring Swarm, basic features, managing projects, code reviews, integrations, Swarm administration, guidance on extending Swarm.

Download Swarm here.

What's new in 2017.3

This section provides a summary of the notable changes in Swarm for the 2017.3 release. Full details are available in the distribution's `RELNOTES.txt` file.

Be aware:

Upgrade index on upgrade to improve performance

The upgrade process has changed. The Swarm index must be upgraded to ensure that the Swarm review history is displayed in the correct order. This step is only required the first time you upgrade your Swarm system to 2017.3 or later. Subsequent Swarm upgrades do not require the index to be upgraded. If this is a new Swarm installation, the index does not need to be upgraded.
See " Upgrade index" on page 119 for details

Major new functionality

Groups as reviewers

Swarm groups can now be set as reviewers for reviews. A reviewer group can be set as an optional reviewer, a required reviewer (all votes required), or a required reviewer (one vote required).
See "Reviewers" on page 214 for details.

Groups as project moderators

Swarm groups can now be set as moderators for project branches. When a group is specified as a moderator, all of the members of that group have the same moderator privileges for that project branch as if they were added individually.
See "Add a project" on page 188 for details.

Group mailing list and additional group notifications

Swarm groups can now have a group mailing list email address, notification emails are sent to the group email address rather than the group members individual email addresses. When **group mailing list** is enabled for a group, additional notifications can be set for the group.
See "Add a group" on page 180 for details.

Group avatars

Swarm groups now have avatars, these can be customized in the same way as a user avatar. The group avatar is used in reviews when the group is set as a reviewer, in projects when the group is a project member, and in projects when the group is a moderator of a project branch.
See "Avatars" on page 241 for details.

Group @@mention

You can now use `@@mention` for groups. When a user @@mentions a group in a comment, a list of groups is provided via a dropdown menu. This helps ensure that the right group of people are notified.

Using `@@mention` in a review, changelist, or comment causes the referenced groupid members to receive a notification and be included in any future notifications regarding the associated file or review. When a comment is added to a job, Swarm sends a notification to groups listed in group fields in the job, groups @@mentioned in the job description, and the authors of any associated changes.
See "@mention" on page 171 for details.

Security Enhanced Linux on CentOS 7

Swarm now supports SELinux on CentOS 7. SELinux is an advanced access control mechanism that improves security for Linux distributions.
See "Security-enhanced Linux (SELinux)" on page 47 for details.

Review sort order and user filter

You can now choose to sort reviews by review **Created** or **Last Activity** from a dropdown menu. The User filter dropdown now has a new filter called **Reviews I've Authored Or Am Participating in**.
See "Review queues" on page 204 for details.

API updated to version 7

A number of new endpoints have been added to the API, and it has been updated to v7.
See "Swarm API" on page 327 for details.

Minor new functionality

User notification options updated

The user notification settings are now ordered by user role.
See "Settings" on page 153 for details.

User email shown on user profile sidebar

The user email address is now shown in the sidebar below the user avatar.
See "Users" on page 152 for details

Group email shown on group sidebar

If a group has a group mailing list, the group mailing list email address is now shown in the sidebar below the group avatar.
See "Viewing a group" on page 157 for details

Bell icons removed from groups page

The bell notification icons have now been removed from the groups page.

Fixed review state filters in IE11

Review state filters were not working in Microsoft Internet Explorer 11, they have now been fixed

Known limitations

Access Control

Swarm maintains a variety of information in the Helix Versioning Engine's keys facility. By default, users with list-level privileges can read these keys, which can include comments that contain excerpts of code they may not normally have access to.
The Helix Versioning Engine, version 2013.1/659207 or higher, has a configuration setting to require admin-level privileges for access to read and write keys. See "Hiding Swarm storage from regular users" on page 98.

Task Stream Reviews

Pre-commit reviews in a task stream are not yet supported.

2 | Quickstart

Often, the best way to learn how software works is to try it. If the interface is sufficiently discoverable, you may not need to refer to documentation much, or at all. Sometimes, you just need to see the required steps to complete a task; that's what this chapter is all about.

How do I start a code review?

There are several ways:

Important
If your Helix Versioning Engine is configured as a commit-edge deployment, and your normal connection is to an edge server, Swarm refuses to start reviews for shelved changes that have not been promoted to the commit server.

Within Swarm, this means that the **Request Review** button does not appear for unpromoted shelved changes. Outside of Swarm, attempts to start reviews for unpromoted shelved changelists appear to do nothing. Ask your Helix Versioning Engine administrator for assistance if you cannot start a review.

An administrator of the Helix Versioning Engine can automatically promote shelved changes to the commit server by setting the configurable `dm.shelve.promote` to `1`.

1. When you use Swarm to view a shelved changelist, click the **Request Review** button to request a review of that shelved change.

   Note
   To view a shelved changelist, use a "Quick URLs" on page 170. For example, if your shelved change is 54321, visit the URL: `https://myswarm.url/`*`54321`*

2. When you use Swarm to view a submitted change, click the **Request Review** button to request a review of that change.

3. When you are about to shelve or submit files:

   a. Include `#review` within your changelist description (separated from other text with whitespace, or on a separate line).

      Once the review begins, Swarm replaces `#review` with `#review-12345`, where `12345` is the review's identifier.

      Note
      The `#review` keyword is customizable. For details, see "Review keyword" on page 279.

   b. At this time, you can add reviewers to the code review by using @mention for users, and @@mention for groups in the changelist description for each desired reviewer.

      If your `@mention` or `@@mention` includes an asterisk (`*`) before the *userid* or *groupid*, for example `@*userid`, that user or all of the group members become required reviewers. If your `@@mention` includes an exclamation mark (`!`) before the *groupid*, for example `@@!groupid`, the members of that group become required reviewers but only one member of the group is required to vote. See "Required reviewers" on page 229 for details.

   c. Complete your shelve or submit operation.

      Warning
      If you shelve a change and subsequently edit the description to include `#review`, a review is **not started**. You must re-shelve the files after adding `#review`.

4. When you use Git Fusion, you can start a review by pushing your changes to a target branch using the following command:

   ```
   $ git push origin task1:review/master/new
   ```

   `task1` is the name of the current Git task branch, and master is the target branch that the proposed changes are intended for.

   > **Important**
   > The target branch must be mapped to a named Helix Core branch in the Git Fusion repo configuration.
   >
   > See "Setting up Repos" in the *Git Fusion Guide* for details on converting a lightweight branch into a fully populated Helix Core branch.

   When the command completes, the output indicates the *review id* that has been created:

   ```
   remote: Perforce: Swarm review assigned: review/master/1234
   ```

   where *1234* is the review id that was just created.

   For more information on Git Fusion, see the *Git Fusion Guide*.

   > **Note**
   > If you are using P4V and its Swarm integration, and you encounter the error Host requires authentication, ask your Helix Versioning Engine administrator for assistance. See "P4V Authentication" on page 247 for details.

Once a review has started

Wait for someone else to review your code, or see "How do I contribute comments or code changes to a code review?" below. More review activities are available.

How do I contribute comments or code changes to a code review?

Contribute to a code review by adding comments to the review, adding comments to specific lines within the review's files, or making edits to the review's files.

Commenting on a review

1. Visit the review's page.
2. On the **Comments** tab, add your comment in the provided text area.
3. Click **Post**.

Commenting on a specific line in a file

1. Visit the review's page.
2. On the **Files** tab, click a line you would like to comment on.

   The comment text area appears.
3. Add your comment in the provided text area.
4. Click **Post**.

Editing files in a review

1. Get a local copy of the review's files.

   > **Note**
   > If you are using Git Fusion, follow the steps in the next section.

2. Edit the files as required.
3. Prepare a changelist with the edited files and include `#review-1234` within the changelist's description (separated from other text with whitespace, or on a separate line), where `1234` is the review's identifier.

   > **Warning**
   > If you use an invalid review identifier, it will appear that nothing happens. Swarmis currently unable to notify you of this situation.

4. Depending on the model of code review you are using, you would:
   - Shelve the files (for pre-commit).
   - Submit the files (for post-commit).

Editing files in a review with Git Fusion

> **Important**
> You can only update Git Fusion-initiated reviews using Git Fusion.

In the following example, the current local task branch is ***`task1`***, the target branch is ***`master`***, the review id is ***`1234`***, the Git Fusion hostname is ***`gfserver`***, and the remote repo name is ***`p4gf_repo`***.

1. Fetch the review's head version:

```
$ git fetch --prune origin
From gfserver:p4gf_repo
```

```
* [new_branch]      review/master/1234 ->
origin/review/master/1234
x [deleted]         (none)     -> origin/review/master/new
```

The `--prune` option lets the local Git repo delete the unwanted review/master/new reference created by the initial `git push origin task1:review/master/new` command.

2. Check out the review's head version:

```
$ git checkout review/master/1234
```

3. Edit the files as required.
4. Add the edited files to the index of files, in preparation for the next commit.

   There are several ways to do this. For example, to add all modified files to the index, run:

```
$ git add -A
```

5. Commit the files in Git:

```
$ git commit -m "made some changes"
```

6. Push the Git changes to the review:

```
$ git push origin review/master/1234
```

Note
If you get review feedback that is better expressed as a Git rebase and cleaned up history, you can make your changes and push them as a new review.

You cannot clean up history and then push your changes to the same review.

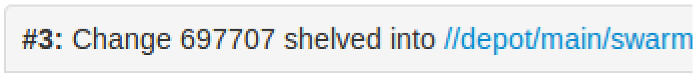
For more information on Git Fusion, see the *Git Fusion Guide*.

How do I get a local copy of the review's code for evaluation?

Swarm manages one or more changelists containing shelved copies of all of the files belonging to a specific review. You can unshelve the files to receive a copy of the review's code, or you can click the Download .zip to download a ZIP archive containing all of the review's files.

Determine the changelist containing the review's files

1. Visit the review's page.
2. The current review version's changelist appears in the file list heading.

   #3: Change 697707 shelved into //depot/main/swarm

   In this example, the changelist is 697707. You use the identified changelist in place of shelved changelist below.

Note
Swarm can version file updates in reviews. For more information, see "Review display" on page 211.

Using P4

For a shelved changelist, use a command-line shell and type:

```
$ p4 unshelve -s shelved changelist
```

For a committed changelist, use a command-line shell and type:

```
$ p4 sync @committed changelist
```

Note
Your client's view mappings need to include the changelist's path.

Using P4V

For a shelved changelist:

1. Select **Search > Go To**.
2. Change the select box to **Pending Changelist**.
3. Type in the shelved changelist number and click **OK**.
4. Select the files in the **Shelved Files** area.
5. Right-click and select **Unshelve**.
6. Click **Unshelve**.

For a committed changelist:

1. Select **Search > Go To**.
2. Change the select box to **Submitted Changelist**.

3. Type in the submitted changelist number and click **OK**.
4. Select the files in the **Files** area.
5. Right-click and select **Get this Revision**.
6. Click **Close**.

Using Git Fusion

Git Fusion-initiated reviews include the Git logo beside the main review identifier. This indicator is important because Perforce users cannot update Git Fusion-initiated reviews.

Review 773273

In the following example, the current local task branch is ***task1***, the target branch is ***master***, the review id is ***773273***, the Git Fusion hostname is ***gfserver***, and the remote repo name is ***p4gf_repo***.

1. Fetch the review's head version:

   ```
   $ git fetch --prune origin
   From gfserver:p4gf_repo
   * [new_branch]      review/master/773273 ->
   origin/review/master/773273
   x [deleted]         (none)     -> origin/review/dev/new
   ```

   The `--prune` option lets the local Git repo delete the unwanted review/master/new reference created by the initial `git push origin task1:review/master/new` command.

2. Check out the review's head version:

   ```
   $ git checkout review/master/773273
   ```

Important
You can only update Git Fusion-initiated reviews using Git Fusion.

For more information on Git Fusion, see the *Git Fusion Guide*.

Downloading a ZIP archive

When the `zip` command-line tool is available, Swarm can provide a ZIP archive containing all of the files in a review. The version of the files downloaded matches those displayed when using the "Review timeline" on page 218.

Note
The **Download .zip** button does not appear if the `zip` command-line tool is not available.

When you click the **Download .zip** button, Swarm performs the following steps:

1. Scan the review's files, to determine if you have permission to access their contents (according to the Helix Versioning Engine protections), and if the total file size is small enough to be processed by Swarm.
2. Sync the file contents to the Swarmserver from the Helix Versioning Engine.
3. Create the ZIP archive by compressing the file content.
4. Start a download of the generated ZIP archive.

You might not see all of the above steps; Swarm caches the resulting ZIP archives so that repeated requests to download the same review files can skip the sync/compress steps whenever possible.

If an error occurs while scanning, syncing, or compressing, Swarm indicates the error.

For information on the configuration for ZIP archives, see "Archives configuration" on page 240.

How can I fix 'not mergeable' errors in a review?

The problem can occur when you attempt to **Commit** or **Approve and Commit** via the Swarm UI and the shelved files are out of date.

Helix Swarm cannot currently help with resolving conflicts; you need to use a Helix Core client such as p4 or P4V to resolve conflicts.

Resolve via P4

1. Acquire a local copy of the files.
2. Sync the files to the head version:

```
$ p4 sync
```

3. Begin resolving files with:

```
$ p4 resolve
```

4. Choose an appropriate option to resolve each file. For example:

```
$ p4 resolve
/home/bruno/bruno_ws/dev/main/jam/command.c - merging
//depot/dev/main/jam/command.c#9
Diff chunks: 4 yours + 2 theirs + 1 both + 1 conflicting
Accept(a) Edit(e) Diff(d) Merge (m) Skip(s) Help(?) e:
```

5. Re-shelve the resolved files with:

```
$ p4 shelve
```

Note
Ensure that the changelist description contains `#review-12345` (separated from other text by whitespace, or on a separate line), where `12345` is the identifier of the review you are updating.

Warning
If you use an invalid review identifier, it will appear that nothing happens. Swarm is currently unable to notify you of this situation.

For more information, see "Resolve" in the *Helix Versioning Engine User Guide*.

Resolve via P4V

1. Acquire a local copy of the files.
2. In P4V, right-click your workspace folder and select **Resolve Files**.

   The **Resolve** dialog appears.
3. Choose the appropriate options to resolve each file.

4. Right-click your workspace folder and select **Shelve Files**.

   The **Shelve** dialog appears.

   Note
   Ensure that the changelist description contains `#review-12345` (separated from other text with whitespace, or on a separate line), where `12345` is the identifier of the review you are updating.

   Warning
   If you use an invalid review identifier, it will appear that nothing happens. Swarm is currently unable to notify you of this situation.

5. Click **Shelve**.

For more information, see "Resolving Files" in the *P4V User Guide*.

How do I see a list of Git-created reviews?

You can view all reviews that were initiated in Git. First, you need to fetch any review branches that may exist:

```
$ git fetch --prune origin
```

Then you can list all branches, which includes any review branches, for the current Git Fusion repo:

```
$ git branch -a
task1
* master
remotes/origin/task1
remotes/origin/master
remotes/origin/review/task1/1234
remotes/origin/review/task1/1244
remotes/origin/review/task1/1347
remotes/origin/review/master/1235
remotes/origin/review/master/1236
remotes/origin/review/master/1358
```

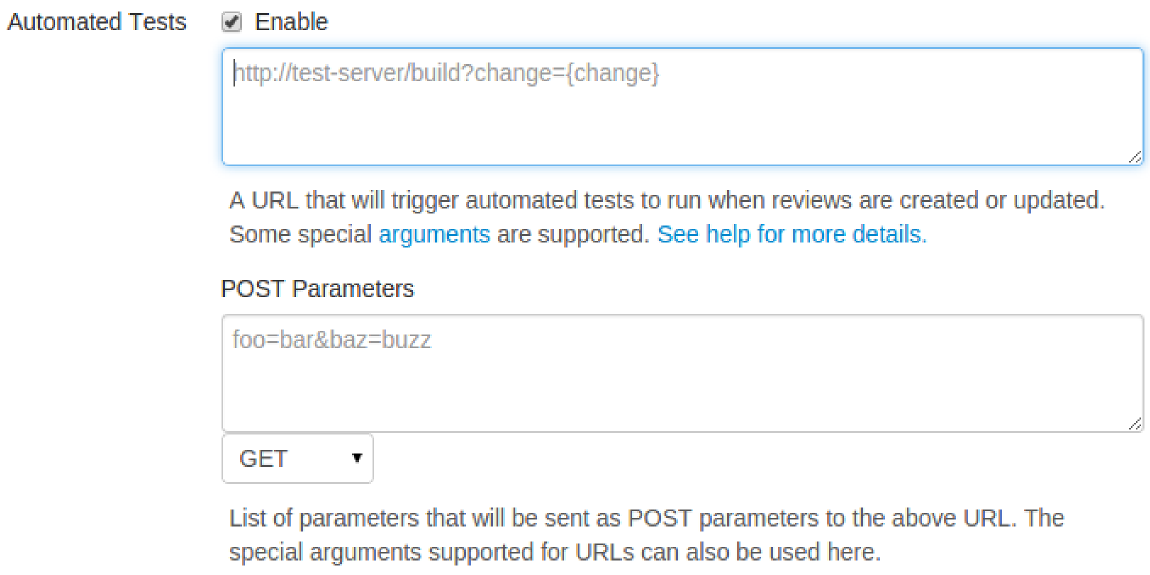
Note
Git users cannot see Swarm reviews initiated in Helix Core.

For more information on Git Fusion, see the *Git Fusion Guide*.

How can I integrate my test suite to inform review acceptance or rejection?

Integrating Helix Swarm with a test suite involves enabling **Automated Tests** in your project's configuration and providing a t*rigger URL*. When the *trigger URL* is requested, Swarm expects your test suite to be executed. When the tests complete, Swarm expects either a *pass callback URL* or *fail callback URL* to be requested by your test suite.

1. Visit your project page.
2. Click **Edit**. The Edit Project page is displayed.
3. Ensure that paths in each named branch configured for the project do not overlap with paths in other named branches.
4. Select the **Enable** check box next to Automated Tests to display the Automated Tests configuration fields:


Automated Tests ✓ Enable

http://test-server/build?change={change}

A URL that will trigger automated tests to run when reviews are created or updated. Some special arguments are supported. See help for more details.

POST Parameters

foo=bar&baz=buzz

GET

List of parameters that will be sent as POST parameters to the above URL. The special arguments supported for URLs can also be used here.

5. Provide a URL that triggers your test suite execution.

   Special arguments are available to inform your test suite of various details from Swarm:

   `{change}`
   : The change number

   `{status}`
   : Status of the shelved change, *shelved* or *committed*

   `{review}`
   : The review's identifier

   `{project}`
   : The project's identifier

   `{projectName}`
   : The project's name

   `{branch}`
   : The branch identifier(s) impacted by the review, comma-separated

   `{branchName}`
   : The branch name(s) impacted by the review, comma-separated

   `{pass}`
   : Tests pass callback URL

   `{fail}`
   : Tests fail callback URL

   > **Note**
   > The `{pass}` and `{fail}` are composed automatically by Swarm, and include Swarm's own per-review authentication tokens.

6. Optionally, specify any parameters that your automated tests require that must be sent via HTTP POST in the **POST Parameters** field. The POST parameters can include the special arguments listed above.

   You can also choose the format of the POST parameters, either GET or JSON. When GET is selected, the POST parameters are parsed into name=value pairs. When JSON is selected, any specified parameters are passed raw in the POST body.

Configuring Jenkins for Swarm integration

> **Important**
> Your Jenkins host needs to be able to communicate with the Swarm host, and vice versa. Ensure that the appropriate DNS/host configuration is in place, and that each server can reach the other via HTTP/HTTPS.

1. Install the p4-plugin in Jenkins:

   https://wiki.jenkins-ci.org/display/JENKINS/P4+Plugin

2. Configure a Jenkins project:
   a. Specify the job name so that it matches the project identifier used in the trigger URL, as defined below.

      For example, the computed value of `{projectName}_{branchName}`.

      Or, edit the trigger URL to use the Jenkins job name you specify.
   b. Make the build parameterized to accept these parameters (note that these are named to match up with the script that is called):

      {status}
      Whether the changelist to be tested is shelved or submitted

      {change}
      Changelist # to run tests against

      {review}
      The review's identifier

      {pass}
      The URL to wget if the build succeeds

      {fail}
      The URL to wget if the build fails
   c. Select `Perforce Software` for the **Source Code Management** section.

      Important
      You may see **Perforce** in the **Source Code Management** section. This represents an earlier community-provided Perforce plugin that does not include support for Swarm.
   d. Set up credentials and workspace behavior as needed.

      See the Credentials and Workspaces sections of the p4-plugin documentation for details.

      Important
      The client workspace configured in Jenkins must have a view that includes the paths defined for that branch in Swarm.

3. Configure your Swarm project to run automated tests with a URL like this:

   ```
   http://jenkins_host:8080/job/{projectName}_
   {branchName}/review/build?status={status}&review={review}&change=
   {change}&pass={pass}&fail={fail}
   ```

   Important
   For Jenkins, the job name needs to match the job identifier in the URL. In the example above, this is the computed value of `{projectName}_{branchName}`.

   If you prefer a different naming scheme in Jenkins, replace `{projectName}_{branchName}` in the URL above with the project name actually defined in Jenkins.

   Note
   If your build script has access to the results of test execution, include a GET or POST parameter called url when calling the pass or fail URLs. Swarm uses the provided url to link reviews to the test results.

   Important
   If security is enabled in Jenkins, the trigger URL needs to include credentials. Follow these steps:

   a. Create a Jenkins user that will trigger Swarm builds. For example swarm.
   b. Log into Jenkins as the new user.
   c. Click on the user's username in the Jenkins toolbar.
   d. Scroll down to **API Token**.
   e. Click **Show API Token**.
   f. Incorporate the value of the **API Token** into the Swarm trigger URL.

      For example, if the username is swarm and the API Token value is 832a5db7e5500c1288324c1441460610, the Swarm trigger URL should be:

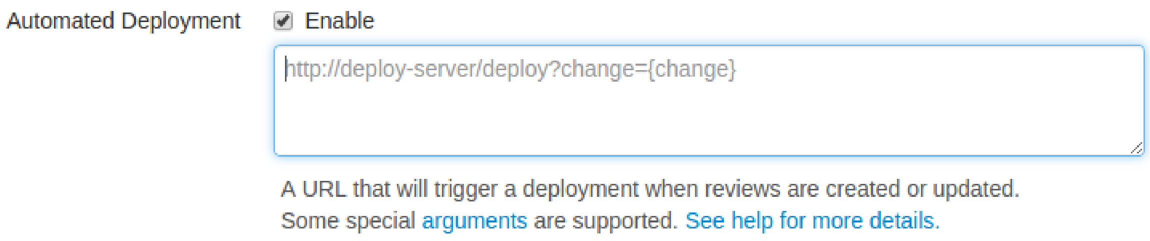
      ```
      http://swarm:832a5db7e5500c1288324c1441460610@jenkins_
      host:8080/job/{projectName}_
      {branchName}/review/build?cause=Automated%20test%20triggered%2
      0for%20Swarm%20project%20{projectName},%20branch%20
      {branchName},%20review%20{review}&status={status}&review=
      {review}&change={change}&pass={pass}&fail={fail}
      ```

How can I automatically deploy code within a review?

Deploying code in a code review automatically involves enabling **Automated Deployment** in your project's configuration and providing a *trigger URL*. When the *trigger URL* is requested, Swarm expects a deployment program to be executed.

When the deployment processing ends, Swarm expects either a *success callback URL* or *failure callback URL* to be requested by your deployment program. These callback URLs should include a url parameter (either via GET or POST); when a valid-looking URL is included, clicking the deployment status indicator directs the user to the specified URL. This is intended to facilitate easy viewing of the successfully deployed review, or a report indicating why the deployment failed. The url parameter is mandatory for successful deployments, but is optional for failures.

1. Visit your project page.
2. Click **Edit**. The **Edit Project** page is displayed.
3. Select the **Enable** check box next to Automated Deployment to display the Automated Deployment configuration fields.



A URL that will trigger a deployment when reviews are created or updated. Some special arguments are supported. See help for more details.

5. Provide a URL that triggers your deployment execution.

   Special arguments are available to inform your deployment program of various details from Swarm:

   `{change}`
   : The change number

   `{status}`
   : Status of the shelved change, shelved or committed

   `{review}`
   : The review's identifier

   `{project}`
   : The project's identifier

   `{projectName}`
   : The project's name

   `{branch}`
   : The branch identifier(s), comma-separated

   `{branchName}`
   : The branch name(s), comma-separated

   `{success}`
   : Deployment successful callback URLd

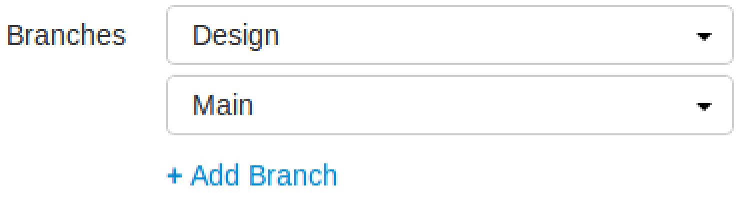
   `{fail}`
   : Deployment failure callback URL

How do I manage project branches?

Initial steps:

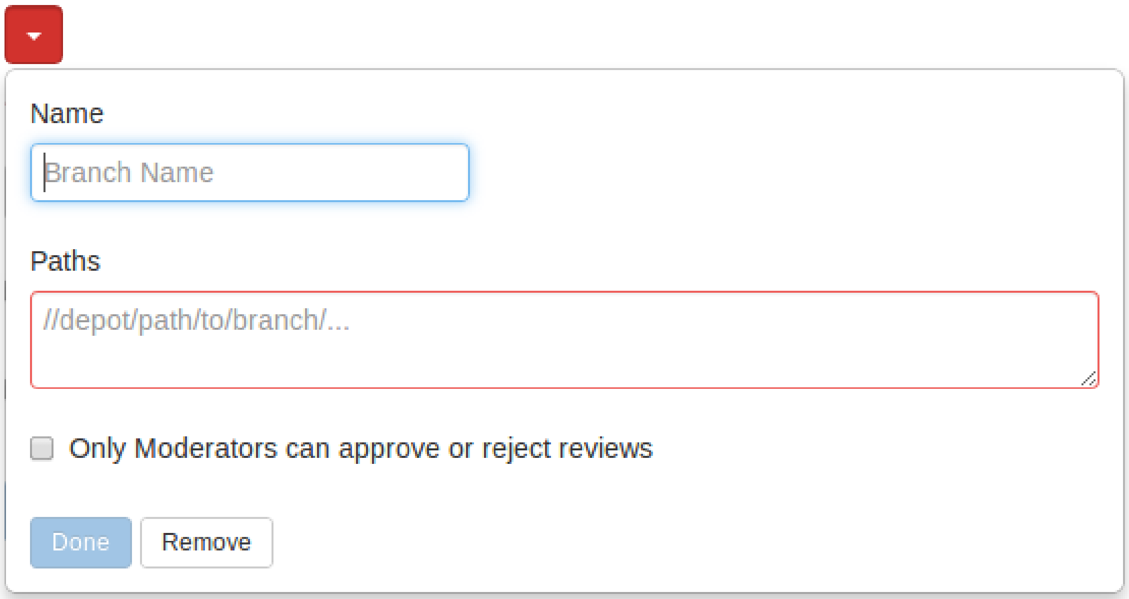
1. Visit your project page.
2. Click **Edit**.

   Next to the **Branches** label, a drop-down button for each branch is displayed, plus an **Add Branch** button.



Adding a branch

1. Follow the initial steps.
2. Click **Add Branch**. The branch drop-down dialog opens.


Name
Branch Name
Paths
//depot/path/to/branch/...
Only Moderators can approve or reject reviews
Done
Remove

3. In the **Name** field, enter a short name for your branch.

4. In the **Paths** field, enter one or more branch paths, one per line.

   > **Note**
   > Each branch path should be expressed in depot syntax. Wildcards should not be used; the only exception is the branch path can end with the Helix Core wildcard ...
   >
   > For example: //depot/main/swarm/...
   >
   > For more information, see File Specifications in *P4 Command Reference*.

5. Optionally, check the **Only Moderators can approve or reject reviews** checkbox.

   When checked, a field is displayed, allowing you to add a new moderator. The field auto-suggests groups and users within the Helix Versioning Engine as you type.

   If a group is specified as a moderator, all of the members of that group have the same moderator privileges for that project branch as if they were added individually.

   Once the branch specification is complete and the project has been saved, changing the state of any review associated with this moderated branch is restricted as follows:

   - Only moderators can approve or reject the review. Moderators can also transition a review to any other state.
   - The review's author, when she is not a moderator, can change the review's state to **Needs Review**, **Needs Revision**, **Archived**, and can attach committed changelists.

     Normally, the review's author cannot change the review's state to **Approved** or **Rejected** on moderated branches. However, authors that are also moderators have moderator privileges, and may approve or reject their own review.

     When `disable_self_approve` is enabled, authors who are moderators (or even users with admin privileges) cannot approve their own reviews.
   - Project members can change the review's state to **Needs Review** or **Needs Revision**, and can attach committed changelists. Project members cannot change the review's state to **Approved**, **Rejected**, or **Archived**.
   - Users that are not project members, moderators, or the review's author cannot transition the review's state.
   - For the review's author and project members, if a review is not in one of their permitted states, for example if the review's state is **Rejected**, they cannot transition the review to another state.

     These restrictions have no effect on who can start a review.

6. Click **Done** to accept your branch specification.

   Once the branch definition has completed, if any moderators were specified, the number of moderators for that branch is displayed in the list of branches.

7. Click **Save** to save the branch changes to your project.

Note
The project name does not need to be included in the branch name; Swarm displays the project name with the branch name when appropriate.

Editing a branch

1. Follow the initial steps.
2. Click the branch drop-down button you want to edit.
3. Revise the **Name**, **Paths**, or moderators as required.
4. Click **Save**.

Removing a branch

1. Follow the initial steps.
2. Click the branch drop-down button you want to remove.
3. Click **Remove**.
4. Click **Save**.

How do I change the logging level?

Swarm logs various activities to the data/log file. Change the logging level to increase or decrease the volume of log data by editing a configuration file.

An example configuration, in the *SWARM_ROOT*/**data/config.php** file:

```
<?php
       // this block should be a peer of 'p4'
       'log' => array(
           'priority'  => 3, // 7 for max, defaults to 3
       ),
```

The maximum value for the log priority is 7; higher values do not result in increased logging. The minimum value is 0, which means no logging; lower values do not result in further logging reductions. For more information, see "Logging" on page 264.

How do I check on the queue workers?

Helix Swarm uses a custom queue system to process events, provide notifications, and more. The queue system is required to handle the potentially large volume of events from a busy Helix server.

Check the status of the queue by making an HTTP request to `/queue/status`. The response is formatted in JSON and looks like this:

```
{"tasks":0,"futureTasks":1,"workers":3,"maxWorkers":3,"workerLifetime":"59
5s"}
```

This response indicates that the queue has no current tasks, there is 1 task scheduled for processing later, there are 3 queue workers available, at most 3 workers are created, and queue workers run for at most 10 minutes before self-terminating.

A queue manager ensures that sufficient queue workers are available to process items. If the queue manager has stopped for some reason, start a new one by making an HTTP request to `/queue/worker`. No response is provided for this request.

Note
A cron job should be setup to ensure that workers are running to process events. See "Set up a recurring task to spawn workers" on page 99.

3 | Setting up

This chapter covers the initial installation and configuration of Swarm. Also covered is upgrading Swarm if you have previously installed Swarm.

First, review the "Runtime dependencies" below section. Note that Swarm's dependencies can mostly easily be installed using "Swarm packages" on page 48.

You have several choices for installing Swarm. Choose one approach:

- Use the Swarm packages (RPM or Debian):
  - Follow the steps provided in "Swarm packages" on page 48.
  - Once Swarm is installed, it needs some initial configuration (common to any installation approach). Skip to the "Helix Core configuration for Swarm" on page 81.
- Use the Swarm OVA (Open Virtualization Appliance):
  - Follow the steps provided in "OVA configuration" on page 65.
  - Once Swarm is installed, it needs some initial configuration (common to any installation approach). Skip to the "Helix Core configuration for Swarm" on page 81.
- Use the Swarm source distribution archive:
  - Follow all of the steps provided in the "Initial manual installation" on page 69 and subsequent sections.

After completing the installation and initial configuration steps, you may want to review the "Post-install configuration options" on page 103 that may be useful for your deployment of Swarm.

If you have not already done so, download Swarm.

Runtime dependencies

In order to successfully install, configure, and deploy Swarm, the following dependencies are required:

- A "Supported operating system platforms" on the next page
- An "Apache web server" on page 42 with `mod_rewrite` and `mod_php5`

- A supported version of "PHP" on the facing page with the following extensions:
  - iconv
  - JSON
  - Session
  - P4PHP
  - one of APC or Zend OPCache (for optimal performance)
  - php-mbstring (for multi-byte character strings)

  In addition, Swarm greatly benefits from the following optional extensions:
  - Imagick (used for viewing non-web safe images)
  - php-xml (for RSS feeds)
- A supported Helix Versioning Engine deployment, and the ability to connect to it from the system hosting Swarm.

  > **Note**
  > "Helix server deployment" can refer to a running **`p4d`** or a proxy, replica, edge server, or commit server.

- **`curl`** or **`wget`** (for Swarm worker operation)
- A supported version of perl (to integrate with Helix Versioning Engine triggers)
- That "Security-enhanced Linux (SELinux)" on page 47, if installed, is configured

Optional dependencies:

- LibreOffice (for viewing office-type documents)
- **`zip`**, the command-line archiving tool (for downloading archives of files/folders

Supported operating system platforms

Because Swarm includes binary versions of P4PHP (the Perforce extension for PHP), we support Swarm on the following operating systems:

- Linux 2.6+ Intel (x86, x86_64) with glibc 2.3.3+
- Mac OS X 10.6+ (x86_64)

You may be able to get Swarm running on another platform if you build P4PHP yourself and satisfy the other runtime dependencies. Instructions on how to obtain and build P4PHP from source can be found here.

> **Important**
> P4PHP does not support threaded operation. If you compile P4PHP from source, ensure that the

version of PHP you compile for is non-threaded.

Apache web server

Swarm requires Apache HTTP Server 2.2 or newer:

- https://httpd.apache.org

Swarm also requires the following Apache modules:

- mod_php5 for interacting with PHP (usually installed with PHP)
- mod_rewrite URL rewriting engine

  https://httpd.apache.org/docs/2.2/mod/mod_rewrite.html

Important
Only the `prefork` MPM is supported. Use of the worker or event MPMs is not supported and is likely to cause problems because P4PHP does not support threaded operation.

For more information on the prefork MPM, see:

- https://httpd.apache.org/docs/2.2/mod/prefork.html
- https://httpd.apache.org/docs/2.4/mod/prefork.html

PHP

Swarm requires PHP 5.3.3+, 5.4.x, 5.5.x, 5.6.x, or 7.0:

- https://secure.php.net

Important
PHP must be non-threaded because P4PHP does not support threaded operation.

Swarm requires the following PHP extensions:

- iconv (character encoding converter)
  https://secure.php.net/iconv

  This is typically enabled by default with most PHP distributions
- JSON (JavaScript Object Notation)
  https://secure.php.net/json

  This is typically enabled by default with most PHP distributions, although recent distributions are making this optional

- Session (session handling)

  This is typically enabled by default with most PHP distributions
- P4PHP (the Perforce PHP Extension)

  Included with the Swarm package. See the install directions.
- php-xml (DOM API for XML manipulation)

  Included with PHP on many operating systems, but must be manually installed on CentOS/RHEL. When not installed, the Swarm RSS feed does not work.
- php-mbstring (multi-byte character strings)

  Included with PHP on many operating systems, but must be manually installed on CentOS/RHEL. When not installed, Swarm's RSS feed does not work.

Swarm greatly benefits from the following PHP extensions:

- One of (but not both):
  - APC (the Alternative PHP Cache)
    https://secure.php.net/apc

    Installation instructions for APC.
  - Zend OPCache
    https://secure.php.net/opcache

    Installation instructions for Zend OPCache.
- Imagick (integrates ImageMagick into PHP)
  https://secure.php.net/imagick

  Installation instructions for Imagick.

Helix server requirements

Swarm works with any supported version of the Helix Versioning Engine. The versions supported in this release of Swarm include the following versions:

- 2015.2
- 2016.1
- 2016.2
- 2017.1
- 2017.2

Swarm performs best with the Helix Versioning Engine version 2016.2 or newer.

- https://www.perforce.com/downloads/helix-versioning-engine-p4d

Swarm requires a user with at least admin privileges in the Helix Versioning Engine. This can be an existing user, or a new user created specifically to support Swarm.

For more information about how to set up a Helix Versioning Engine, see *Helix Versioning Engine Administrator Guide: Fundamentals*.

Trigger dependencies

The Swarm triggers, which are installed on the Helix server in a later step, require perl 5.08+: https://www.perl.org/get.html

On the Windows platform, we have tested Swarm against Strawberry Perl. There are two Perl modules that are also required which may not be part of a minimal Perl installation.

- `HTTP::Tiny` is required to make calls to the Swarm server. If this is not present, then the trigger will attempt to use the command line curl program. This module is standard on Strawberry Perl on Windows, and available as a package with the version of Perl provided on CentOS 7, Ubuntu 14.04 and Ubuntu 16.04.

  `IO::Socket::SSL` is required if the Swarm server is configured to use SSL and HTTP::Tiny is present. This is provided as standard by Strawberry Perl, and available on Linux.

  Warning
  If the `HTTP::Tiny` module is not available, for example on CentOS 6, then the triggers require the use of `curl`. This must be installed for the triggers to function. On CentOS 6 this can be done using the `yum` package installer using `yum install curl` if it isn't already installed.

Worker dependencies

The recurring task to invoke Swarm workers, installed in a later step, requires either of:

- curl
  https://curl.haxx.se/download.html

  Note
  For Windows, `curl.exe` depends on `MSVCR100.dll`. You can get a copy by installing the **Microsoft Visual C++ Redistributable Package**, available for:

  - 32-bit systems: https://www.microsoft.com/download/en/details.aspx?id=5555
  - 64-bit systems: https://www.microsoft.com/download/en/details.aspx?id=14632

  If you install Swarm with HTTPS, `curl.exe` requires recent CA certificates (or HTTPS connections silently fail). You can get a copy of the `cacert.pem` from:

  https://curl.haxx.se/docs/caextract.html

Once downloaded, copy `cacert.pem` to the same folder where you installed `curl.exe`, and rename it to `curl-ca-bundle.crt`.

Warning
If `curl` (or `curl.exe` on Windows) cannot execute as expected, trigger execution may block or fail. For example, if `MSVCR100.dll` is missing from a Windows system, invoking `curl.exe` causes a dialog to appear.

Prior to configuring the triggers, verify that `curl` executes. On Linux systems, run:

```
 $ curl -h
```

On Windows systems, run:

```
C:\> curl.exe -h
```

The start of the output should be similar to:

```
Usage: curl [options...] <url>
Options: (H) means HTTP/HTTPS only, (F) means FTP only
     --anyauth       Pick "any" authentication method (H)
 -a, --append        Append to target file when uploading (F/SFTP)
     --cacert FILE   CA certificate to verify peer against (SSL)
     --capath DIR    CA directory to verify peer against (SSL)
...[truncated for brevity]...
```

For a more thorough test that actually fetches content over a network, try the following test:

- For Linux systems, run:

  ```
  $ curl https://www.perforce.com/
  ```

- For Windows systems, run:

  ```
   C:\> curl.exe https://www.perforce.com/
  ```

The output should look like HTML.

- wget
  https://ftp.gnu.org/gnu/wget
  http://gnuwin32.sourceforge.net/packages/wget.htm (for Windows)

Note
If you are using Powershell on Windows systems, be aware that Powershell includes aliases for **`curl`** and **`wget`** that call the Powershell command **`Invoke-WebRequest`** instead of **`curl.exe`** or **`wget.exe`**. **`Invoke-WebRequest`** has different command-line options than either **`curl`** or **`wget`**, which can be confusing.

If you want to remove the built-in aliases for **`curl`** and **`wget`** from Powershell, follow these steps:

1. Create a Powershell profile (only if you have not already done so):

   ```
   PS C:\> New-Item $profile -force -itemtype file
   ```

2. Edit your profile:

   ```
   PS C:\> notepad $profile
   ```

3. Add the following line to your profile:

   ```
   remove-item alias:curl
   remove-item alias:wget
   ```

4. Save the profile and close **`notepad`**.
5. Reload your profile:

   ```
   PS C:\> . $profile
   ```

Supported web browsers

The following browsers are supported for use with Swarm:

- Apple Safari, latest stable version
- Google Chrome, latest stable version
- Microsoft Internet Explorer 10+
- Mozilla Firefox, latest stable version

Other web browsers might also work, including prior, development or beta builds of the above web browsers, but are not officially supported.

Swarm requires that JavaScript and cookies are enabled in the web browser.

Optional dependencies

Swarm can display previews of office-type documents when LibreOffice is installed. Installation is not required, but when LibreOffice is installed Swarm automatically detects its presence.

For more information, see "LibreOffice" on page 237.

To use the **Download .zip** feature (see "Downloading files as ZIP archive" on page 125), Swarm requires that the `zip` command-line tool be available.

For more information, see "Archives configuration" on page 240.

Security-enhanced Linux (SELinux)

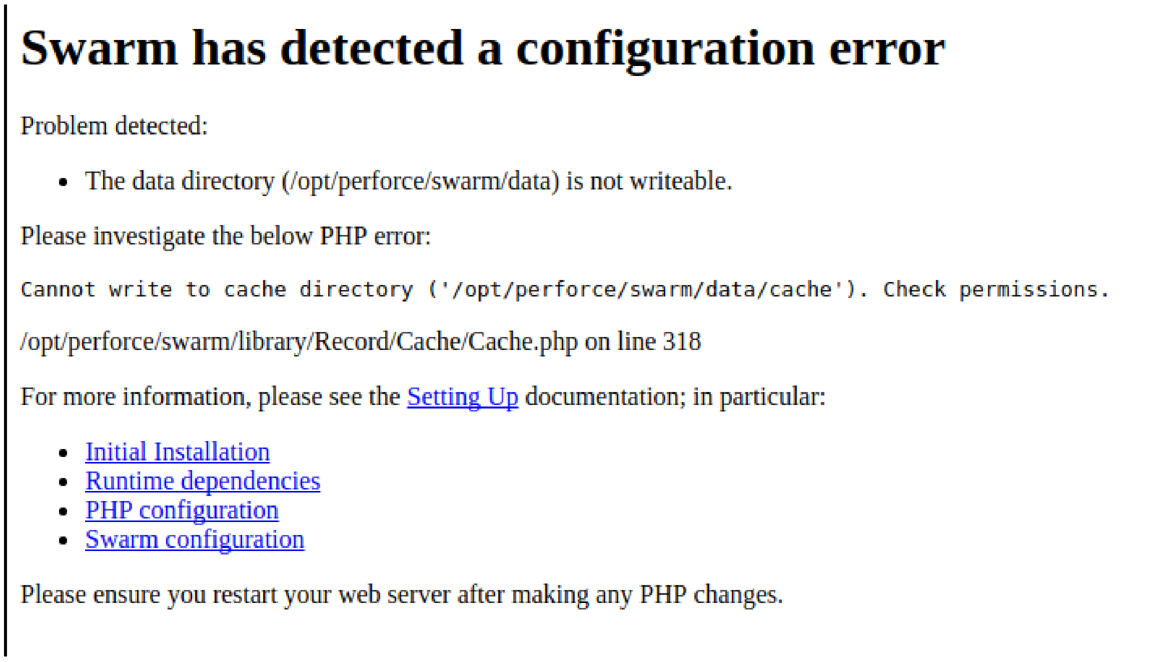
Swarm supports SELinux on CentOS 7. SELinux is an advanced access control mechanism that improves security for Linux distributions.

SELinux operates in one of three modes:

- `enforcing`: this mode blocks and logs any actions that do not match the defined security policy. This is the default mode for SELinux on CentOS 7.
- `permissive`: this mode logs actions that do not match the defined security policy but these actions are not blocked.
- `disabled`: in this mode SELinux is off, actions are not blocked and are not logged.

See "SELinux on CentOS 7 configuration" on page 63 for details.

If you see a Swarm configuration error similar to the error shown below, SELinux has not been correctly configured for Swarm. Check you have configured SELinux on CentOS 7 correctly.



Swarm has detected a configuration error

Problem detected:

- The data directory (/opt/perforce/swarm/data) is not writeable.

Please investigate the below PHP error:

```
Cannot write to cache directory ('/opt/perforce/swarm/data/cache'). Check permissions.
```

/opt/perforce/swarm/library/Record/Cache/Cache.php on line 318

For more information, please see the Setting Up documentation; in particular:

- Initial Installation
- Runtime dependencies
- PHP configuration
- Swarm configuration

Please ensure you restart your web server after making any PHP changes.

Choose installation approach

Once you are aware of Swarm dependencies and can satisfy them, you have the following choices for installing Swarm. Choose one approach:

- Use the Swarm packages (RPM or Debian):
  1. Follow the steps provided in "Swarm packages" below.
  2. Configure Swarm. Skip to the "Helix Core configuration for Swarm" on page 81.
- Use the Swarm OVA (Open Virtualization Appliance):
  1. Follow the steps provided in "OVA configuration" on page 65.
  2. Configure Swarm. Skip to the "Helix Core configuration for Swarm" on page 81.
- Use the Swarm source distribution archive:
  - Follow steps provided in the "Initial manual installation" on page 69 and subsequent sections.

If you have not already done so, download Swarm.

Swarm packages

Helix Swarm is available in two distribution package formats: Debian (`.deb`) for Ubuntu systems and RPM (`.rpm`) for CentOS and RedHat Enterprise Linux (RHEL).

Using distribution packages greatly simplifies the installation, updating, and removal of software, as the tools that manage these packages are aware of the dependencies for each package.

Note
The Swarm packages have been thoroughly tested on Ubuntu 14.04 LTS and Ubuntu 16.04 LTS, and CentOS/RHEL 6.1-6.7, and CentOS/RHEL 7. While the packages should work on other compatible distributions, these have not been tested.

Note
Helix Versioning Engine can refer to a Helix server machine (`p4d`), proxy, broker, replica, edge server, or commit server. It does not refer to a service user; service users are used to coordinate replication in a Helix Versioning Engine. For simplicity, the term *Helix server* is used to refer to any configuration of a Helix Versioning Engine machine.

Installation

1. Configure the Perforce package repository, on the server to host Swarm and on the server hosting your Helix Versioning Engine.

   Important
   If the server hosting your Helix Versioning Engine cannot use packages, for example when it is running Windows, skip this step on that server.

   As root, run one of the following:

   - **For Ubuntu 14.04:**

     Create the file `/etc/apt/sources.list.d/perforce.list` with the following content:

     ```
     deb http://package.perforce.com/apt/ubuntu/ trusty release
     ```

   - **For Ubuntu 16.04:**

     Create the file `/etc/apt/sources.list.d/perforce.list` with the following content:

     ```
     deb http://package.perforce.com/apt/ubuntu/ xenial release
     ```

   - **For CentOS/RHEL 6:**

     Create the file `/etc/yum.repos.d/helix-swarm.repo` with the following content:

     ```
     [Perforce]
     name=Perforce
     baseurl=http://package.perforce.com/yum/rhel/6/x86_64/
     enabled=1
     gpgcheck=1
     ```

   - **For CentOS/RHEL 7:**

     Create the file `/etc/yum.repos.d/helix-swarm.repo` with the following content:

     ```
     [Perforce]
     name=Perforce
     baseurl=http://package.perforce.com/yum/rhel/7/x86_64/
     enabled=1
     gpgcheck=1
     ```

2. Import the Perforce package signing key, on the server to host Swarm and the server hosting your Helix Versioning Engine.

   Important
   If the server hosting your Helix server cannot use packages, for example when it is running Windows, skip this step on that server.

   Run one of the following:

   - **For Ubuntu:**

     ```
     $ wget -qO - https://package.perforce.com/perforce.pubkey |
     sudo apt-key add -
     $ sudo apt-get update
     ```

   - **For CentOS/RHEL (run this command as root):**

     ```
     # rpm --import https://package.perforce.com/perforce.pubkey
     ```

   For information about how to verify the authenticity of the signing key, see:
   https://www.perforce.com/perforce-packages

3. Install the main Swarm package on the server to host Swarm.

   Run one of the following:

   - **For Ubuntu:**

     ```
     $ sudo apt-get install helix-swarm
     ```

   - **For CentOS/RHEL (run this command as root):**

     ```
     # yum install helix-swarm
     ```

     Note
     For CentOS/RHEL, the firewall configuration may need to be adjusted to allow access to the web server.

     - **For CentOS/RHEL 6.x:**

       ```
       $ sudo lokkit -s http
       ```

       If you subsequently wish to enable HTTPS, run (as root):

       ```
       $ sudo lokkit -s https
       ```

     - **For CentOS/RHEL 7.x:**

       ```
       $ sudo firewall-cmd --permanent --add-service=http
       $ sudo systemctl reload firewalld
       ```

       If you subsequently wish to enable HTTPS, run (as root):

       ```
       $ sudo firewall-cmd --permanent --add-service=https
       $ sudo systemctl reload firewalld
       ```

4. Install the Swarm triggers package on the server hosting your Helix Versioning Engine.

   Install this package on the server hosting your Helix Versioning Engine, which may be the same server that is hosting Swarm, or elsewhere on your network.

   > **Important**
   > If the server hosting your Helix server cannot use packages, for example when it is running Windows, you need to copy the appropriate Swarm trigger script from `/opt/perforce/swarm/p4-bin/scripts` to the server hosting your Helix server. The `swarm-trigger.pl` is for both Linux and Windows systems. Once copied, the trigger script needs to be configured. See "Helix Core configuration for Swarm" on page 81 for details.

   Run one of the following:

   - **For Ubuntu:**

     ```
     $ sudo apt-get install helix-swarm-triggers
     ```

   - **For CentOS/RHEL (run this command as root):**

     ```
     $ yum install helix-swarm-triggers
     ```

   > **Important**
   > The package installs a config file at `/opt/perforce/etc/swarm-trigger.conf` that you will need to modify. See "Helix Core configuration for Swarm" on page 81 for more details on configuring that file.

5. Optional: Install the Swarm optional package, on the server hosting Swarm.

   While not required, installing this package installs the dependencies required to use the Imagick and LibreOffice Swarm modules. These modules provide previews of a variety of image and office documents.

   Run one of the following:

   - **For Ubuntu:**

     ```
     $ sudo apt-get install helix-swarm-optional
     ```

   - **For CentOS/RHEL (run this command as root):**

     ```
     # yum install helix-swarm-optional
     ```

     Important
     This package depends on the package `php-pecl-imagick` which is available from the EPEL project. In order to install packages from EPEL, you will need to add the EPEL repository and accept its signing key. Instructions are available at: https://fedoraproject.org/wiki/EPEL

     Note
     Installation of this package also installs APC for CentOS/RHEL 6, or Zend OPCache for CentOS/RHEL 7.

6. Complete the "Post-installation configuration" on page 55 steps.

Updating

Important
For the Swarm 2015.2 release, the packages have been renamed. The following instructions upgrade your Swarm packages to the latest versions.

Run one of the following:

1. **For Ubuntu:**

   ```
   $ sudo apt-get update
   $ sudo apt-get install helix-swarm helix-swarm-triggers helix-swarm-
   optional
   ```

2. **For CentOS/RHEL (run this command as root):**

   ```
   # yum install helix-swarm helix-swarm-triggers helix-swarm-optional
   ```

Swarm generally has several major updates each year, and may occasionally have a patch update between major updates. To determine whether a Swarm update is available, run one of the following:

1. **For Ubuntu:**

   ```
   $ sudo apt-get update
   $ sudo apt-get -s upgrade | grep swarm
   ```

2. **For CentOS/RHEL (run this command as root):**

   ```
   # yum list updates | grep swarm
   ```

Uninstall

1. Remove the Swarm triggers from your Helix server.
2. Remove the Swarm trigger scripts from the server hosting your Helix server.

   Important
   If you manually installed the trigger script, perhaps because the server hosting your Helix server cannot use packages (e.g. Windows), manually remove the script. **`swarm-trigger.pl`** is for Linux systems. **`swarm-trigger.vbs`** is for Windows systems.

   Run one of the following:

   - **For Ubuntu:**

     ```
     $ sudo apt-get remove helix-swarm-triggers
     ```

   - **For CentOS/RHEL (run this command as root):**

     ```
     # yum remove helix-swarm-triggers
     ```

3. Remove the Swarm packages from the server hosting Swarm.

   Run one of the following:

   - **For Ubuntu:**

     ```
     $ sudo apt-get remove helix-swarm
     ```

   - **For CentOS/RHEL (run this command as root):**

     ```
     # yum remove helix-swarm
     ```

4. If you installed the optional package, remove it from the server hosting Swarm.

   Run one of the following:

   - **For Ubuntu:**

     ```
     $ sudo apt-get remove helix-swarm-optional
     ```

   - **For CentOS/RHEL (run this command as root):**

     ```
     # yum remove helix-swarm-optional
     ```

Post-installation configuration

Once the helix-swarm package has been installed, additional configuration is required. Perform the following steps:

1. Use the Swarm configuration script to setup Swarm, on the server hosting Swarm.

   > **Note**
   > The Swarm configuration script can be used in a few different ways. The steps below outline the most straightforward configuration using an interactive install, but you can review the options by running:
   >
   > ```
   > $ sudo /opt/perforce/swarm/sbin/configure-swarm.sh -h
   > ```

   Run an interactive install:

   ```
   $ sudo /opt/perforce/swarm/sbin/configure-swarm.sh
   ```

   The configuration script displays the following summary:

   ```
   ------------------------------------------------------------
   configure-swarm.sh: Thu Aug 25 11:29:49 PDT 2016: commencing
   configuration of Swarm
   Summary of arguments passed:
   Interactive?        [yes]
   Force?              [no]
   P4PORT              [(not specified)]
   Swarm user          [(not specified, will suggest swarm)]
   Swarm password      [(not specified)]
   Email host          [(not specified)]
   Swarm host          [(not specified, will suggest myhost)]
   Swarm port          [80]
   Swarm base URL      [(default (empty))]
   ```

```
Create Swarm user? [no]
Super user         [(not specified)] * not needed
Super password     [(not specified)] * not needed
```

2. Provide information to the configuration script.

   After the summary, the configuration script prompts for the following information:

   a. Specify a value for **`P4PORT`**.

   ```
   No P4PORT specified

   Swarm requires a connection to a Helix Versioning Engine.
   Please supply the P4PORT to connect to.

   Helix Versioning Engine address (P4PORT):
   ```

   Specify the hostname and port for your Helix server. If defined, the value for P4PORT is used as the default. The configuration script verifies that it can connect:

   ```
   -response: [myp4host:1666]

   Checking P4PORT [myp4host:1666]...
   -P4 command line to use: [/opt/perforce/bin/p4 -p myp4host:1666]
   Attempting connection to [myp4host:1666]...
   -connection successful:
     Server address: myp4host:1666
     Server version: P4D/LINUX26X86_64/2016.1/1411799 (2016/07/12)
     Server license: 10000 users (support ends 2017/05/16)
     Server license-ip: 192.168.0.1
   ```

   Important
   If your Helix Versioning Engine is deployed using the commit-edge architecture, ensure that the Swarm port value points to the commit server.

   For more information, see "Commit-edge Architecture" in the *Helix Versioning Engine Administrator Guide: Multi-Site Deployment*.

b. Specify the userid and password of a normal user with admin-level privileges in the Helix Versioning Engine.

```
Checking Swarm user credentials...
No Swarm user specified

Swarm requires a Helix user account with 'admin' rights.
Please provide a username and password for this account.
If this account does not have 'admin' rights, it will
be set for this user.

Helix username for the Swarm user [swarm]:
```

Enter the userid. The default is *swarm*.

Note
If the Helix Core user account is given 'super' rights, then this allows a user to clean up a review created by another user when the review is committed. See "Review cleanup" on page 277.

```
-response: [swarm]

Helix password or login ticket for the Swarm user (typing
hidden):
```

Enter the login ticket, or password, for the userid.

Note
You can obtain a login ticket by running (in another shell):

```
$ p4 -p myp4host:1666 -u userid login -p
```

If the login ticket you provide would expire in less than a year, you will receive a warning.

```
Checking Swarm user credentials...
-checking if user [swarm] exists in [myp4host:1666]...
-user exists
Obtaining Helix login ticket for [swarm] in [myp4host:1666]...
-login ticket obtained
Checking user [swarm]'s ticket against [myp4host:1666]...
-login ticket is good
```

```
Checking user [swarm] has at least access level [admin]...
-user has maximum access level [admin]
-user meets minimum access level [admin]
```

c. Specify the hostname for the Swarm UI.

```
Swarm needs a distinct hostname that users can enter into their
browsers to
access Swarm. Ideally, this is a fully-qualified domain name,
e.g.
'swarm.company.com', but it can be just a hostname, e.g. 'swarm'.

Whatever hostname you provide should be Swarm-specific and not
shared with
any other web service on this host.

Note that the hostname you specify typically requires
configuration in your
network's DNS service. If you are merely testing Swarm, you can
add a
hostname->IP mapping entry to your computer's hosts
configuration.

Hostname for this Swarm server [myhost]:
```

Note
The default is the current hostname. The configuration script does not verify that the hostname actually works (DNS configuration may not exist yet).

d. Specify a mail relay host.

```
Swarm requires an mail relay host to send email notifications.

Mail relay host (e.g.: mx.yourdomain.com):
```

Note
The configuration script does not verify that the mail relay host you provide actually accepts SMTP connections.

Once this information has been provided, the configuration script performs the following steps (some of the detail depends on the version of PHP and Apache that is installed):

```
Configuring Cron...
`/opt/perforce/etc/swarm-cron-hosts.conf.new' -&gt;
`/opt/perforce/etc/swarm-cron-hosts.conf'
-updated cron configuration file with supplied Swarm host
Configuring Swarm installation...
-composed new Swarm config file contents
`/opt/perforce/swarm/data/config.php.new' -&gt;
`/opt/perforce/swarm/data/config.php'
-wrote new Swarm config file to reflect new configuration
-identified Apache user:group: [www-data:www-data]
-setting permissions on the Swarm data directory...
-ensured file permissions are set properly
Configuring Apache...
-identified Swarm virtual host config file: [/etc/apache2/sites-
available/perforce-swarm-site.conf]
-identified Apache log directory: [/var/log/apache2]
-updated the vhost file to set Apache log directory
-updated the vhost file to reflect Swarm host
-checking Apache modules...
Enabling module rewrite.
Module php5 already enabled
To activate the new configuration, you need to run:
  service apache2 restart
```

```
-proper Apache modules are enabled
-enabling Swarm Apache site...
Enabling site perforce-swarm-site.conf.
To activate the new configuration, you need to run:
  service apache2 reload
-Swarm Apache site enabled
-restarting Apache...
-Apache restarted
configure-swarm.sh: Thu Aug 25 11:31:36 PDT 2016: completed
configuration of Helix Swarm


::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::
::
::  Swarm is now configured and available at:
::
::      http://myhost/
::
::  You may login as the Swarm user [swarm] using the password
::  you specified.
::
::  Please ensure you install the following package on the server
::  hosting your Helix Versioning Engine.
::
::      helix-swarm-triggers
::
::  (If your Helix Versioning Engine is hosted on an OS and
::  platform that is not compatible with the above package, you
can
::  also install the trigger script manually.)
::
::  You will need to configure the triggers, as covered in the
Swarm
```

```
::  documentation:
::
::
http://www.perforce.com/perforce/doc.current/manuals/swarm/setup.
perforce.html
::
::  Documentation for optional post-install configuration, such
as
::  configuring Swarm to use HTTPS, operate in a sub-folder, or
on a
::  custom port, is available:
::
::
https://www.perforce.com/perforce/doc.current/manuals/swarm/setup
.post.html
::

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::
```

Note
If you have installed Swarm on a host that does not provide other web services, you may wish to disable Apache's default site configuration. Doing so means that regardless of the hostname a user might use to reach the web server hosting Swarm, Swarm would be presented.

Be aware that disabling Apache's default site configuration could disable existing web services or content.

Disabling Apache's default site configuration on Ubuntu hosts is easy. Run:

```
$ sudo a2dissite 000-default
```

For CentOS hosts, or for non-standard Apache installations, you would need to manually adjust the Apache configuration. Such changes require familiarity with Apache configuration; for more details, see:
https://httpd.apache.org/docs/current/configuring.html

3. Configure the Swarm triggers on the server hosting your Helix Versioning Engine.

   As the script's output notes, the triggers required for Swarm need to be configured if you have not already done so. See "Helix Core configuration for Swarm" on page 81 for details.

4. Review the "Post-install configuration options" on page 103 for additional configuration alternatives for your Swarm installation.

SELinux on CentOS 7 configuration

Swarm supports SELinux on CentOS 7. SELinux is an advanced access control mechanism that improves security for Linux distributions.

SELinux operates in one of three modes:

- `enforcing`: this mode blocks and logs any actions that do not match the defined security policy. This is the default mode for SELinux on CentOS 7.
- `permissive`: this mode logs actions that do not match the defined security policy but these actions are not blocked.
- `disabled`: in this mode SELinux is off, actions are not blocked and are not logged.

SELinux must be configured to enable it to work correctly with Swarm, these configuration steps are shown below.

Note
You must complete the Helix Swarm package "Installation" on page 49 steps, and the "Post-installation configuration" on page 55 steps before configuring SELinux.

Configure SELinux on CentOS 7 to enforcing mode

Run the following commands as root:

1. Install the `policycoreutils-python` package, this contains **semange** which is used to configure SELinux:

   ```
   root $ yum install policycoreutils-python
   ```

2. Check the current SELinux mode:

   ```
   root $ getenforce
   ```

3. SELinux will report its mode as; `enforcing`, `permissive`, or `disabled`.
   a. If the mode is not set correctly edit the `/etc/selinux/config` file with vi or a similar editor.

      ```
      root $ vi /etc/selinux/config
      ```

   b. Edit the config file so that `SELinux=` is set to `enforcing` .
   c. Save the config file.
   d. Reboot the server to complete the SELinux mode change.

4. Define the context of the `/opt/perforce/swarm` directory and the files in it to `httpd_sys_rw_content_t`:

   ```
   root $ semanage fcontext -a -t httpd_sys_rw_content_t
   "/opt/perforce/swarm(/.*)?"
   root $ restorecon -R /opt/perforce/swarm
   ```

5. Set the SELinux Boolean value to `httpd_can_network_connect 1` to allow Swarm to connect to p4d and other services:

   ```
   root $ setsebool -P httpd_can_network_connect 1
   ```

6. Define the context of the `/opt/perforce/swarm/p4-bin` directory and the files in it to `httpd_sys_script_exec_t`

   ```
   root $ semanage fcontext -a -t httpd_sys_script_exec_t
   '/opt/perforce/swarm/p4-bin(/.*)?'
   root $ restorecon -R -v /opt/perforce/swarm/p4-bin
   ```

7. Restart the system:

   ```
   root $ systemctl restart httpd
   ```

8. Check that you can log in to Swarm.
9. Reboot the server.
10. Check that you can log in to Swarm.

SELinux is now configured for Swarm.

Note
If you can not log in to Swarm it is possible that SELinux is blocking Swarm because its configuration is incorrect. You will need to troubleshoot the SELinux configuration to find any issues.

Install the `setroubleshoot` package, this contains `sealert` which is used when troubleshooting SELinux:

```
root $ yum install setroubleshoot
```

`sealert` helps you to interpret the contents of the `audit.log`. Run the following command:

```
root $ sealert -a /var/log/audit/audit.log
```

Error message: If you see an error message with a title similar to the message below, it may be because you are running CentOS 7 on a Virtual Machine (VM).

```
root $ SELinux is preventing /usr/sbin/ldconfig from write access on the
directory etc.
```

Install `open-vm-tools` on the VM and reboot the VM.

```
root $ yum install open-vm-tools
```

Configure SELinux on CentOS 7 to permissive or disabled mode

Run the following as root:

1. Check the current SELinux mode:

   ```
   root $ getenforce
   ```

2. SELinux will report its mode as; `enforcing`, `permissive`, or `disabled`.
   a. If the mode is not set correctly edit the `/etc/selinux/config` file with vi or a similar editor.

      ```
      root $ vi /etc/selinux/config
      ```

   b. Edit the config file so that `SELinux=` is set to `permissive` or `disabled` as required.
   c. Save the config file.
   d. Reboot the sever to complete the SELinux mode change.
3. Check that you can log in to Swarm.

OVA configuration

Swarm is available as an OVA, an open virtualization appliance that requires minimal configuration.

Note
Helix Versioning Engine can refer to a Helix server machine (`p4d`), proxy, broker, replica, edge server, or commit server. It does not refer to a service user; service users are used to coordinate replication in a Helix Versioning Engine. For simplicity, the term *Helix server* is used to refer to any configuration of a Helix Versioning Engine machine.

Use the OVA if you want to:

- Simplify the installation and configuration steps
- Experiment with Swarm without using additional hardware
- Install Swarm without having a Linux-based server available

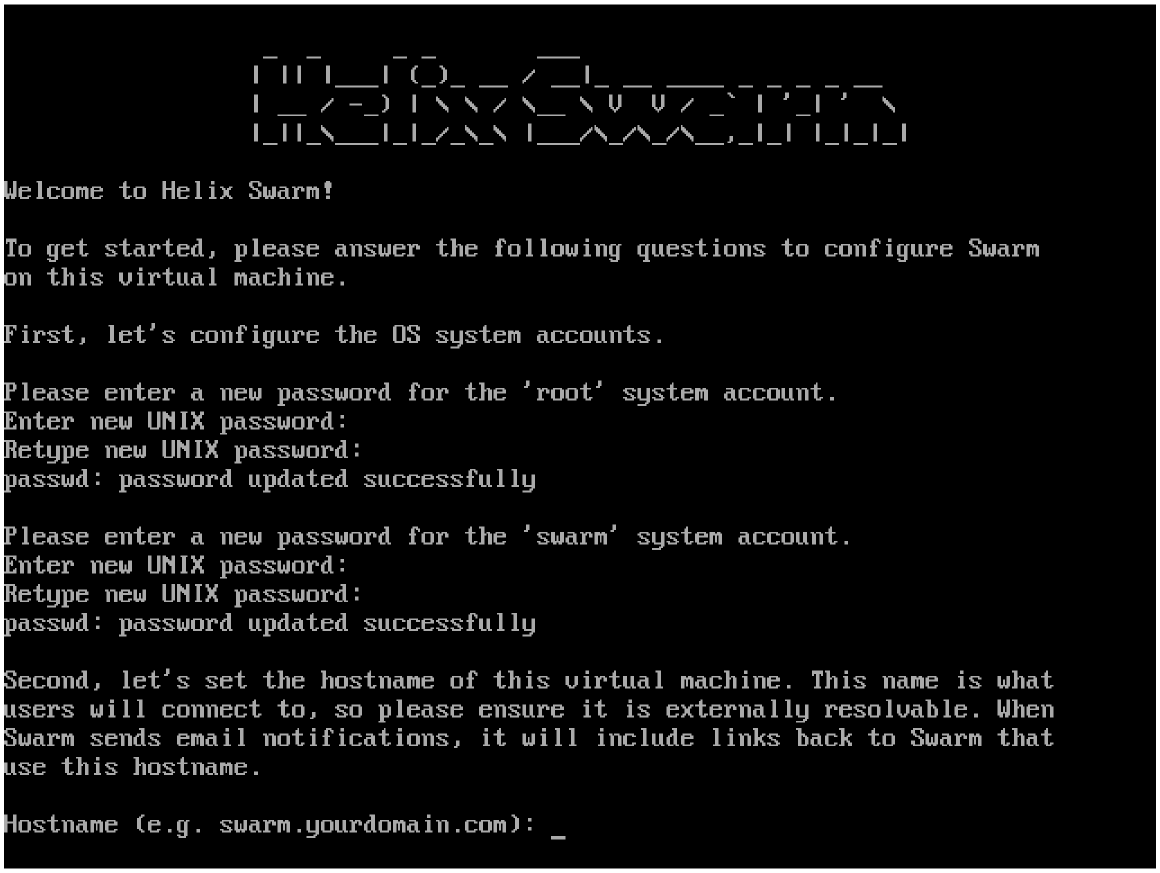
To use the OVA, follow the instructions on this page and skip to the "Establish trigger token" on page 81 section.

1. Download the Swarm OVA.
2. Import the OVA into your virtualization environment. See "VMWare OVA import" on page 69 or

"Oracle VirtualBox import" on page 69.

3. Start the virtual machine; diagnostic and boot information appears.

4. Several configuration prompts appear in sequence:



```
                  _   _      _ _        ____
                 | | | | ___| (_)_  __ / ___|_      ____ _ _ __ _ __ ___
                 | |_| |/ _ \ | \ \/ / \___ \ \ /\ / / _` | '__| '_ ` _ \
                 |  _  |  __/ | |>  <   ___) \ V  V / (_| | |  | | | | | |
                 |_| |_|\___|_|_/_/\_\ |____/ \_/\_/ \__,_|_|  |_| |_| |_|

Welcome to Helix Swarm!

To get started, please answer the following questions to configure Swarm
on this virtual machine.

First, let's configure the OS system accounts.

Please enter a new password for the 'root' system account.
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully

Please enter a new password for the 'swarm' system account.
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully

Second, let's set the hostname of this virtual machine. This name is what
users will connect to, so please ensure it is externally resolvable. When
Swarm sends email notifications, it will include links back to Swarm that
use this hostname.

Hostname (e.g. swarm.yourdomain.com): _
```

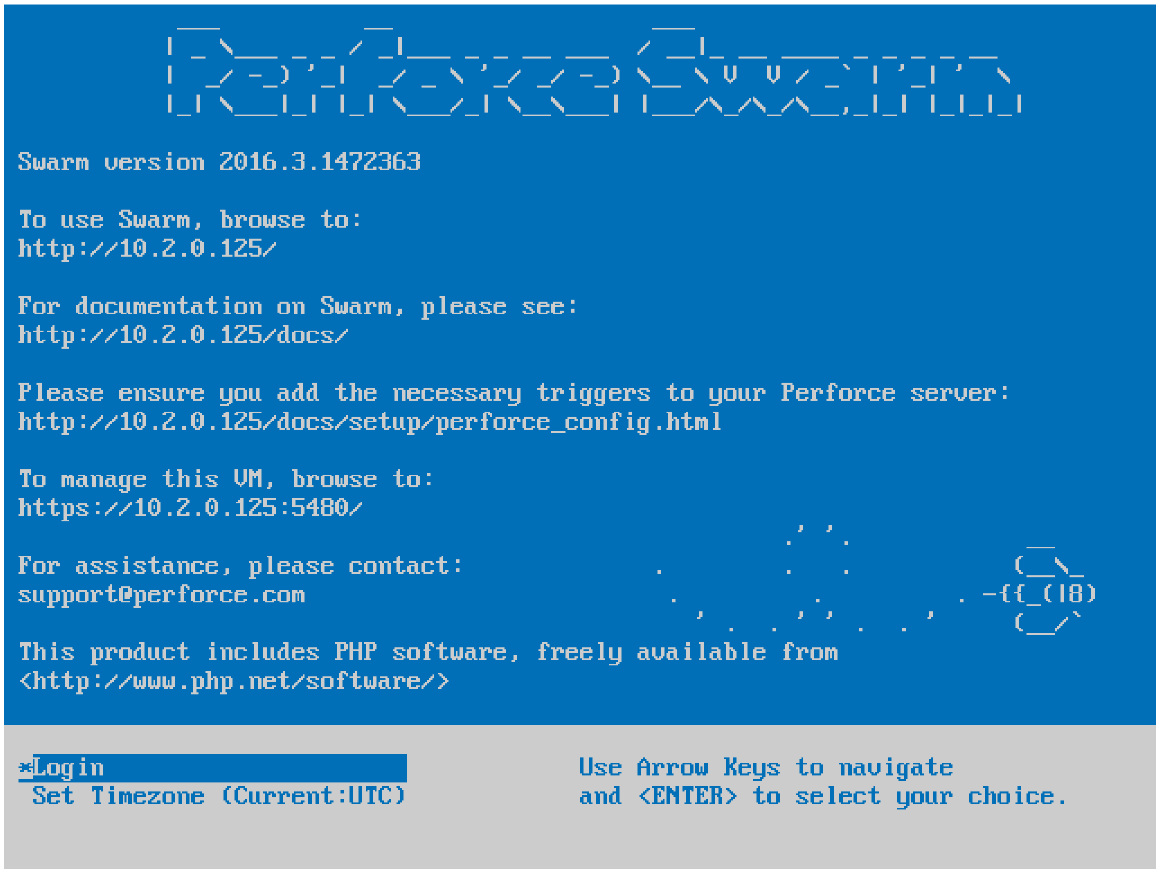
   a. Password for the root user
   b. Password for the system swarm user
   c. Hostname for the virtual machine
   d. Helix Versioning Engine port (P4PORT)

   Important
   If your Helix Versioning Engine is deployed using the commit-edge architecture, ensure that the Swarm port value points to the commit server.

   For more information, see "Commit-edge Architecture" in the *Helix Versioning Engine Administrator Guide: Multi-Site Deployment*.

   e. Userid of a normal user in the Helix server with *admin* privileges
   f. Ticket, or password, of the *admin-level* Perforce user
   g. Mail relay host

Once the prompts have been answered successfully, the virtual machine completes its configuration activities. When ready, a welcome screen is displayed:



```
Swarm version 2016.3.1472363

To use Swarm, browse to:
http://10.2.0.125/

For documentation on Swarm, please see:
http://10.2.0.125/docs/

Please ensure you add the necessary triggers to your Perforce server:
http://10.2.0.125/docs/setup/perforce_config.html

To manage this VM, browse to:
https://10.2.0.125:5480/

For assistance, please contact:
support@perforce.com

This product includes PHP software, freely available from
<http://www.php.net/software/>

*Login                                  Use Arrow Keys to navigate
 Set Timezone (Current:UTC)             and <ENTER> to select your choice.
```

The welcome screen provides URLs to access Swarm, its documentation, and the virtual machine management console.

5. Update the OVA with security updates and bug fixes:

   a. Use `ssh` to log into the OVA as the *root* user.

   b. Enter the following commands to update the OVA's list of packages and to apply any available upgrades.

   ```
   $ apt-get update
   $ apt-get upgrade
   ```

   See OVA Management for more details.

Note
After the OVA is configured and running, you can adjust the configuration by using `ssh` to connect to the virtual machine as the system *swarm* user and editing the "Swarm configuration" on page 78 file `/opt/perforce/swarm/data/config.php`. The Swarm installation folder is

`/opt/perforce/swarm/`.

The OVA setup is now complete. Continue with the steps listed in Helix Core configuration for Swarm to complete the installation of Swarm.

VMWare OVA import

The Swarm OVA works with several VMWare virtualization products, such as Player, Workstation, or Fusion.

1. In the VMWare product, select **File > Open**.
2. Browse to the **`swarm.ova`** file and click **Open**.
3. Type a name for the virtual machine, such as *Swarm*, and click **Import**.

Oracle VirtualBox import

The Swarm OVA works with Oracle VirtualBox, version 4.x+.

1. In VirtualBox, select **File > Import Appliance**.
2. Click **Open Appliance**.
3. Browse to the **`swarm.ova`** file and click **Open**.
4. Click **Continue** (might be **Next >** for some versions of VirtualBox)
5. Click **Import**.

Initial manual installation

1. Expand the Swarm package (a *compressed tarball*).

   Many graphical file manager applications (Nautilus on Linux, Finder on Mac, etc.) can automatically expand the tarball package by simply double-clicking it.

   From the command line, expand it via the tar command:

   ```
   $ tar -zxf swarm.tgz
   ```

   The contents of the Swarm package are expanded into a top-level folder named **`swarm-`*`version`***, where ***`version`*** corresponds to the version downloaded.
2. Move the contents of the Swarm package to the correct location.

   Identify a location for the Swarm files; this should correspond to a location associated to the virtual host configured under Apache (see "Apache configuration" on the facing page).

   ```
   $ mv /path/to/swarm-version /path/to/vhosts/swarm
   ```

3. Assign correct ownership and permission for the Swarm files.

   The data top-level folder in the Swarm distribution needs to be writeable by the web server. To achieve this effect, simply change ownership of the data folder to the web user:

   ```
   $ sudo chown -R www /path/to/vhosts/swarm/data
   ```

   The **www** user above is an example of what the web server user name might be. Depending on your distribution, this could be **_www**, **web**, **nobody** or something else entirely.

   If your web server is already running, you can discover the user with:

   ```
   $ ps aux | grep -E 'apache|httpd'
   root      3592  0.0  0.5 405240 20708 ?        Ss   May03
   4:32 /usr/sbin/apache2 -k start
   www      20016  0.0  0.2 405264  9796 ?        S    07:45
   0:00 /usr/sbin/apache2 -k start
   ```

   In this example, **www** is the user Apache is running as.

   From a security perspective, we recommend that the minimum file permissions should be granted to the user/group under which the web server runs against the Swarm distribution.

Apache configuration

The configuration of the Apache HTTP Server (Apache) can vary between OS distributions; see the documentation specific to your installation of Apache. For example, on Mac OS X, you may have to enable Web Sharing within the Sharing control panel in System Preferences.

1. Locate your system's Apache configuration.

   Common configuration directories include:

   - `/etc/httpd/conf/`
   - `/etc/apache2/`
   - `/Applications/XAMPP/etc/`

   Within the configuration path, the main Apache configuration file is usually named one of the following:

   - `httpd.conf`
   - `apache2.conf`

   Note
   A longer discussion on the possible locations and names of Apache configuration files is available here:

   https://wiki.apache.org/httpd/DistrosDefaultLayout

2. Set up an Apache virtual host (vhost) for your installation.

   If your Apache configuration directory contains the directories `sites-available` and `sites-enabled`:

   a. Copy the appropriate virtual host definition below into the file `sites-available/swarm`.

   b. Enable the Swarm virtual host definition.

   ```
   $ sudo a2ensite swarm
   ```

   Otherwise, copy the appropriate virtual host definition below into the bottom of the main Apache configuration file, `httpd.conf` or `apache2.conf`.

   - Virtual host definition example for Apache 2.2:

   ```
   <VirtualHost *:80>
       ServerName myswarm.host
       ServerAlias myswarm
       ErrorLog "/path/to/apache/logs/myswarm.error_log"
       CustomLog "/path/to/apache/logs/myswarm.access_log"
   common
       DocumentRoot "/path/to/swarm/public"
       <Directory "/path/to/swarm/public">
           AllowOverride All
           Order allow,deny
           Allow from all
       </Directory>
   </VirtualHost>
   ```

- Virtual host definition example for Apache 2.4:

```
<VirtualHost *:80>
    ServerName myswarm.host
    ServerAlias myswarm
    ErrorLog "/path/to/apache/logs/myswarm.error_log"
    CustomLog "/path/to/apache/logs/myswarm.access_log" common
    DocumentRoot "/path/to/swarm/public"
    <Directory "/path/to/swarm/public">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Note
See Apache's virtual host documentation for details:

https://httpd.apache.org/docs/2.2/vhosts/

https://httpd.apache.org/docs/2.4/vhosts/

3. Customize the virtual host definition.
   a. Replace ***myswarm.host*** with the hostname for Swarm on your network. This may require adjusting the DNS configuration on your network.
   b. Replace ***myswarm*** with the name of the subdomain hosting Swarm. Many administrators choose swarm.

      Note the string ***myswarm*** in the log file paths: this should match the subdomain name and prefix for the log files, to help coordinate the active host with the log files for that host. Doing this is particularly useful when your Apache server hosts multiple instances of Swarm.
   c. Replace ***/path/to/apache/logs*** with the path where your Apache stores its log files. Apache's log files are typically named **access_log** and **error_log**.
   d. Replace ***/path/to/swarm*** with the path to the Swarm directory.

4. Verify that the correct Apache modules are enabled.
   - To query whether the PHP and Rewrite modules are active, use the apachectl utility to list all of the active modules (this may be named `apache2ctl` on your system):

     ```
     $ apachectl -t -D DUMP_MODULES
     ```

   - Simply look for `php5_module` and `rewrite_module` in the output. If you see them, skip ahead to step 5.
   - If the Apache utility `a2enmod` is installed, use it to enable the PHP and Rewrite modules:

     ```
     $ sudo a2enmod php5 rewrite
     ```

   - Without the `a2enmod` utility, edit the Apache configuration file by hand. Locate your Apache configuration file for modules and either uncomment or add the following lines:

     ```
     LoadModule  php5_module     libexec/apache2/libphp5.so
     LoadModule  rewrite_module  libexec/apache2/mod_rewrite.so
     ```

   - Note that your Apache installation may have different paths for the location of its modules (the .so files).
5. Restart your web server.
   - To ensure that the Apache configuration changes you made become active, restart the web server.

     ```
     $ sudo apachectl restart
     ```

   - Query Apache's active virtual hosts and modules to confirm your changes are in effect:

     ```
     $ apachectl -t -D DUMP_VHOSTS
     $ apachectl -t -D DUMP_MODULES
     ```

Important
Apache must be configured to use the prefork MPM because P4PHP does not support threaded operation.

The prefork ***MPM*** is the default for Linux and OSX Apache installations, so you may not have to do anything.

For more information on Apache MPMs and configuration, see:

https://httpd.apache.org/docs/2.2/mpm.html

https://httpd.apache.org/docs/2.4/mpm.html

PHP configuration

PHP can vary between OS distributions; see the documentation specific to your installation of PHP.

1. First determine which `php.ini` file is in use by the PHP Apache module. Note that it may not necessarily be the same `php.ini` file that is in use when calling PHP from the command line (running `php --ini` from the command line reports this).

   If you are having trouble determining which `php.ini` the PHP Apache module is using, create a PHP file that can be served through Apache with the following contents:

   ```
   <?php phpinfo();?>
   ```

   Point your browser to this file and look for this table row in the resulting table:

   ```
   Loaded Configuration File
   ```

2. Ensure that `date.timezone` is set correctly for your system.

   Some distributions do not make a default timezone available to PHP, so the best practice to set the timezone for PHP explicitly. See the list of supported timezones.

   An example `date.timezone` setting in `php.ini`:

   ```
   date.timezone = America/Vancouver
   ```

3. Verify that the iconv, json, and session extensions are present.

   They are usually enabled by default, although you may have to install packages for them through your OS distribution. Verify they are present by searching for their respective names in the `phpinfo` output above.

4. Enable P4PHP, the Perforce extension for PHP:

   For Swarm to communicate with Helix server, it needs the P4PHP extension. Swarm comes with a number of variants of the P4PHP binary, for Linux platforms (32- and 64-bit) and Mac OS X (Darwin), and for PHP 5.3, 5.4, and 5.5.

   > **Note**
   > For Linux, the default variants are compiled with glibc 2.11. We have also included PHP 5.3 variants compiled with glibc 2.3.3 to support those customers on older distributions, such as Red Hat Enterprise Linux 5.9.

   To enable P4PHP, edit the web server's `php.ini` file and add the following line:

   ```
   extension=/path/to/swarm/p4-bin/bin.<platform>/perforce-<variant>.so
   ```

   Example 1: for a 64-bit Linux system running PHP 5.4:

   ```
   extension=/path/to/swarm/p4-bin/bin.linux26x86_64/perforce-php54.so
   ```

   Example 2: for a 32-bit Linux system running PHP 5.3 with glibc older than 2.11:

   ```
   extension=/path/to/swarm/p4-bin/bin.linux26x86/perforce-php53-glibc2.3.3.so
   ```

   Alternatively, copy the extension file to the default location for PHP extensions, and then just add this line instead:

   ```
   extension=perforce-<variant>.so
   ```

5. Restart Apache for the changes to become active.

6. To verify that P4PHP is active, navigate to the `phpinfo` file you created above. You should then see a perforce section (search for "Perforce Module"). It should report that the module is enabled and display the version information.

> **Note**
> Be aware that any operating system upgrades on the machine hosting Swarm may involve updates to PHP. If this occurs, the `php.ini` needs to be updated to point to the correct ***`variant`*** of P4PHP to match the version of PHP that the upgraded operating system is using.

Alternative PHP Cache (APC) extension for PHP

APC is a free, open, and robust framework for caching and optimizing PHP intermediate code. Enabling APC improves Swarm performance by caching Swarm's compiled bytecode.

For more information, see:
https://secure.php.net/apc
https://pecl.php.net/package/APC

1. We recommend that you install APC from your OS distribution, via `apt-get`, `yum`, etc. If your distribution does not offer the APC package for PHP, do so via ***PECL*** (although you may have to resolve system dependencies):

   ```
   $ sudo pecl install apc
   ```

2. Verify that APC is enabled in your PHP Apache module's `php.ini` file (as determined in the section for P4PHP). You may need to add the following line:

   ```
   extension=apc.so
   ```

3. Restart Apache for the changes to become active.

4. To verify that APC is active, navigate to the phpinfo file you created earlier. You should then see an apc section (you may have to search for "APC Support"). It should report its version information and a table for its directives.

   We currently do not have any specific recommendations for which APC directives to set.

   Warning
   Once you have completed installing and enabling P4PHP and APC, we recommend that you remove the phpinfo file you created to avoid disclosing information about your installation.

Zend OPCache extension for PHP

Zend OPCache, like APC, improves PHP performance by storing compiled PHP code into a cache, removing the need for PHP to load and parse scripts on each request.

Zend OPCache is bundled with PHP 5.5.x, and is recommended for use instead of APC. For PHP versions 5.3 and 5.4, Zend OPCache is not bundled, but is available via PECL.

For more information, see:
https://secure.php.net/opcache
https://pecl.php.net/package/ZendOpcache

Install Zend OPCache

1. We recommend that you install Zend OPCache from your OS distribution, via `apt-get`, `yum`, etc. If your distribution does not offer the Zend OPCache package for PHP, do so via PECL (although you may have to resolve system dependencies):

   ```
   $ sudo pecl install zendopcache
   ```

2. Verify that Zend OPCache is enabled in your PHP Apache module's php.ini file (as determined in the section above for P4PHP). You may need to add the following line:

   ```
   zend_extension=/path/to/opcache.so
   ```

3. Restart Apache for the changes to become active.

4. To verify that Zend OPCache is active, navigate to the phpinfo file you created earlier. You should then see a Zend `OPcache` section. It should report its version information and a table for its directives.

   We currently do not have any specific recommendations for which Zend OPCache directives to set.

   Warning
   Once you have completed installing and enabling P4PHP and Zend OPCache, we recommend that you remove the phpinfo file you created to avoid disclosing information about your installation.

Enable Zend OPCache

For PHP 5.5+, while Zend OPCache is included, it may not be enabled. If you choose to use it, you may need to add the following line to your PHP Apache module's `php.ini` file:

```
opcache.enable = 1
```

If you have edited `php.ini`, ensure that you restart Apache for the changes to become active.

ImageMagick (imagick) extension for PHP

Imagick is a PHP extension that integrates the ImageMagick graphics library's API for the creation and manipulation of images. Enabling Imagick improves Swarm's ability to preview graphics formats that web browsers typically cannot display.

For more information, see:
https://secure.php.net/imagick
https://pecl.php.net/package/imagick

1. We recommend that you install Imagick from your OS distribution, via `apt-get`, `yum`, etc. If your distribution does not offer the imagick package for PHP, do so via PECL (although you may have to resolve system dependencies):

   ```
   $ sudo pecl install imagick
   ```

2. Verify that imagick is enabled in your PHP Apache module's `php.ini` file (as determined in the section above for P4PHP). You may need to add the following line:

   ```
   extension=imagick.so
   ```

3. Restart Apache for the changes to become active.

4. To verify that imagick is active, navigate to the `phpinfo` file you created earlier. You should then see an imagick section. It should report its version information and a table for its directives, supported image file formats, and more.

   Warning
   Once you have completed installing and enabling P4PHP and imagick, we recommend that you remove the `phpinfo` file you created to avoid disclosing information about your installation.

Swarm configuration

Now that Swarm is ready for use, you need to configure it to work in your environment.

Note
Helix Versioning Engine can refer to a Helix server machine (`p4d`), proxy, broker, replica, edge server, or commit server. It does not refer to a service user; service users are used to coordinate replication in a Helix Versioning Engine. For simplicity, the term *Helix server* is used to refer to any configuration of a Helix Versioning Engine machine.

Swarm configuration file

Create a file named `config.php` under the data directory with the following contents:

```
<?php
    return array(
        'p4' => array(
            'port'      => 'my-helix-versioning-engine:1666',
            'user'      => 'admin_userid',
            'password'  => 'admin user ticket or password',
         ),
         'log' => array(
             'priority'  => 3, // 7 for max, defaults to 3
         ),
         'mail' => array(
             'transport' => array(
                 'host' => 'my.mx.host',
             ),
         ),
    );
```

- For the port value, replace my-helix-versioning-engine:1666 with the P4PORT value used to connect to your Helix server.

  Important
  If your Helix server is deployed using the commit-edge architecture, ensure that Swarm's port value points to the commit server.

  For more information, see "Commit-edge Architecture" in the *Helix Versioning Engine Administrator Guide: Multi-Site Deployment*.

  Warning
  If the port points to a Helix Broker, ensure that the broker does not delegate commands to different replicas, edge servers, or proxies. Such delegation can cause odd problems or outright failures in Swarm.

  Swarm needs to have a consistent, current view of the state of Helix server, and works best when it connects to a central/commit server.

- For the user value, replace ***admin_userid*** with a normal Helix Core userid that has ***admin***-level access to Helix server.

- For the password value, while a plain-text password works, we recommend that you use a ticket value instead. Obtain the ticket value for the ***admin_userid*** during login with this command:

  ```
  $ p4 -p my-helix-versioning-engine:1666 -u admin_userid login -p
  ```

  > **Note**
  > For Helix server with the security configurable set to level 3, or when authentication is configured to use LDAP, ticket-based authentication is required.

  > **Important**
  > When using ticket-based authentication, ensure that the ticket has a very long expiration. We recommend creating a group with an unlimited timeout, and adding ***admin_userid*** user to this group.
  >
  > An expired ticket causes many Swarm operations to fail.

  You can determine when the admin userid's ticket will expire with:

  ```
  $ p4 -p my-helix-versioning-engine:1666 -u admin_userid -P
  ticket_value login -s
  ```

  For more information about tickets, see the section "Ticket-based authentication" in the *Helix Versioning Engine Administrator Guide: Fundamentals*.

- For the host value, replace ***my.mx.host*** with the hostname of the mail exchanger service that Swarm should use to send its email notifications.

> **Note**
> Since this configuration file contains the credentials for a Helix Core*admin*-level user, we recommend that this file's ownership and permissions be adjusted such that only the web server user can read the file, and that no user can write the file.

Optional additional Swarm configuration

Swarm provides optional functionality that could be enabled at this time:

- JIRA integration
- LibreOffice

Swarm hostname

Swarm normally auto-detects the hostname it operates under. In some system configuration, the auto-detection logic might not choose the correct hostname, such as when there are multiple virtual hosts configured for a single Swarm instance. When auto-detection chooses the wrong hostname, email notifications, worker startup, and more could be affected.

If you need to specify the Swarm hostname, see "hostname" on page 260 for details.

Establish trigger token

Trigger tokens prevent unwanted events from influencing Swarm operations; trigger requests to Swarm without a valid token are ignored.

1. Log in to Swarm as a *super* user.
2. Click your userid, found at the right of the main toolbar.
3. Select **About Swarm**.

   The **About Swarm** dialog appears and Swarm generates an API token if none exists.
4. Note the trigger token value, from the bottom of the dialog, for use in the next section. Click the token to select it, which makes it easy to copy.

Helix Core configuration for Swarm

Now that you have a configured instance of Swarm, the last piece is to configure your Helix server to tell Swarm about interesting events. This is accomplished through the use of triggers.

For more information about Helix Core triggers, see "Using triggers to customize behavior" in *Helix Versioning Engine Administrator Guide: Fundamentals*.

> **Note**
> *Helix Versioning Engine* can refer to a Helix server machine (**p4d**), proxy, broker, replica, edge server, or commit server. It does not refer to a service user; service users are used to coordinate replication in a Helix Versioning Engine. For simplicity, the term *Helix server* is used to refer to any configuration of a Helix Versioning Engine machine.

Using triggers to push events to Swarm

Helix server provides a facility called *triggers* to customize the operation of the server, or to invoke additional processing for specific kinds of versioning operations. Swarm provides a trigger script written in Perl that notifies Swarm about activity within the Helix server.

See "Trigger options" on page 299 for more information on configuring the Perl trigger.

Set up Swarm triggers with a CentOS/RHEL 6 hosted Helix server

The latest updates to the trigger script require dependencies which are not available on version 6 of the CentOS/RHEL platform. If you are running the triggers on CentOS 6, then the triggers have a dependency on curl which may not be installed by default on a minimal install of CentOS 6.

Set up Swarm triggers with a Windows-hosted Helix server

1. Ensure that the required "Trigger dependencies" on page 44 have been installed on the machine hosting Helix server.
2. Install the Swarm trigger script on the Helix server.

   If the Swarm trigger script has not already been installed on your Helix server machine, you need to copy `p4-bin/scripts/swarm-trigger.pl` from your Swarm install to the Helix server machine so that it can be executed by the Helix server.

   > **Note**
   > If you are using the Swarm OVA, the full path to the trigger script within the OVA's filesystem is: `/opt/perforce/swarm/p4-bin/scripts/swarm-trigger.pl`

3. Configure the Swarm trigger script.

   You need to use the API token established in the "Establish trigger token" on page 81 section.

   `swarm-trigger.pl` can be configured directly, but the preferred approach is to create a configuration file called `swarm-trigger.conf`, as using the configuration file greatly simplifies upgrades. `swarm-trigger.conf` should be created in the same directory as `swarm-trigger.pl`.

   If your Helix server is configure using a commit-edge configuration, copy `swarm-trigger.pl` and `swarm-trigger.conf` to the commit server and all edge servers, making sure that they exist in the same path on all servers.

   The following is a sample of what your `swarm-trigger.conf` should contain:

```
# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or
"https://".
SWARM_HOST="http://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To
obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to
see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional)
# For enforcing reviewed changes, optionally specify the normal
Perforce user
# with admin privileges (to read keys); if not set, will use whatever
Perforce
# user is set in environment.
ADMIN_USER=

# ADMIN_TICKET_FILE (optional)
# For enforcing reviewed changes, optionally specify the location of
the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very
long
# timeout; then, manually login as this user from the Perforce server
machine to
# set the ticket.
ADMIN_TICKET_FILE=

# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls
whether
# the SSL certificate is validated or not. By default this is set to
1, which
# means any SSL certificates must be valid. If the web server is
using a self
# signed certificate, then this must be set to 0.
VERIFY_SSL=1
```

Modify the `swarm-trigger.conf` configuration file to set the **SWARM_HOST** and the **SWARM_TOKEN** variables appropriately.

Note
You may need to edit the trigger script to specify the full path to `curl.exe`.

Important
If you specify **ADMIN_USER**, the ticket contained in `%USERPROFILE%/p4tickets.txt` (or the ticket file specified with **ADMIN_TICKET_FILE**) must use the port that was used to start Helix server. For example, if `p4d` is started with:

```
C:\> p4d -p my-helix-versioning-engine:1666 ...
```

then the ticket for the *`admin`* user specified with **ADMIN_USER** should be established with:

```
C:\> p4 -p my-helix-versioning-engine:1666 -u admin_userid
login
```

If the ticket was established using the wrong port, the error message you encounter includes the *`port`* that the trigger is attempting to use:

```
'swarm.strict.1' validation failed: Invalid login credentials to
[port] within this trigger script; please contact your
administrator.
```

4. Verify that the trigger script executes correctly.

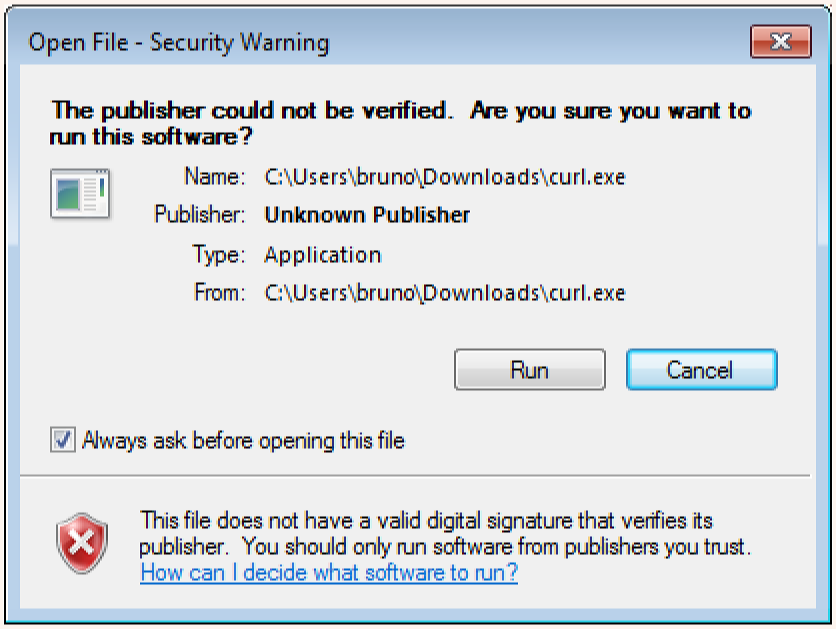
   Run:

   ```
   C:\> perl "C:\path\to\swarm-trigger.pl" -t ping -v 0
   ```

   Use the full path to `perl` if it is not available in your command path.

   You should expect to see no output. If the trigger is misconfigured, such as using an invalid trigger token, you would see an error.

   Warning
   Installation of the triggers may cause a security warning dialog to appear when `curl.exe` executes:

   Open File - Security Warning

   The publisher could not be verified. Are you sure you want to run this software?

   Name: C:\Users\bruno\Downloads\curl.exe
   Publisher: **Unknown Publisher**
   Type: Application
   From: C:\Users\bruno\Downloads\curl.exe

   Run | Cancel

   ☑ Always ask before opening this file

   This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust.
   How can I decide what software to run?

   If this occurs, the triggers hang, creating zombie perl processes. Due to the way triggers are invoked by Helix server, the dialog is normally not visible even though Windows is waiting on interaction.

   To resolve this:

   a. Clear the **Always ask before opening this file** check box and click **Run**.
   b. Right-click `curl.exe`, select **Properties**, and click **Unblock**.

5. Update the Helix Core triggers table to run the trigger script.

   As a Helix Core user with *super* privileges, edit the Helix Core trigger table by running the `p4 triggers` command and add the following lines (including the initial tab character):

```
	swarm.job        form-commit   job
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
job -v %formname%"
	swarm.user       form-commit   user
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
user -v %formname%"
	swarm.userdel    form-delete   user
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
userdel -v %formname%"
	swarm.group      form-commit   group
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
group -v %formname%"
	swarm.groupdel   form-delete   group
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
groupdel -v %formname%"
	swarm.changesave form-save     change
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
changesave -v %formname%"
	swarm.shelve     shelve-commit //...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
shelve -v %change%"
	swarm.commit     change-commit //...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
```

```
commit -v %change%"
#       swarm.enforce.1 change-submit  //DEPOT_PATH1/...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
enforce -v %change% -p %serverport%"
#       swarm.enforce.2 change-submit  //DEPOT_PATH2/...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
enforce -v %change% -p %serverport%"
#       swarm.strict.1  change-content //DEPOT_PATH1/...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
strict -v %change% -p %serverport%"
#       swarm.strict.2  change-content //DEPOT_PATH2/...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -t
strict -v %change% -p %serverport%"
```

Update the `perl.exe`, trigger script, and configuration file paths in each line above to reflect the actual paths on your Helix server.

Important
If your Helix server has SSL enabled and is older than the 2014.1 release, the **`%serverport%`** trigger variable does not include the necessary transport indicator, which can cause the **`enforce`** and **`strict`** triggers to fail.

To solve this problem, add **`ssl:`** immediately before **`%serverport%`** in the trigger lines. For example:

```
#       swarm.enforce.1 change-submit  //DEPOT_PATH1/...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -
t enforce -v %change% -p ssl:%serverport%"
#       swarm.enforce.2 change-submit  //DEPOT_PATH2/...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -
t enforce -v %change% -p ssl:%serverport%"
```

```
#      swarm.strict.1  change-content //DEPOT_PATH1/...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -
t strict -v %change% -p %ssl:serverport%"
#      swarm.strict.2  change-content //DEPOT_PATH2/...
"%quote%C:\path\to\perl.exe%quote% %quote%C:\path\to\swarm-
trigger.pl%quote% -c %quote%C:\path\to\swarm-trigger.conf%quote% -
t strict -v %change% -p %ssl:serverport%"
```

Warning
The use of `%quote%` is not supported on 2010.2 servers (it is harmless though); if you are using this version, ensure that you do not have any spaces in the path to `perl.exe` or the script's path.

Note
The last four trigger lines are commented out as they are optional, and require that the `DEPOT_PATH1` and `DEPOT_PATH2` values are configured appropriately.

- The first two lines configure the enforce feature, which rejects any submitted changes that are not tied to an approved review.
- The second two lines configure the strict feature, which rejects any submitted changes when the contents of the changelist do not match the contents of its associated approved review.

If you need to apply *`enforce`* or strict to more depot paths, copy the lines and tweak their depot paths as necessary.

The trigger script can provide the list of trigger lines that should work, with little to no adjustment, by executing it with the `-o` option:

```
C:\> perl "C:\path\to\swarm-trigger.pl" -o
```

6. Configure the Helix server to promote all shelved changes.

```
C:\> p4 configure set dm.shelve.promote=1
```

When this configurable is set, Swarm has access to all shelved changelists, which is a requirement for pre-commit reviews. When it is not set, users connected to an edge server must remember to use the `-p` option when shelving files to promote their shelves to the commit server when initiating a pre-commit review.

7. Optionally forward logins to the commit server.

   If you intend to use P4V and its Swarm integration, you should consider forwarding logins to the commit server. See "P4V Authentication" on page 247 for details.

Setup Swarm triggers with a Linux-hosted Helix server

1. Ensure that the required "Trigger dependencies" on page 44 have been installed on the machine hosting Helix server.
2. Copy the perl trigger to the Helix server.

   If your Helix server is version 2014.1 (or later), we recommend submitting the trigger file, `p4-bin/scripts/swarm-trigger.pl`, to Helix Core and running it from the depot. The recommended depot location would be `//.swarm/triggers/swarm-trigger.pl`, especially if you have already setup "Comment attachments" on page 244.

   If your Helix server is older than version 2014.1, or prefer that the trigger exist in the filesystem, you must copy the `p4-bin/scripts/swarm-trigger.pl` script to the server hosting Helix server. If your Helix server deployment uses the commit-edge architecture, the script must also be copied to all edge servers, and it must exist in the same path on all servers.

   Note
   If you are using the Swarm OVA, the full path to the trigger script within the OVA's filesystem is: `/opt/perforce/swarm/p4-bin/scripts/swarm-trigger.pl`

3. Configure the Perl trigger.

   You need to use the API token established in the "Establish trigger token" on page 81.

   `swarm-trigger.pl` can be configured directly, but the preferred approach is to create a configuration file called `swarm-trigger.conf`, as using the configuration file greatly simplifies upgrades.

   If you are using the Swarm triggers package described in "Swarm packages" on page 48, the file is available at `/opt/perforce/etc/swarm-trigger.conf`, otherwise create `swarm-trigger.conf` in the same directory as `swarm-trigger.pl`.

   If you submitted the trigger script to the depot in the previous step, you should similarly submit the configuration file to the depot. The recommended path is `//.swarm/triggers/swarm-trigger.conf`.

   If you copied the trigger script to the commit server and all edge servers in the previous step, also copy the configuration file to the commit server and all edge servers, making sure that it exists in the same path on all servers.

   The following is a sample of what your `swarm-trigger.conf` should contain:

```
# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or
"https://".
SWARM_HOST="https://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To
obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to
see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional)
# For enforcing reviewed changes, optionally specify the normal
Perforce user
# with admin privileges (to read keys); if not set, will use whatever
Perforce
# user is set in environment.
ADMIN_USER=
# ADMIN_TICKET_FILE (optional)
# For enforcing reviewed changes, optionally specify the location of
the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very
long
# timeout; then, manually login as this user from the Perforce server
machine to
# set the ticket.

# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls
whether
# the SSL certificate is validated or not. By default this is set to
1, which
# means any SSL certificates must be valid. If the web server is
using a self
# signed certificate, then this must be set to 0.
VERIFY_SSL=1
```

Modify the `swarm-trigger.conf` configuration file to set the `SWARM_HOST` and the `SWARM_TOKEN` variables appropriately.

Note
`swarm-trigger.pl` looks for configuration in the following files. Variables defined in the later files will override the earlier defined variables of the same name:

- Variables set inside the `swarm-trigger.pl` script itself
- `/etc/perforce/swarm-trigger.conf`
- `/opt/perforce/etc/swarm-trigger.conf`
- The `swarm-trigger.conf` file stored in the same directory as `swarm-trigger.pl`
- Any file passed to the `swarm-trigger.pl` script using the -c command line argument

Important
If you specify **ADMIN_USER**, the ticket contained in **`$HOME/.p4tickets`** (or the ticket file specified with **ADMIN_TICKET_FILE**) must use the port that was used to start the Helix server. For example, if **`p4d`** is started with:

```
$ p4d -p my-helix-versioning-engine:1666 ...
```

then the ticket for the ***`admin`*** user specified with **ADMIN_USER** should be established with:

```
$ p4 -p my-helix-versioning-engine:1666 -u admin_userid
login
```

If the ticket was established using the wrong port, the error message you encounter includes the port that the trigger is attempting to use:

```
'swarm.strict.1' validation failed: Invalid login credentials to
[port] within this trigger script; please contact your
administrator.
```

4. Ensure that the script has execute permissions.

Important
Skip this step if you have committed the script to the Helix server.

```
$ chmod +x /path/to/swarm-trigger.pl
```

5. Verify that the trigger script executes correctly.

Important
Skip this step if you have committed the script to the Helix server.

```
$ /path/to/swarm-trigger.pl -t ping -v 0
```

You should expect to see no output. If the trigger is misconfigured, such as using an invalid trigger token, you would see an error.

Note
Run the trigger script without any arguments to see additional usage information.

6. Update the Helix Core triggers table to run the Perl trigger.

   As a Helix Core user with super privileges, edit the Helix Core triggers table by running the `p4 triggers` command and add the appropriate set of lines (including the initial tab character):

   a. If you have committed both the trigger script and the configuration file to the Helix server:

```
	swarm.job        form-commit   job    "%//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t job
  -v %formname%"
	swarm.user       form-commit   user   "%//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t user
  -v %formname%"
	swarm.userdel    form-delete   user   "%//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t userdel
  -v %formname%"
	swarm.group      form-commit   group  "%//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t group
  -v %formname%"
	swarm.groupdel   form-delete   group  "%//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t groupdel
  -v %formname%"
	swarm.changesave form-save     change "%//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t
changesave -v %formname%"
	swarm.shelve     shelve-commit //...  "%//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t shelve
  -v %change%"
	swarm.commit     change-commit //...  "%//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t commit
  -v %change%"
#	swarm.enforce.1 change-submit  //DEPOT_PATH1/...
"%//.swarm/triggers/swarm-trigger.pl% -c
%//.swarm/triggers/swarm-trigger.conf% -t enforce -v %change% -p
%serverport%"
#	swarm.enforce.2 change-submit  //DEPOT_PATH2/...
"%//.swarm/triggers/swarm-trigger.pl% -c
```

```
%//.swarm/triggers/swarm-trigger.conf% -t enforce -v %change% -p
%serverport%"
#       swarm.strict.1  change-content //DEPOT_PATH1/...
"%//.swarm/triggers/swarm-trigger.pl% -c
%//.swarm/triggers/swarm-trigger.conf% -t strict -v %change% -p
%serverport%"
#       swarm.strict.2  change-content //DEPOT_PATH2/...
"%//.swarm/triggers/swarm-trigger.pl% -c
%//.swarm/triggers/swarm-trigger.conf% -t strict -v %change% -p
%serverport%"
```

b. If you have copied the trigger script and configuration file to common paths on all servers:

```
       swarm.job        form-commit   job    "%quote%/path/to/swarm-
trigger.pl%quote% -c %quote%/path/to/swarm-trigger.conf%quote% -t
job        -v %formname%"
       swarm.user       form-commit   user   "%quote%/path/to/swarm-
trigger.pl%quote% -c %quote%/path/to/swarm-trigger.conf%quote% -t
user       -v %formname%"
       swarm.userdel    form-delete   user   "%quote%/path/to/swarm-
trigger.pl%quote% -c %quote%/path/to/swarm-trigger.conf%quote% -t
userdel    -v %formname%"
       swarm.group      form-commit   group  "%quote%/path/to/swarm-
trigger.pl%quote% -c %quote%/path/to/swarm-trigger.conf%quote% -t
group      -v %formname%"
       swarm.groupdel   form-delete   group  "%quote%/path/to/swarm-
trigger.pl%quote% -c %quote%/path/to/swarm-trigger.conf%quote% -t
groupdel   -v %formname%"
       swarm.changesave form-save     change "%quote%/path/to/swarm-
trigger.pl%quote% -c %quote%/path/to/swarm-trigger.conf%quote% -t
changesave -v %formname%"
       swarm.shelve     shelve-commit //...  "%quote%/path/to/swarm-
trigger.pl%quote% -c %quote%/path/to/swarm-trigger.conf%quote% -t
shelve     -v %change%"
       swarm.commit     change-commit //...  "%quote%/path/to/swarm-
trigger.pl%quote% -c %quote%/path/to/swarm-trigger.conf%quote% -t
commit     -v %change%"
#      swarm.enforce.1 change-submit  //DEPOT_PATH1/...
"%quote%/path/to/swarm-trigger.pl%quote% -c
%quote%/path/to/swarm-trigger.conf%quote% -t enforce -v %change%
-p %serverport%"
#      swarm.enforce.2 change-submit  //DEPOT_PATH2/...
"%quote%/path/to/swarm-trigger.pl%quote% -c
%quote%/path/to/swarm-trigger.conf%quote% -t enforce -v %change%
-p %serverport%"
#      swarm.strict.1  change-content //DEPOT_PATH1/...
```

```
"%quote%/path/to/swarm-trigger.pl%quote% -c
%quote%/path/to/swarm-trigger.conf%quote% -t strict -v %change% -
p %serverport%"
#      swarm.strict.2  change-content //DEPOT_PATH2/...
"%quote%/path/to/swarm-trigger.pl%quote% -c
%quote%/path/to/swarm-trigger.conf%quote% -t strict -v %change% -
p %serverport%"
```

Update the trigger script and configuration file paths in each line above to reflect the actual paths on your Helix server.

Important

If your Helix server has SSL enabled and is older than the 2014.1 release, the `%serverport%` trigger variable does not include the necessary transport indicator, which can cause the enforce and strict triggers to fail.

To solve this problem, add `ssl:` immediately before `%serverport%` in the trigger lines. For example:

```
#      swarm.enforce.1 change-submit  //DEPOT_PATH1/...
"%quote%/path/to/swarm-trigger.pl%quote% -c
%//.swarm/triggers/swarm-trigger.conf% -t enforce -v %change% -p
ssl:%serverport%"
#      swarm.enforce.2 change-submit  //DEPOT_PATH2/...
"%quote%/path/to/swarm-trigger.pl%quote% -c
%//.swarm/triggers/swarm-trigger.conf% -t enforce -v %change% -p
ssl:%serverport%"
#      swarm.strict.1  change-content //DEPOT_PATH1/...
"%quote%/path/to/swarm-trigger.pl%quote% -c
%//.swarm/triggers/swarm-trigger.conf% -t strict -v %change% -p
ssl:%serverport%"
#      swarm.strict.2  change-content //DEPOT_PATH2/...
"%quote%/path/to/swarm-trigger.pl%quote% -c
%//.swarm/triggers/swarm-trigger.conf% -t strict -v %change% -p
ssl:%serverport%"
```

> **Note**
> The last four trigger lines in either block are commented out as they are optional, and require that the `DEPOT_PATH1` and `DEPOT_PATH2` values are configured appropriately.
>
> - The first two lines configure the ***`enforce`*** feature, which rejects any submitted changes that are not tied to an approved review.
> - The second two lines configure the ***`strict`*** feature, which rejects any submitted changes when the contents of the changelist do not match the contents of its associated approved review.
>
> If you need to apply ***`enforce`*** or ***`strict`*** to more depot paths, copy the lines and tweak their depot paths as necessary.

The trigger script can provide the list of trigger lines that should work, with little to no adjustment, by executing it with the `-o` option:

```
$ /path/to/swarm-trigger.pl -o
```

7. Configure Helix server to promote all shelved changes.

```
$ p4 configure set dm.shelve.promote=1
```

When this configurable is set, Swarm has access to all shelved changelists, which is a requirement for pre-commit reviews. When it is not set, users connected to an edge server must remember to use the `-p` option when shelving files to promote their shelves to the commit server when initiating a pre-commit review.

9. Optionally, forward logins to the commit server.

If you intend to use P4V and its Swarm integration, you should consider forwarding logins to the commit server. See "P4V Authentication" on page 247 for details.

Hiding Swarm storage from regular users

Swarm information storage uses Helix server's ***`keys`*** facility. By default, users with *list*-level access can search keys and potentially obtain information they would not otherwise have access to, and users with *review*-level access can write or modify keys potentially corrupting or destroying data.

We recommend that you set the `dm.keys.hide` configurable to 2 to require *admin*-level access for searching and modifying keys. Note that `dm.keys.hide` is available in Helix server versions 2013.1 and newer.

When `dm.keys.hide` is set to 2, both the `p4 keys` and `p4 key` commands require *admin*-level access in the Helix server. When `dm.keys.hide` is set to 1, only the `p4 keys` command requires *admin*-level access in the Helix server. When `dm.keys.hide` is set to 1, or is not set, users who know (or can deduce) key names can read values (if they have *list*-level access) or write values (if they have *review*-level access) with the `p4 key` command.

To set `dm.keys.hide`:

```
$ p4 configure set dm.keys.hide=2
```

To confirm the current value of `dm.keys.hide`:

```
$ p4 configure show dm.keys.hide
```

To unset `dm.keys.hide`:

```
$ p4 configure unset dm.keys.hide
```

Handling Exclusive Locks

Swarm takes copies of files when it is creating reviews. Some of the files managed by Helix server may be limited to 'exclusive open' by having the filetype modifier '+l' set. This file-level setting ensures only one user at a time can open the file for editing.

To allow Swarm to work with these 'exclusive open' files, you must enable `filetype.bypasslock` in the Helix server configuration.

To set `filetype.bypasslock`:

```
$ p4 configure set filetype.bypasslock=1
```

To confirm the current value of `filetype.bypasslock`:

```
$ p4 configure show filetype.bypasslock
```

To unset `filetype.bypasslock`:

```
$ p4 configure unset filetype.bypasslock
```

If this setting is not enabled in Helix server, Swarm will report exceptions when working with exclusively opened files similar to "Cannot unshelve review (x). One or more files are exclusively open", and noting that you must have the `filetype.bypasslock` configurable enabled.

Set up a recurring task to spawn workers

To ensure that incoming Helix Core events are automatically processed by Swarm, it is important to set up a cron job to do this. The cron job can be installed on any host, although you may want to place this on the Swarm host.

The recurring task to invoke Swarm workers, installed in a later step, requires either of:

- curl
  https://curl.haxx.se/download.html

  > **Note**
  > For Windows, `curl.exe` depends on `MSVCR100.dll`. You can get a copy by installing the **Microsoft Visual C++ Redistributable Package**, available for:

- 32-bit systems: https://www.microsoft.com/download/en/details.aspx?id=5555
- 64-bit systems: https://www.microsoft.com/download/en/details.aspx?id=14632

If you install Swarm with HTTPS, **`curl.exe`** requires recent CA certificates (or HTTPS connections silently fail). You can get a copy of the **`cacert.pem`** from:

https://curl.haxx.se/docs/caextract.html

Once downloaded, copy **`cacert.pem`** to the same folder where you installed **`curl.exe`**, and rename it to **`curl-ca-bundle.crt`**.

Warning
If **`curl`** (or **`curl.exe`** on Windows) cannot execute as expected, trigger execution may block or fail. For example, if **`MSVCR100.dll`** is missing from a Windows system, invoking **`curl.exe`** causes a dialog to appear.

Prior to configuring the triggers, verify that **`curl`** executes. On Linux systems, run:

```
$ curl -h
```

On Windows systems, run:

```
C:\> curl.exe -h
```

The start of the output should be similar to:

```
Usage: curl [options...] <url>
Options: (H) means HTTP/HTTPS only, (F) means FTP only
     --anyauth       Pick "any" authentication method (H)
 -a, --append        Append to target file when uploading (F/SFTP)
     --cacert FILE   CA certificate to verify peer against (SSL)
     --capath DIR    CA directory to verify peer against (SSL)
...[truncated for brevity]...
```

For a more thorough test that actually fetches content over a network, try the following test:

- For Linux systems, run:

  ```
  $ curl https://www.perforce.com/
  ```

- For Windows systems, run:

  ```
  C:\> curl.exe https://www.perforce.com/
  ```

The output should look like HTML.

- wget
  https://ftp.gnu.org/gnu/wget/

http://gnuwin32.sourceforge.net/packages/wget.htm (for Windows)

Note
If you are using Powershell on Windows systems, be aware that Powershell includes aliases for `curl` and `wget` that call the Powershell command `Invoke-WebRequest` instead of `curl.exe` or `wget.exe`. `Invoke-WebRequest` has different command-line options than either `curl` or `wget`, which can be confusing.

If you want to remove the built-in aliases for `curl` and `wget` from Powershell, follow these steps:

1. Create a Powershell profile (only if you have not already done so):

   ```
   PS C:\> New-Item $profile -force -itemtype file
   ```

2. Edit your profile:

   ```
   PS C:\> notepad $profile
   ```

3. Add the following line to your profile:

   ```
   remove-item alias:curl
   remove-item alias:wget
   ```

4. Save the profile and close `notepad`.
5. Reload your profile:

   ```
   PS C:\> . $profile
   ```

Warning
`curl` or `wget` must be installed or workers do not spawn and Swarm cannot process any events. See below for verification steps.

1. Create a file named `helix-swarm` in `/etc/cron.d`.
2. Edit `/etc/cron.d/helix-swarm` to contain one of the following blocks; select a block depending on whether your system has `curl` or `wget` installed.
   - If you have `curl` installed:

     ```
     # This ensures that a worker is fired up every minute
     * * * * * nobody curl -so /dev/null -m5
     https://myswarm.url/queue/worker
     ```

   - If you have `wget` installed:

     ```
     # This ensures that a worker is fired up every minute
     * * * * * nobody wget -q -O /dev/null -T5
     https://myswarm.url/queue/worker
     ```

3. Replace ***myswarm.url*** above with the actual URL you have configured for Swarm (which may include a sub-folder or a custom port).

   If the cron job is running on the Swarm host, and you have specified the correct `hostname` item in the Environment configuration, this can be set to `localhost`.

   In the example configuration lines above, where you see `-m5` or `-T5`, the `5` is the number of seconds that the cron task will wait for a response from the Swarm host. When the cron task is installed on the Swarm host, such as in the Swarm OVA, that value could be reduced to `1` seconds (e.g. `-m1` or `-T1`).

   Note
   If you configure Swarm to use HTTPS, and you install a self-signed certificate, the cron jobs need to be adjusted to avoid certificate validity test which could cause silent failures to process events.

   - If you have `curl` installed:

     ```
     # This ensures that a worker is fired up every minute
     * * * * * nobody curl -so /dev/null --insecure -m5
     https://myswarm.url/queue/worker
     ```

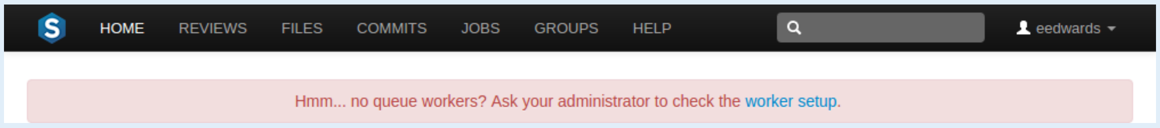
   - If you have `wget` installed:

     ```
     # This ensures that a worker is fired up every minute
     * * * * * nobody wget -q -O /dev/null --no-check-
     certificate -T5 https://myswarm.url/queue/worker
     ```

4. Save the edited file.

You are now all set to start using Swarm. Enjoy!

Note
If the recurring task is disabled, or stops functioning for any reason, logged-in users see the following error message when Swarm detects that no workers are running:



Hmm... no queue workers? Ask your administrator to check the worker setup.

curl/wget verification

The cron job depends on having `curl` or `wget` installed, as indicated in "Runtime dependencies" on page 40.

To verify that `curl` or `wget` is installed, use the `which` command. For example:

```
$ which curl
```

If you see any output, the referenced command is installed.

Post-install configuration options

There are a few options for customizing your Swarm installation's operation. This section covers the options that are officially supported:

- "HTTPS" on the facing page
- "Run Swarm in a sub-folder of an existing web site" on page 109
- "Run Swarm's virtual host on a custom port" on page 112

Before undertaking any of the following customization options, ensure that you have backed up your Swarm virtual host configuration. Choose the most appropriate option:

- If your Apache configuration directory contains the directories `sites-available` and `sites-enabled`:

  ```
  $ cd /path/to/apache/configuration/..
  $ cp -a sites-available sites-available.bak
  ```

  Important
  If the `sites-enabled` directory contains files, and not just symbolic links, you need to backup this folder as well:

  ```
  $ cd /path/to/apache/configuration/..
  $ cp -a sites-enabled sites-enabled.bak
  ```

- For CentOS/RHEL systems, if you used the "Swarm packages" on page 48 to install Swarm:

  ```
  $ cd /path/to/apache/configuration/..
  $ cp -a conf.d conf.d.bak
  ```

- Otherwise, back up your Apache configuration.

HTTPS

This section describes how to make your Swarm installation more secure by using HTTPS.

Before you begin the following procedure, locate your system's Apache configuration. Common configuration directories include:

- `/etc/httpd/conf/`
- `/etc/apache2/`
- `/Applications/XAMPP/etc/`

Within the Apache configuration path, the main Apache configuration file is usually named one of the following:

- `httpd.conf`
- `apache2.conf`

A longer discussion on the possible locations and names of Apache configuration files is available here: https://wiki.apache.org/httpd/DistrosDefaultLayout

1. Enable SSL in Apache.

   If the Apache utility `a2enmod` is installed:

   ```
   $ sudo a2enmod ssl
   ```

   Without the `a2enmod` utility, edit the Apache configuration file by hand. Locate your Apache configuration file for modules and either uncomment or add the following lines:

   ```
   LoadModule  ssl_module  libexec/apache2/mod_ssl.so
   ```

2. Create a directory to store certificates.

   ```
   $ sudo mkdir -p /etc/apache2/ssl
   ```

3. Create a certificate/key pair.

```
$ cd /etc/apache2/ssl
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -
keyout apache.key -out apache.crt
```

This command generates a private key and a certificate. To form the certificate, `openssl` prompts you for several details:

```
Generating a 2048 bit RSA private key
...................+++
....................................+++
writing new private key to 'apache.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CA
State or Province Name (full name) [Some-State]:British Columbia
Locality Name (eg, city) []:Victoria
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Perforce
Software
Organizational Unit Name (eg, section) []:Swarm development team
Common Name (e.g. server FQDN or YOUR name) []:myswarm.host
Email Address []:admin@myswarm.host
```

The output above includes some example details. You should replace anything in italics with your own details. Since the certificate request details that can help users determine whether your certificate is valid, enter legitimate information whenever possible.

Important
The Common Name field must match the hostname for your Swarm installation exactly.

4. Secure the certificate directory.

   ```
   $ sudo chmod 600 /etc/apache2/ssl
   ```

5. Edit the virtual host configuration.

Note
The virtual host configuration should be in the file you backed up initially.

- For Apache 2.2, edit the virtual host configuration to match:

```
<VirtualHost *:80>
       ServerName myswarm.host
       ServerAlias myswarm
       ErrorLog "/path/to/apache/logs/myswarm.error_log"
       CustomLog "/path/to/apache/logs/myswarm.access_log" common
       DocumentRoot "/path/to/swarm/public"
       <Directory "/path/to/swarm/public">
               AllowOverride All
               Require all granted
       </Directory>

       Redirect permanent / https://myswarm.host
</VirtualHost>

<VirtualHost *:443>
       SSLEngine on
       SSLCertificateFile /etc/apache2/ssl/apache.crt
       SSLCertificateKeyFile /etc/apache2/ssl/apache.key

       ServerName myswarm.host
       ServerAlias myswarm
       ErrorLog "/path/to/apache/logs/myswarm.error_log"
       CustomLog "/path/to/apache/logs/myswarm.access_log" common
       DocumentRoot "/path/to/swarm/public"
       <Directory "/path/to/swarm/public">
               AllowOverride All
               Order allow,deny
               Allow from all
       </Directory>
</VirtualHost>
```

- For Apache 2.4, edit the virtual host configuration to match:

```
<VirtualHost *:80>
       ServerName myswarm
       ServerAlias myswarm.host
       ErrorLog "/path/to/apache/logs/myswarm.error_log"
       CustomLog "/path/to/apache/logs/myswarm.access_log" common
       DocumentRoot "/path/to/swarm/public"
       <Directory "/path/to/swarm/public">
               AllowOverride All
               Require all granted
       </Directory>
</VirtualHost>

<VirtualHost *:443>
       SSLEngine on
       SSLCertificateFile /etc/apache2/ssl/apache.crt
       SSLCertificateKeyFile /etc/apache2/ssl/apache.key

       ServerName myswarm.host
       ServerAlias myswarm
       ErrorLog "/path/to/apache/logs/myswarm.error_log"
       CustomLog "/path/to/apache/logs/myswarm.access_log" common
       DocumentRoot "/path/to/swarm/public"
       <Directory "/path/to/swarm/public">
               AllowOverride All
               Require all granted
       </Directory>
</VirtualHost>
```

See Apache's virtual host documentation for details:
https://httpd.apache.org/docs/2.2/vhosts/
https://httpd.apache.org/docs/2.4/vhosts/

6. Customize the virtual host definition.
   a. Replace ***myswarm.host*** with the hostname for Swarm on your network.
   b. Replace ***myswarm*** with the name of the subdomain hosting Swarm. Many administrators choose swarm.

      Note the string myswarm in the log file paths: this should match the subdomain name and prefix for the log files, to help coordinate the active host with the log files for that host. Doing this is particularly useful when your Apache server hosts multiple instances of Swarm.
   c. Replace ***/path/to/apache/logs*** with the path where your Apache store its log files. Apache's log files are typically named `access_log` and `error_log`.
   d. Replace ***/path/to/swarm*** with the path to the Swarm directory.
7. Restart your web server.

   ```
   $ sudo apachectl restart
   ```
8. Adjust your firewall configuration to allow connections to the standard SSL port for web servers.
   - For CentOS/RHEL 6.6+:

     ```
     $ sudo lokkit -p 443:tcp
     ```
   - For CentOS/RHEL 7+:

     ```
     $sudo firewall-cmd --zone=public --add-port=443/tcp --
     permanent
     $sudo systemctl reload firewalld
     ```
   - For other distributions, consult with your network administrator or operating system documentation to determine how to adjust your firewall configuration.
9. Test your HTTPS URL from a web browser.

Important
If the ***myswarm.host*** value in the virtual host configuration and the certificate do not match, the P4V integration with Swarm fails with the message `SSL handshake failed`.

Also, when a reverse DNS lookup is performed, `myswarm.host` should be the answer when querying for the Swarm server's IP address.

Run Swarm in a sub-folder of an existing web site

If you cannot run Swarm in its own virtual host, which might be necessary when you do not control the hostname to be used with Swarm, installing Swarm in a sub-folder of an existing virtual host configuration can be a good solution.

Installing Swarm in a sub-folder requires modification of the previous installation steps covered in this chapter:

- The "Apache configuration" on page 70 is entirely different; instead of establishing a new virtual host, you need to modify an existing virtual host configuration. Often, this would be Apache's default site.
- Swarm's configuration file requires an extra item.

The following sections cover the specifics of sub-folder installation.

See "base_url" on page 261 for more details.

Important
If you used the Swarm OVA or "Swarm packages" on page 48 to install Swarm, you can adjust Swarm's configuration using the package configuration script **`/opt/perforce/swarm/sbin/configure-swarm.sh`**.

`configure-swarm.sh` does not read any existing Swarm configuration; you must provide all of the configuration details each time you execute **`configure-swarm.sh`**:

```
$ sudo /opt/perforce/swarm/sbin/configure-swarm.sh -n -p myp4host:1666
-u swarm -w password -e mx.example.com -H myhost -B /swarm
```

In the example above, the **`-B`** option is used to specify the name of the sub-folder.

If you use **`configure-swarm.sh`** to adjust the Swarm configuration, you only need to follow the "Apache configuration" below steps described below; all of the changes listed in the "Swarm configuration" on page 112 section below have been completed by **`configure-swarm.sh`**.

Apache configuration

1. Ensure that the ***`SWARM_ROOT`*** is not within the document root of the intended virtual host.

   This step ensures that Swarm's source code and configuration is impossible to browse, preventing access to important details such as stored credentials, and active sessions and workspaces.

2. Adjust the virtual host configuration that you are already using.

   Note
   Depending on the method used to install Swarm, the filename for virtual host configuration you need to edit is:

   - For Swarm OVA or Swarm package installations, edit **`perforce-swarm-site.conf`**.
   - For manual installations following Swarm's recommended "Apache configuration" on page 70 edit **`swarm`**.
   - For other installations, you may have to edit **`httpd.conf`** or nearby files.

   - For Apache 2.2, add the following lines to the virtual host definition:

     ```
     Alias /swarm SWARM_ROOT/public

     <Directory "SWARM_ROOT/public">
       AllowOverride All
       Order allow,deny
       Allow from All
     </Directory>
     ```

   - For Apache 2.4, add the following lines to the virtual host definition:

     ```
     Alias /swarm SWARM_ROOT/public
     <Directory "SWARM_ROOT/public">
     AllowOverride All
     Require all granted
     </Directory>
     ```

   The **`Alias`** line configures Apache to respond to requests to **`https://myhost/swarm`** with content from Swarm's **`public`** folder. You can change the **`/swarm`** portion of the Alias line to anything you want.

   The **`<Directory>`** block grants access to everything within Swarm's **`public`** folder. Replace **SWARM_ROOT** with the actual path to Swarm.

3. Restart your web server.

   ```
   $ sudo apachectl restart
   ```

Swarm configuration

To successfully operate within a sub-folder, the `"swarm_root" on page 296/data/config.php` file needs to be adjusted to contain the following lines (as a peer of the p4 item):

```
        'environment' => array(
                  'base_url' => '/swarm'
         ),
```

Ensure that `/swarm` matches the first item in the `Alias` line in the virtual host configuration.

See "Environment" on page 259 for more details.

Cron configuration

Swarm's recurring task configuration must be updated to reflect the sub-folder that you have configured in Apache's and Swarm's configurations.

1. Edit `/etc/cron.d/helix-swarm`.
2. Replace:

   ```
   https://myswarm.url/queue/worker
   ```

   with:

   ```
   https://myswarm.url/swarm/queue/worker
   ```

   Where ***`myswarm.url`*** is the hostname of your Swarm installation, and ***`swarm`*** is the sub-folder you wish to use.
3. Save the edited file.

   New workers should be started at the start of the next minute.

Run Swarm's virtual host on a custom port

If you cannot run Swarm on port 80 (or port 443 for HTTPS), perhaps because you do not have root access, it is possible to run Swarm on a custom port.

Installing Swarm to use a custom port requires modification of the previous installation steps covered in this chapter: The Apache configuration is slightly different, requiring modification of Swarm's virtual host definition.

The following section covers the specifics of the custom port configuration.

Note
In addition to the following instructions, you may also need to apply the `external_url` item described in the "Environment" on page 259 section if your Swarm is behind a proxy, or you have multiple Swarm instances connected to Helix server.

Important
If you used the Swarm OVA or "Swarm packages" on page 48 to install Swarm, you can adjust Swarm's configuration using the package configuration script `/opt/perforce/swarm/sbin/configure-swarm.sh`.

`configure-swarm.sh` does not read any existing Swarm configuration; you must provide all of the configuration details each time you execute `configure-swarm.sh`:

```
$ sudo /opt/perforce/swarm/sbin/configure-swarm.sh -n -p
myp4host:1666 -u swarm -w password -e mx.example.com -H myhost
-P 8080
```

In the example above, the `-P` option is used to specify the custom port that Swarm should use.

If you use `configure-swarm.sh` to adjust Swarm's configuration, follow the additional steps that it describes. Once those steps are complete, do not perform any of the steps described below.

Apache configuration

1. Edit the virtual host configuration.

   Note
   Depending on the method used to install Swarm, the filename for virtual host configuration you need to edit is:

   - For Swarm OVA or Swarm package installations, edit `perforce-swarm-site.conf`.
   - For manual installations following Swarm's recommended Apache configuration, edit `swarm`.
   - For other installations, you may have to edit `httpd.conf` or nearby files.

   a. Add the following line *outside* of the `<VirtualHost>` block:

   ```
   Listen 8080
   ```

   b. Edit the `<VirtualHost *:80>` line to read:

   ```
   <VirtualHost *:8080>
   ```

   For both lines, replace *`8080`* with the custom port you wish to use.

   Important
   If you choose a port that is already in use, Apache refuses to start.

2. Restart your web server.

   ```
   $ sudo apachectl restart
   ```

3. Adjust your firewall configuration to allow connections to the custom port.
   - For CentOS/RHEL 6.6+:

     ```
     $ sudo lokkit -p 8080:tcp
     ```

     Replace *8080* with the custom port you wish to use.
   - For CentOS/RHEL 7+:

     ```
     $ sudo firewall-cmd --zone=public --add-port=8080/tcp --
     permanent
     $ sudo systemctl reload firewalld
     ```

     Replace *8080* with the custom port you wish to use.
   - For other distributions, consult with your network administrator or operating system documentation to determine how to just your firewall configuration.

Cron configuration

Swarm's recurring task configuration must be updated to reflect the custom port that you have configured in Apache's configuration.

1. Edit `/etc/cron.d/helix-swarm`.
2. Replace:

   ```
   https://myswarm.url/queue/worker
   ```

   with:

   ```
   https://myswarm.url:8080/queue/worker
   ```

   Where *myswarm.url* is the hostname of your Swarm installation, and *8080* is the custom port you wish to use.
3. Save the edited file.

   New workers should be started at the start of the next minute.

Upgrading Swarm

The section covers upgrading Swarm to a newer release. If you are not already running Swarm, none of these instructions apply to you. Instead, see the Swarm installation instructions.

If you have installed Swarm via packages, see the package update instructions.

Note
The instructions below can be applied to an OVA. The OVA's SWARM_ROOT, the folder where Swarm is installed, is `/opt/perforce/swarm`.

However, we recommend downloading the new OVA and then following the OVA setup steps. This provides you with an upgraded Swarm plus an updated web hosting environment within the OVA, which can include distribution, web server, PHP, and security updates.

If you have customized the original OVA's Swarm configuration, copy `/opt/perforce/swarm/data/config.php` to the same path in the new OVA.

Copy all token files in `/opt/perforce/swarm/data/queue/tokens/` to the same path in the new OVA.

If you are running the Swarm 2014.2 OVA, or newer, Swarm was installed using system packages and can be upgraded by following the package update instructions.

The following process attempts to minimize downtime, but a short period of downtime for Swarm users is unavoidable. There should be no downtime for your Helix server. After a successful upgrade, all Swarm users are logged out.

If you are using Swarm in a production environment, we encourage you to test this upgrade process in a non-production environment first.

Warning
P4PHP should be upgraded to the version included in the new Swarm release. If you have already configured PHP to use the Swarm-provided P4PHP (as recommended), this happens automatically. If you have manually installed P4PHP in some other fashion, upgrade P4PHP before you perform any of the upgrade steps below. See "PHP configuration" on page 73 for details.

Upgrade Swarm

If you have not already done so, download the Swarm TAR option.

The steps in this section describe how to upgrade Swarm using the provided archive file. `SWARM_ROOT` refers to the current Swarm installation.

1. Expand the new **`swarm.tgz`**:

   ```
   $ tar -zxf swarm.tgz
   ```

   The contents of `swarm.tgz` are expanded into a top-level folder named **`swarm-`*`version`***, where ***`version`*** corresponds to the version downloaded. This directory is identified as **SWARM_NEW** below.

2. Move **SWARM_NEW** to be a peer of **SWARM_ROOT**:

   ```
   $ mv SWARM_NEW SWARM_ROOT/../
   ```

3. Copy the *`SWARM_ROOT`*`/data/config.php` file from `SWARM_ROOT` to `SWARM_NEW`:

   ```
   $ cp -p SWARM_ROOT/data/config.php SWARM_NEW/data/
   ```

4. Create the queue token directory:

   ```
   $  mkdir SWARM_NEW/data/queue
   ```

5. Copy the existing trigger token(s):

   ```
    $ sudo cp -pR SWARM_ROOT/data/queue/tokens SWARM_
   NEW/data/queue/
   ```

6. Assign correct ownership to the new Swarm's data directory:

   ```
   $ sudo chown -R www-data SWARM_NEW/data
   ```

   Note
   The *`www-data`* user above is an example of what the web server user name might be, and can vary based on distribution or customization. For example, the user is typically `apache` for Red Hat/Fedora/CentOS, `www-data` for Debian/Ubuntu, `wwwrun` for SuSE, `_www` for Mac OSX.

7. Copy the new Swarm trigger script to your Helix Versioning Engine machine. The trigger script is `SWARM_NEW/p4-bin/scripts/swarm-trigger.pl`, and requires installation of Perl 5.08+ (use the latest available) on the Perforce server machine. If Swarm is using SSL, then the triggers also require the `IO::Socket::SSL` Perl module.

   Warning
   Do not overwrite any existing trigger script at this time. Give the script a new name, for example: `swarm-trigger-new.pl`

8. Configure the Swarm trigger script by creating, in the same directory on the Helix server machine, `swarm-trigger.conf`. It should contain:

```
# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or
"https://".
SWARM_HOST="http://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To
obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to
see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional)
# For enforcing reviewed changes, optionally specify the normal
Perforce user
# with admin privileges (to read keys); if not set, will use whatever
Perforce
# user is set in environment.
ADMIN_USER=

# ADMIN_TICKET_FILE (optional)
# For enforcing reviewed changes, optionally specify the location of
the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very
long
# timeout; then, manually login as this user from the Perforce server
machine to
# set the ticket.
ADMIN_TICKET_FILE=
# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls
whether
# the SSL certificate is validated or not. By default this is set to
1, which
# means any SSL certificates must be valid. If the web server is
using a self
# signed certificate, then this must be set to 0.
# set the ticket.
VERIFY_SSL=1
```

Fill in the required SWARM_HOST and SWARM_TOKEN variables with the configuration from any previous Swarm trigger script, typically `swarm-trigger.pl`.

> **Note**
> **Swarm 2015.4 and earlier:** Swarm trigger script files were available as shell scripts in these earlier Swarm versions, typically `swarm-trigger.sh`.
>
> When upgrading to Swarm 2016.1 and later the Helix server machine must be configured to use Perl trigger scripts because the shell trigger scripts are no longer supported and have been removed.

> **Note**
> If you already have a `swarm-trigger.conf` file, no additional configuration is required.

9. For Linux systems, ensure that the script is executable:

```
$ sudo chmod +x swarm-trigger-new.pl
```

10. Rename the new trigger script:

```
$ mv swarm-trigger-new.pl swarm-trigger.pl
```

On Windows:

```
C:\> ren swarm-trigger-new.pl swarm-trigger.pl
```

11. Update the triggers in your Helix server.
    a. Run the Swarm trigger script to capture (using **Ctrl+C** on Windows and Linux, **Command+C** on Mac OSX) the trigger lines that should be included in the Perforce trigger table:

    ```
    $ ./swarm-trigger.pl -o
    ```

    On Windows:

    ```
    C:\> path/to/perl swarm-trigger.pl -o
    ```

    b. As a Perforce user with *super* privileges, update the Perforce trigger table by running **`p4 triggers`** command and replacing any **`swarm.*`** lines with the previously captured trigger line output (using **Ctrl+V** on Windows and Linux, **Command+V** on Mac OSX).

    Important
    If you previously customized the Swarm trigger lines, perhaps to apply various "Trigger options" on page 299, be sure to repeat those customizations within the updated trigger lines.

12. Replace the old Swarm with the new Swarm. **Downtime occurs in this step.**

    ```
    $ sudo apache2ctl stop; mv SWARM_ROOT SWARM.old; mv SWARM_NEW
    SWARM_ROOT; sudo apache2ctl start
    ```

Upgrade index

The upgrade index process can be configured to suit your Swarm system specifications. See "Upgrade index" on page 305 for details.

Run the upgrade as an *Admin* user by visiting the following URL:

```
http://SWARM-HOST/upgrade
```

Important
The Swarm index must be upgraded to ensure that the Swarm review history is displayed in the correct order.

Note
This step is only required the first time you upgrade your Swarm system to 2017.3 or later. Subsequent Swarm upgrades do not require the index to be upgraded.

Note
This step is not required if your Swarm system has less than 100 reviews. In this case Swarm will automatically upgrade the index on the fly.

Note
This step is not required if this is a new Swarm installation .

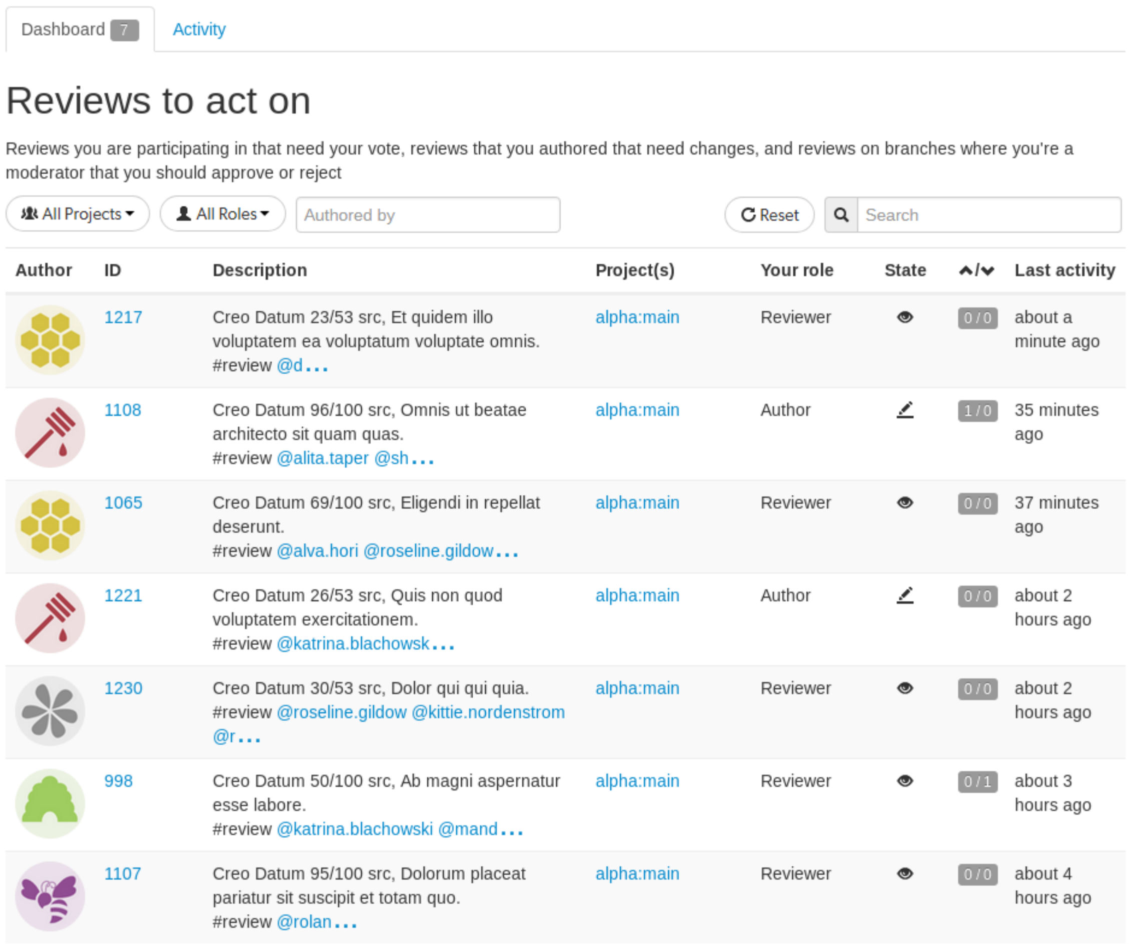
All done!

4 | Basics

This chapter covers the basic operations provided by Swarm. These include:

Dashboard

Your dashboard displays a list of reviews that you may need to act on. Since it is tied to the logged in user, the dashboard is only available if you are logged in. It is available as a tab on the **Home** page of Swarm, alongside "Activity streams" on page 123.



Reviews to act on

Reviews you are participating in that need your vote, reviews that you authored that need changes, and reviews on branches where you're a moderator that you should approve or reject

Author	ID	Description	Project(s)	Your role	State	^/v	Last activity
	1217	Creo Datum 23/53 src, Et quidem illo voluptatem ea voluptatum voluptate omnis. #review @d...	alpha:main	Reviewer		0 / 0	about a minute ago
	1108	Creo Datum 96/100 src, Omnis ut beatae architecto sit quam quas. #review @alita.taper @sh...	alpha:main	Author		1 / 0	35 minutes ago
	1065	Creo Datum 69/100 src, Eligendi in repellat deserunt. #review @alva.hori @roseline.gildow...	alpha:main	Reviewer		0 / 0	37 minutes ago
	1221	Creo Datum 26/53 src, Quis non quod voluptatem exercitationem. #review @katrina.blachowsk...	alpha:main	Author		0 / 0	about 2 hours ago
	1230	Creo Datum 30/53 src, Dolor qui qui quia. #review @roseline.gildow @kittie.nordenstrom @r...	alpha:main	Reviewer		0 / 0	about 2 hours ago
	998	Creo Datum 50/100 src, Ab magni aspernatur esse labore. #review @katrina.blachowski @mand...	alpha:main	Reviewer		0 / 1	about 3 hours ago
	1107	Creo Datum 95/100 src, Dolorum placeat pariatur sit suscipit et totam quo. #review @rolan...	alpha:main	Reviewer		0 / 0	about 4 hours ago

The purpose of the Dashboard is to allow you to focus on reviews that need to be done, so that other users are not blocked. The dashboard lists reviews according to the most recently modified first, and shows your role in the review.

A review is displayed on your dashboard if any of the following criteria are met:

- You are a reviewer or required reviewer, the review status is "needs review" and you have not already voted on it.
- You are a member of a reviewer group or a required reviewer group, the review status is "needs review" and you have not already voted on it. The review will remain on your dashboard even if the group has met its criteria if you have not already voted on it.
- You are the review author and the review status is "needs revision".
- You are a moderator or a member of a moderator group who can approve the review, and the review status is "needs review".

Filtering

The dashboard can be filtered to display only reviews from a particular project, authored by a given user, or matching a role. You can click the **Reset** button to reset these filters.

Bookmarking the Dashboard page records the current state of any filters that are set.

Filtering options are described below:

- **Project**

  Reviews can be filtered by project. By default, all reviews will be shown. You can chose to filter either by all projects you belong to, or an individual project. The drop down menu will only show projects for which there are reviews in your Dashboard.
- **Roles**

  You can filter by your role, limiting results to either only reviews for which you are the author, or to those for which you are the reviewer.
- **Authored by**

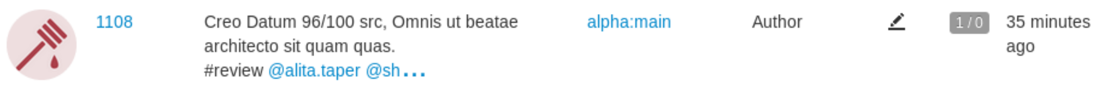
  You can filter the reviews to only those that have been authored by a certain user. Type in this field to get a drop down list of users to filter by.
- **Reset**

  Clicking the Reset button resets all Dashboard filters back to their defaults.
- **Search**

  Typing in the search field filters the reviews by their description.

Review fields

The Dashboard shows a summary of the information for each review.


1108 Creo Datum 96/100 src, Omnis ut beatae architecto sit quam quas. #review @alita.taper @sh... alpha:main Author 1 / 0 35 minutes ago

Reviews that appear here are those which are waiting for action from you. The information presented should help you prioritize what to work on next.

- **Author**

  The author of this review.

- **ID**

  The ID of this review. Click on this to go to the review page.

- **Description**

  The review description. It may be truncated if it is too long, in which case click on the ellipsis ... to expand it.

- **Project(s)**

  List of project branches this review covers. A review may span multiple branches and projects. Click on one of them to navigate to the project page for that branch.

- **Your role**

  The reason this review is in your Dashboard. This can be Author, Reviewer, or Moderator.

- **State**

  The current status of the review.

- **Votes**

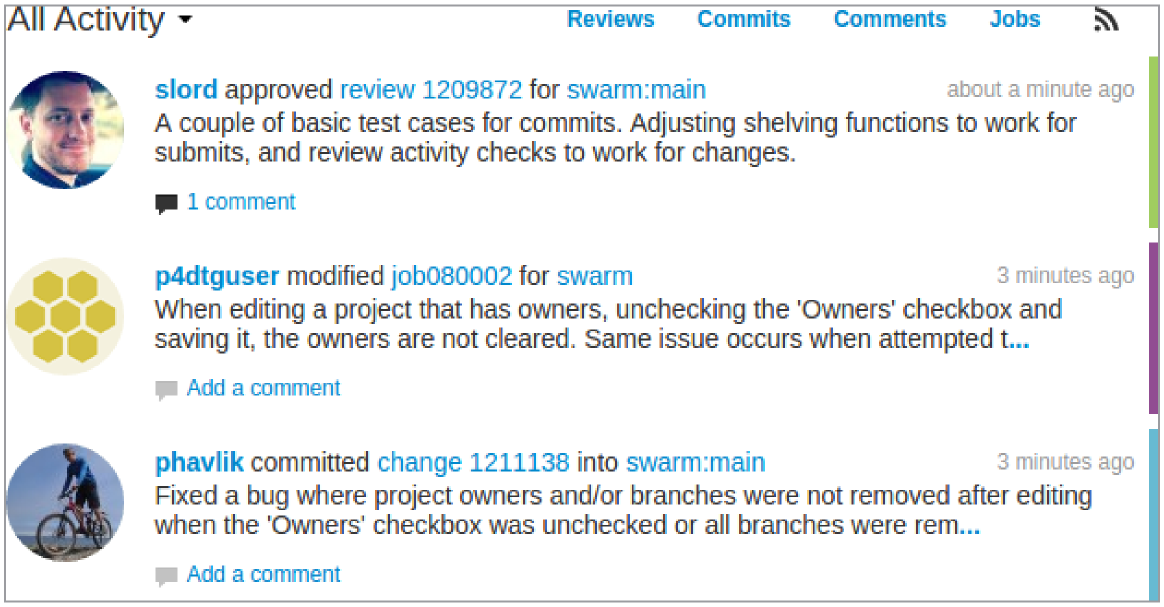
  The double column of votes displays the number of up votes and down votes for the review.

- **Last activity**

  The last time that any changes were made to the review, including votes, comments, commits, and file changes.

Activity streams

An activity stream displays a list of events that have occurred recently in the associated Helix server, whenever changelists are checked in, jobs are created, code reviews are updated, comments posted, etc.



slord approved review 1209872 for swarm:main — about a minute ago
A couple of basic test cases for commits. Adjusting shelving functions to work for submits, and review activity checks to work for changes.
1 comment

p4dtguser modified job080002 for swarm — 3 minutes ago
When editing a project that has owners, unchecking the 'Owners' checkbox and saving it, the owners are not cleared. Same issue occurs when attempted t...
Add a comment

phavlik committed change 1211138 into swarm:main — 3 minutes ago
Fixed a bug where project owners and/or branches were not removed after editing when the 'Owners' checkbox was unchecked or all branches were rem...
Add a comment

Activity streams are presented for global server activity, as well as for events occurring for projects and users. Logged-in users can click the **Activity** drop-down menu to choose between viewing all activity or just the activity of the projects and users that they are following.

Activity streams can be filtered to only display events related to reviews, changes, comments, or jobs. Click a filter label to enable that filter. Click the label again to disable the filter, or click a different filter to change filters.

When active, each filter label displays a distinct background color that matches the color stripe on the right side of each event in the activity stream, to help quickly identify each event's type.

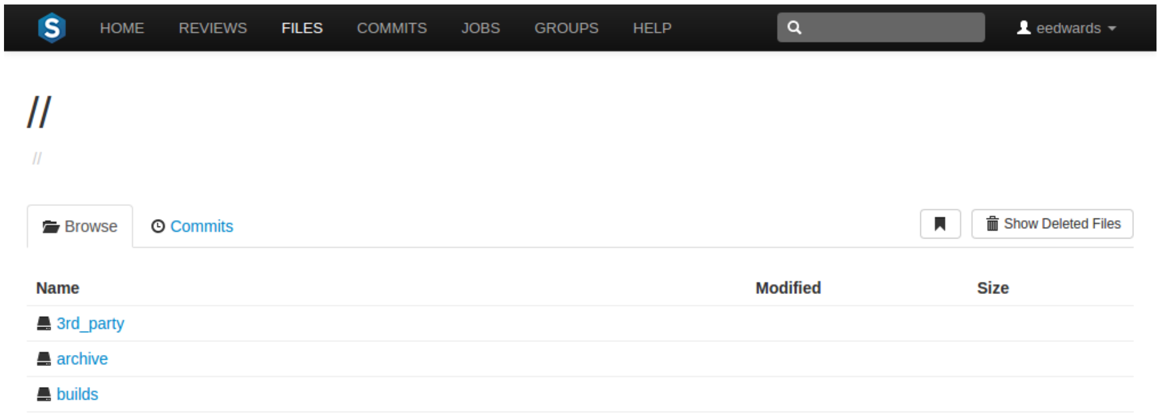
RSS icon Anyone can subscribe to an RSS (Really Simply Syndication) feed for a particular activity stream so that events can be monitored in your favorite feed reader.

Events within an activity stream contain links to their respective resources. Click the links to visit users, changelists, projects, comments, and more.

Each activity stream starts with as many as 50 events. As you scroll down the page, Swarm fetches additional events in batches of 50 until the stream is exhausted. For a long-running server, or a server with significant activity, it could take quite a while to receive all of the events.

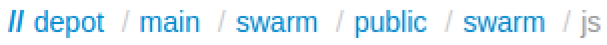
Files

Helix server's primary task is to version files, so Swarm makes it easy to browse the depot. Start browsing by clicking the **Files** link in the top toolbar.



Name	Modified	Size
3rd_party		
archive		
builds		

- Swarm displays a list of breadcrumb links to help you quickly navigate to higher level directories quickly.



- Links with folder icons represent directories of files within the depot. Click a directory link to display the contents of that directory.
- Click the .. link with the up-arrow icon .., when it appears, to navigate to the current directory's parent.
- Links with dog-eared page icons represent individual files within the depot. See "File display" on the facing page for more information.
- Click the **History** tab to display the list of changes made to files in the current directory, or any directories it contains. See "Commits" on page 130 for more information.

Swarm creates links for files and directories wherever they appear in the Swarm UI.

Note
Directories that start with a period, for example .git-fusion, are sorted to appear at the end of the list of directories. The .. link, when it appears, always appears first.

Similarly, files that start with a period, for example .htaccess, are sorted to appear at the end of the list of files.

Downloading files as ZIP archive

When the `zip` command-line tool is available, Swarm can provide a ZIP archive for a file or folder within the Helix server. This makes it easy to get a copy of files without having to setup a client.

> **Note**
> The Download .zip button does not appear if the `zip` command-line tool is not available.

When you click the **Download .zip** button, Swarm performs the following steps:

1. Scan the selected file/folder, to determine if you have permission to access the contents (according to the Helix server protections), and if the file/folder is small enough to be processed by Swarm.
2. Sync the file/folder contents to the Swarm server from the Helix server.
3. Create the ZIP archive by compressing the file content.
4. Start a download of the generated ZIP archive.

You might not see all of the above steps; Swarm caches the resulting ZIP archives so that repeated requests to the same file/folder can skip the sync/compress steps whenever possible.

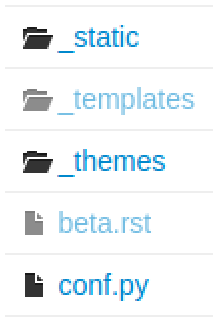
If an error occurs while scanning, syncing, or compressing, Swarm indicates the error.

For information on the configuration for ZIP archives, see "Archives configuration" on page 240.

Browsing deleted files and folders

When the **Show Deleted Files** button is clicked, Swarm toggles the inclusion of deleted folders and files in the file display.

Deleted folders and files are presented slightly muted compared to non-deleted entries.

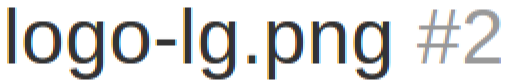


File display

When Swarm is asked to display a file, if the file is a type that Swarm can display, Swarm presents the file's contents. Clicking the **Open** button displays the file content with no surrounding page markup. Clicking the **Download** button causes the file to be downloaded.

When a file is opened it will by default be truncated to 1 MB in size. This limit can be increased (or removed) via the Swarm configurables. See Files configuration for details. This limit does not apply to downloaded files.

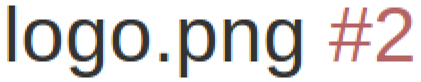
Along with the file's name, Swarm displays the version number for the currently displayed file. For example, this heading indicates that version 2 of `logo-lg.png` is being displayed:



logo-lg.png #2

Every version of a file is available on the **Commits** tab.

If the version of a file being previewed has been deleted, the version number appears in red:

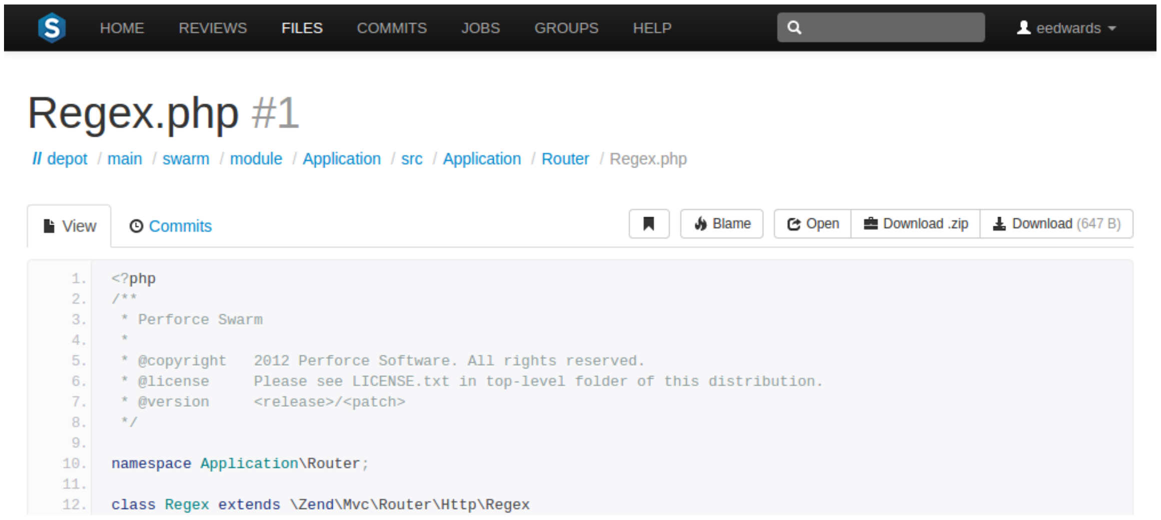


logo.png #2

Text Files

Swarm displays the contents of text files (include the Helix Core filetypes unicode and UTF16) with line numbers. When possible, syntax highlighting is applied to make identification of various elements within the file easier.

For more information on Helix Core filetypes, see "Filte Types" in *P4 Command Reference*.

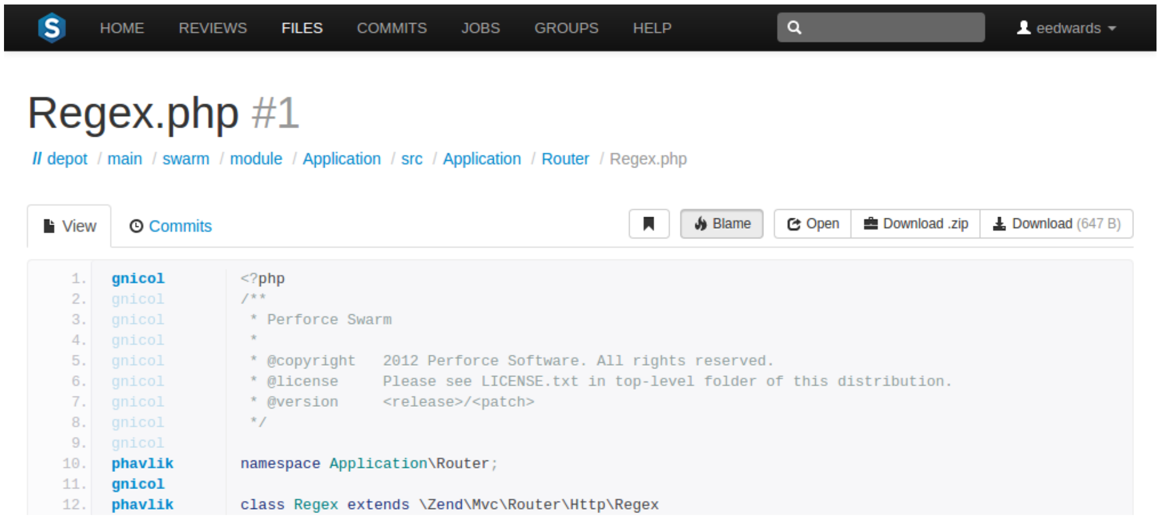


Regex.php #1

// depot / main / swarm / module / Application / src / Application / Router / Regex.php

```
<?php
/**
 * Perforce Swarm
 *
 * @copyright   2012 Perforce Software. All rights reserved.
 * @license     Please see LICENSE.txt in top-level folder of this distribution.
 * @version     <release>/<patch>
 */

namespace Application\Router;

class Regex extends \Zend\Mvc\Router\Http\Regex
```

Click **Blame** to add a column to the display that identifies the userid responsible for each line of the file.



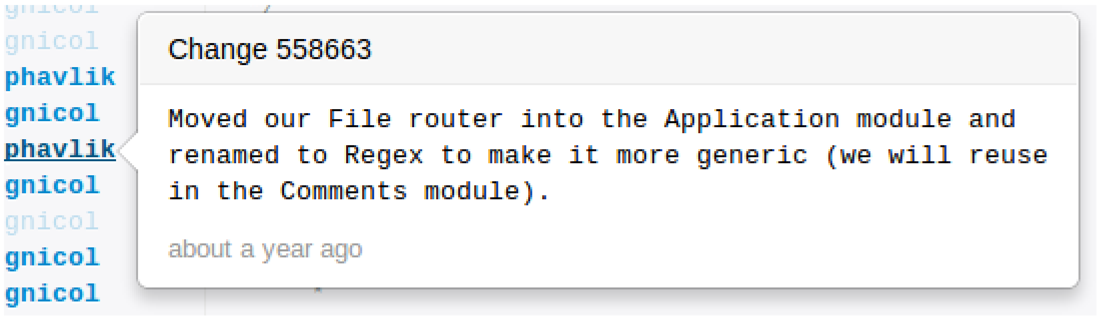
Regex.php #1

// depot / main / swarm / module / Application / src / Application / Router / Regex.php

```
gnicol      <?php
gnicol      /**
gnicol       * Perforce Swarm
gnicol       *
gnicol       * @copyright   2012 Perforce Software. All rights reserved.
gnicol       * @license     Please see LICENSE.txt in top-level folder of this distribution.
gnicol       * @version     <release>/<patch>
gnicol       */
gnicol
phavlik     namespace Application\Router;
gnicol
phavlik     class Regex extends \Zend\Mvc\Router\Http\Regex
```

Each userid presented is a link that, when clicked, displays the changelist that provided the associated text. Muted userids indicate that the associated text is from the same changelist as the line above. For example, the userid *gnicol* is responsible for lines 1 through 9 in the screenshot above.

When you hover your pointer over a userid in the blame column, a tooltip appears displaying the associated changelist description.



Change 558663

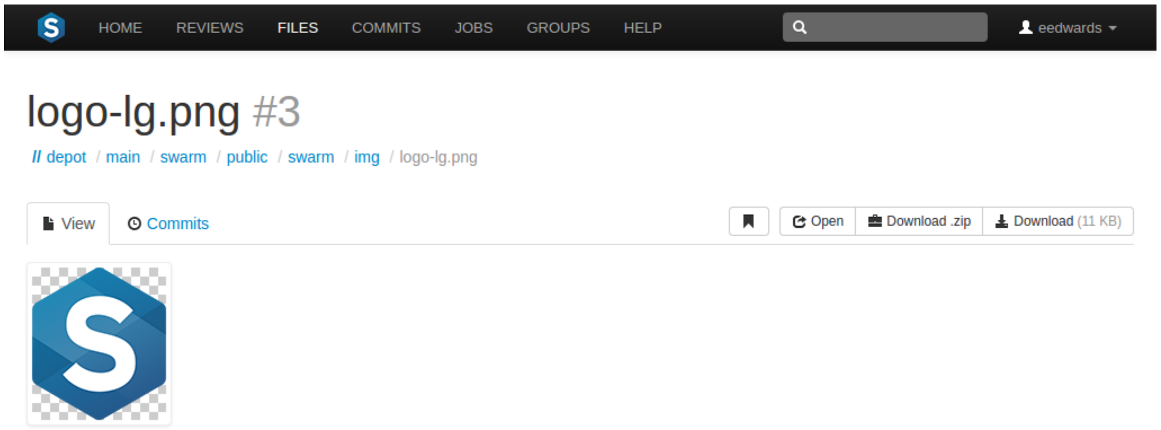
Moved our File router into the Application module and renamed to Regex to make it more generic (we will reuse in the Comments module).

about a year ago

When there are no lines displayed, for example when you are viewing empty or shelved files, the **Blame** button is disabled.

Images

Swarm displays web-safe images.



The checkerboard background in this example is not part of the logo; it helps identify where transparency exists.

Many browsers can display SVG images with no additional plugins, so Swarm attempts to display SVG images rather than displaying the image's definition. When you use a browser that cannot natively display SVG images, you see the broken image icon.

When imagick (an optional module that integrates ImageMagick into PHP) is installed, Swarm can also display the following image formats: BMP, EPS, PSD, TGA, TIFF.

3D models

Swarm 2014.1 includes support for displaying select 3D model file types in the browser:

Supported file types include:

- **DAE** - including any referenced web-safe texture images.
- **STL** - both binary and ASCII versions of the format.
- **OBJ** - including any referenced MTL files, and web-safe texture images.

When Swarm can display a 3D model, it renders a generic grid *stage* and places the model in the center, scaled to make viewing straightforward. A toggle control appears in the top right: when enabled, you can control the view with the mouse, and when not enabled, permits auto-rotation to occur (when possible).

1. Click and hold the **left mouse button** to begin rotating the view. Drag while holding the left mouse button to rotate the view.
2. Click and hold the **right mouse button** to begin panning the view. Drag while holding the right mouse button to pan the view.
3. Roll the **mouse wheel** up or down to adjust the magnification of the view.

When possible, a second control appears allowing you to toggle between showing the model with surfaces, or just showing the model's wireframe.

Note
For systems with hardware acceleration, if your browser supports WebGL and hardware acceleration is enabled, Swarm renders the model and enables auto-rotation.

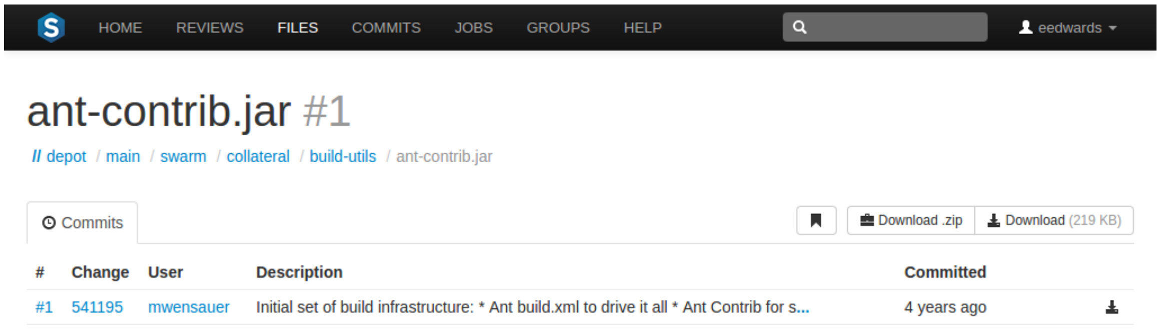
For systems without hardware acceleration or WebGL, but your browser supports HTML5 canvas elements and JavaScript TypedArrays, Swarm renders the model but auto-rotation is disabled. Rendering is likely to be slow and rendering quality is likely to be low.

For browsers without HTML5 canvas elements and JavaScript TypedArrays, no rendering is attempted; instead, users see a message indicating that the browser is not supported.

Other file types

It is possible to view other file types in Swarm, through the addition of additional modules, or by installing "LibreOffice" on page 237 on the Swarm host.

When the file is a type that Swarm cannot display, Swarm presents the file's history, along with the **Download** button.



ant-contrib.jar #1

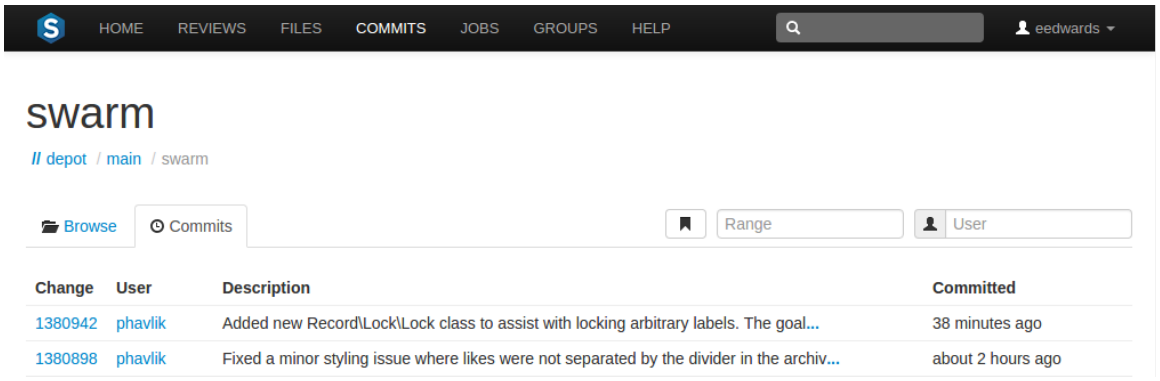
// depot / main / swarm / collateral / build-utils / ant-contrib.jar

#	Change	User	Description	Committed
#1	541195	mwensauer	Initial set of build infrastructure: * Ant build.xml to drive it all * Ant Contrib for s...	4 years ago

Commits

Whenever a new version of a file is checked into the Helix server, a commit record is created. Begin browsing the history of commits by clicking the **Commits** link in the main toolbar.

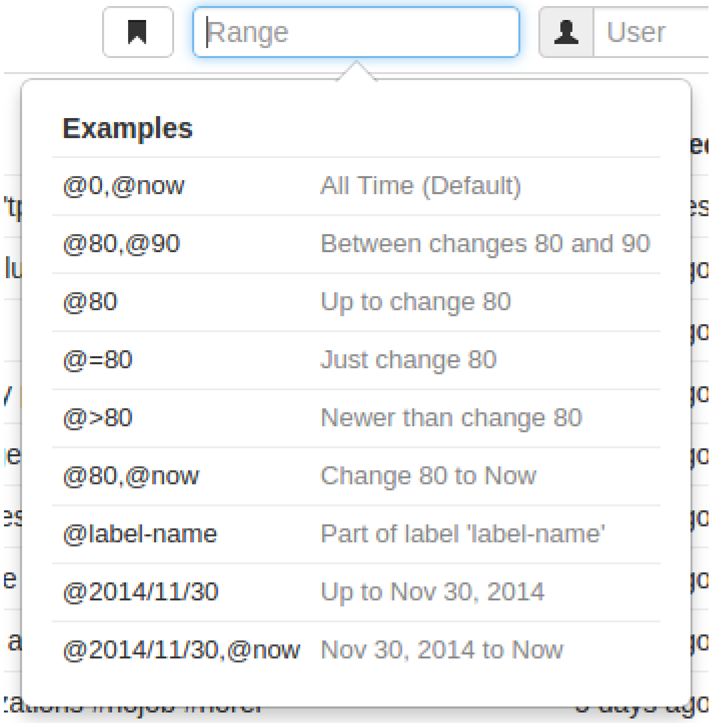
When you are viewing a particular file or directory, clicking the **Commits** tab displays the commits for that location in the depot.



Change	User	Description	Committed
1380942	phavlik	Added new Record\Lock\Lock class to assist with locking arbitrary labels. The goal...	38 minutes ago
1380898	phavlik	Fixed a minor styling issue where likes were not separated by the divider in the archiv...	about 2 hours ago

Range filter

The **Range** field lets you filter the list of changes for the depot path being viewed. When you click the **Range** field, a dropdown syntax guide appears providing sample commit filtering expressions.



Examples

@0,@now	All Time (Default)
@80,@90	Between changes 80 and 90
@80	Up to change 80
@=80	Just change 80
@>80	Newer than change 80
@80,@now	Change 80 to Now
@label-name	Part of label 'label-name'
@2014/11/30	Up to Nov 30, 2014
@2014/11/30,@now	Nov 30, 2014 to Now

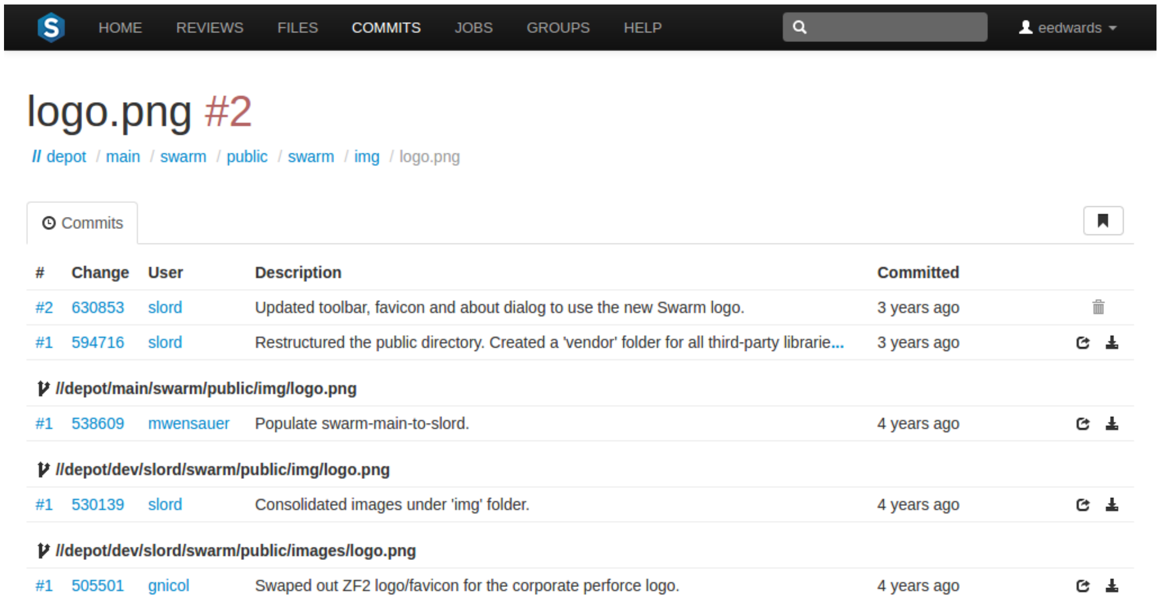
The expressions that can be used within the **Range** field include:

- @0,@now (the default): displays all commits for the current depot path.
- @80,@90: displays all changes between 80 and 90.

- @80: displays all changes up to change 80.
- @=80: displays only change 80. Change 80 might not involve the current depot path, so there may be no commits to display.
- @label-name: any changes represented by label label-name.
- @2014/11/30: displays all changes up to November 30, 2014.
- @2014/11/30,@now: displays all changes from November 30, 2014 to now.

File Commits

A file's commit history presents each version of a file that the Helix server knows about, including the change number, userid, change description, time ago, along with **Open** and **Download** buttons.



#	Change	User	Description	Committed
#2	630853	slord	Updated toolbar, favicon and about dialog to use the new Swarm logo.	3 years ago
#1	594716	slord	Restructured the public directory. Created a 'vendor' folder for all third-party librarie...	3 years ago
//depot/main/swarm/public/img/logo.png				
#1	538609	mwensauer	Populate swarm-main-to-slord.	4 years ago
//depot/dev/slord/swarm/public/img/logo.png				
#1	530139	slord	Consolidated images under 'img' folder.	4 years ago
//depot/dev/slord/swarm/public/images/logo.png				
#1	505501	gnicol	Swaped out ZF2 logo/favicon for the corporate perforce logo.	4 years ago

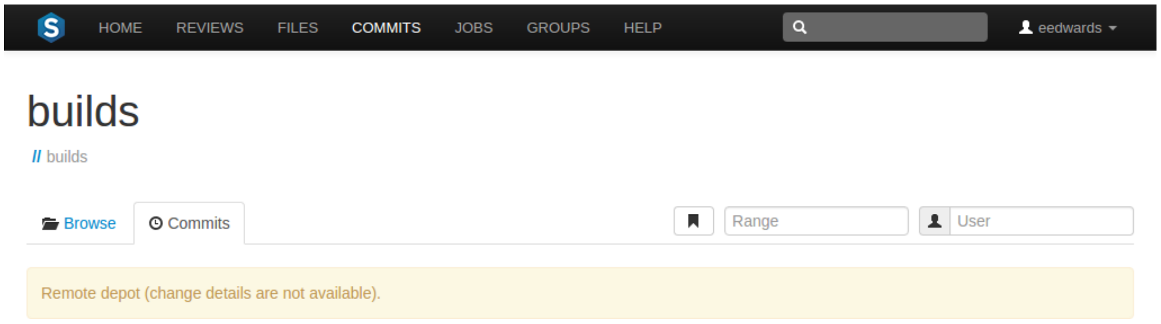
Swarm also displays *contributing commits* when available, such as when a file has been renamed, or integrated from another location in the Helix server.

The trashcan icon represents a deleted version

If a commit represents a deleted revision, the **Open** and **Download** links are replaced with a trashcan icon 🗑 to indicate that this version is no longer available.

Remote depot commits

When your Helix server has a remote depot configured, you can browse the contents of the remote depot, but remote depots do not share their commit history. If you attempt to view the commits of a file provided by a remote depot, Swarm displays:

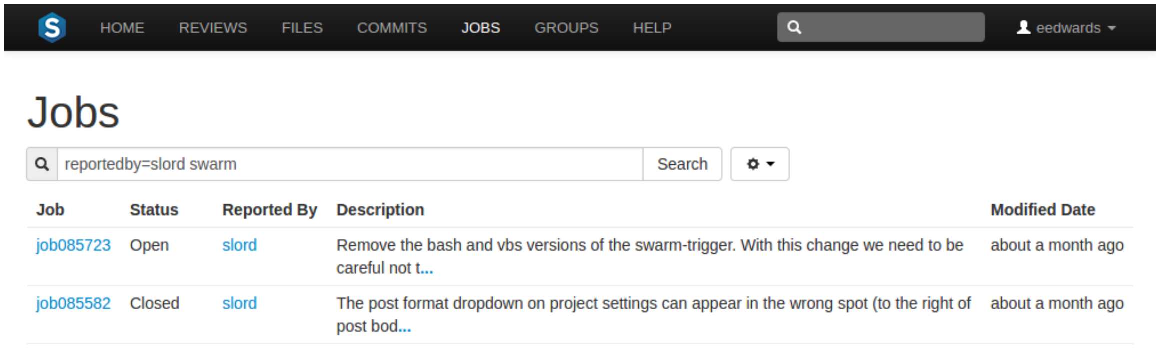


Remote depot (change details are not available).

Jobs

Jobs are a component of Helix Core's defect tracking system and a record of bugs found or improvement requests. Jobs can be associated with changelists to create *fix* records, indicating the work that solved the problem or provided the requested feature. Begin browsing jobs by clicking the **Jobs** link in the main toolbar.

You can search available jobs by entering a job filter in the search box. Words, phrases, and `field=value` pairs can be entered. For example, entering `reportedby=slord swarm` displays jobs that the user slord has reported that also contain the word swarm in its description.



reportedby=slord swarm

Job	Status	Reported By	Description	Modified Date
job085723	Open	slord	Remove the bash and vbs versions of the swarm-trigger. With this change we need to be careful not t...	about a month ago
job085582	Closed	slord	The post format dropdown on project settings can appear in the wrong spot (to the right of post bod...	about a month ago

The fields you can search for depend on the jobspec defined in your Helix server.

Adjusting Jobs columns

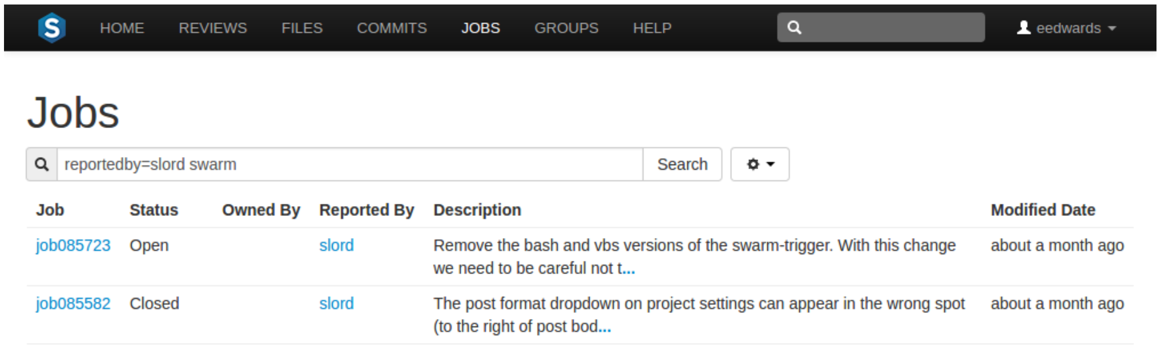
You can configure the columns that are displayed:

1. Click the button beside the search field to display a tooltip menu showing all of the available jobspec fields.
2. While the tooltip menu is displayed, check or uncheck the columns to configure which columns to display.
3. Click the button again, or a blank portion of the page, to hide the tooltip.

You can also adjust the order of the columns, in one of two ways:

1. With the tooltip menu displayed, click and drag column labels up or down to adjust their position.
2. When the tooltip menu is not displayed, click a column heading and drag it to the left or right to move the column to a new position.

The column display updates as columns are rearranged:



Jobs

reportedby=slord swarm

Job	Status	Owned By	Reported By	Description	Modified Date
job085723	Open		slord	Remove the bash and vbs versions of the swarm-trigger. With this change we need to be careful not t...	about a month ago
job085582	Closed		slord	The post format dropdown on project settings can appear in the wrong spot (to the right of post bod...	about a month ago

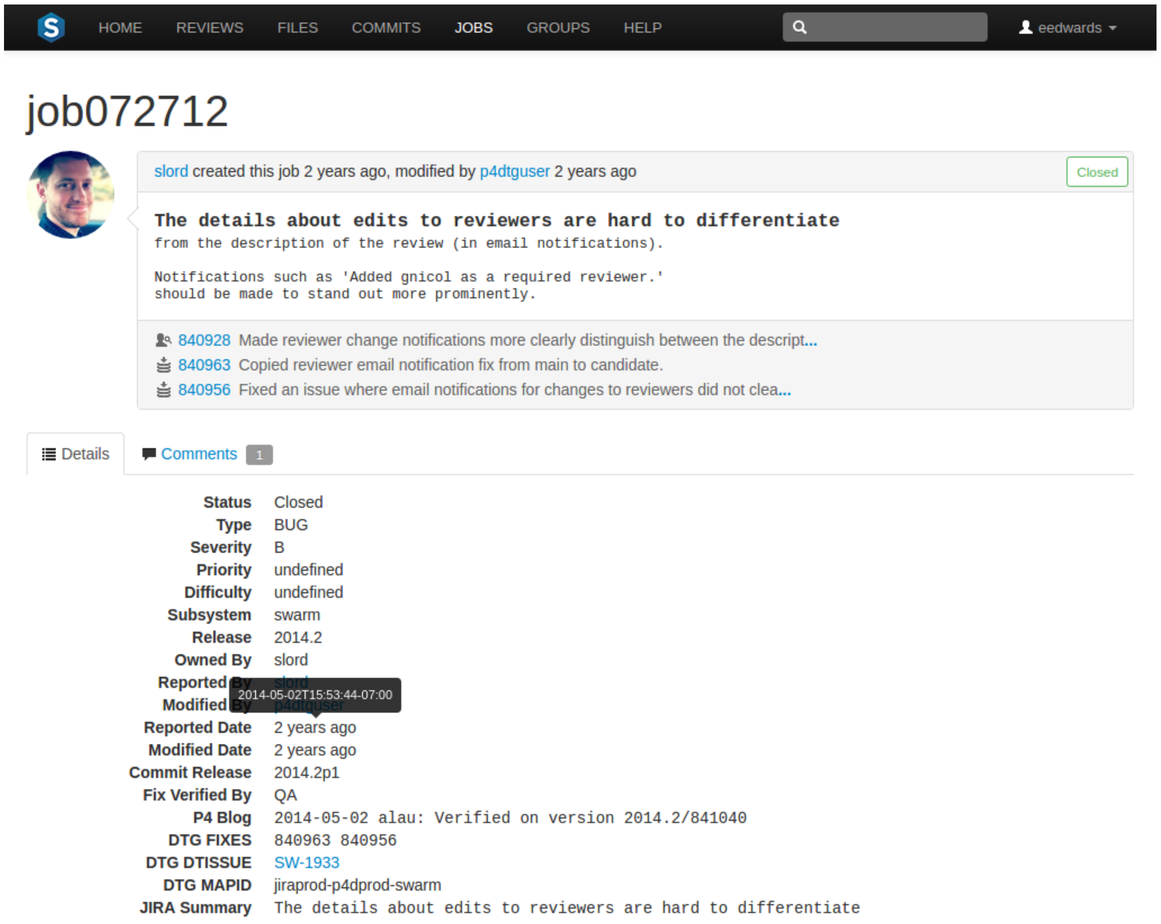
For more information on customizing jobspecs, see "Job Specifications" in *Helix Versioning Engine Administrator Guide: Fundamentals*.

Job display

Jobs are typically identified with the word job followed by six digits, e.g. `job000123`.

View a specific job by clicking on a linked job identifier, or by visiting the URL: `https://myswarm.url/jobs/jobid`

When Swarm displays a job, the presentation is similar to:



job072712

slord created this job 2 years ago, modified by p4dtguser 2 years ago

Closed

```
The details about edits to reviewers are hard to differentiate
from the description of the review (in email notifications).

Notifications such as 'Added gnicol as a required reviewer.'
should be made to stand out more prominently.
```

840928 Made reviewer change notifications more clearly distinguish between the descript...
840963 Copied reviewer email notification fix from main to candidate.
840956 Fixed an issue where email notifications for changes to reviewers did not clea...

Details | Comments 1

Status	Closed
Type	BUG
Severity	B
Priority	undefined
Difficulty	undefined
Subsystem	swarm
Release	2014.2
Owned By	slord
Reported By	slord
Modified By	p4dtguser
Reported Date	2 years ago
Modified Date	2 years ago
Commit Release	2014.2p1
Fix Verified By	QA
P4 Blog	2014-05-02 alau: Verified on version 2014.2/841040
DTG FIXES	840963 840956
DTG DTISSUE	SW-1933
DTG MAPID	jiraprod-p4dprod-swarm
JIRA Summary	The details about edits to reviewers are hard to differentiate

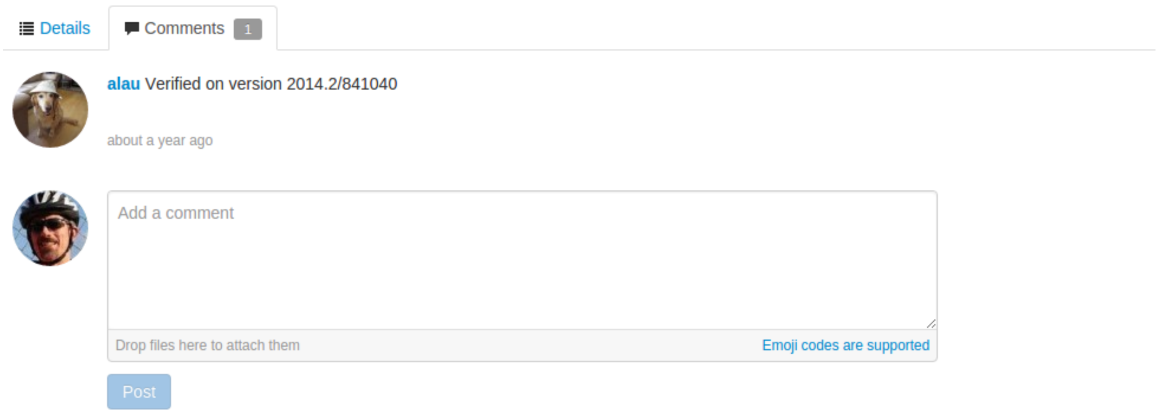
2014-05-02T15:53:44-07:00

The upper portion of the job presentation includes:

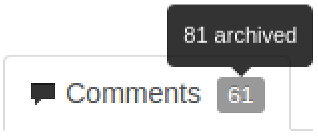
- The avatar and userid of the user that created the job
- The job's creation time
- If changes have been made to the job, the modifying userid and time
- A status indicator
- The job's description
- If any changelists have been submitted that *fix* the job, a list of those changelists and their descriptions. Each associated changelist includes an icon to represent their type: (review), (pending), (committed)

The lower portion of the job presentation lists all of the keys configured in your Helix server's jobspec. Swarm inspects the jobspec and enhances the presentation of fields it recognizes. For example, date fields display as *time ago*, and links are created for userids.

Click the "Comments" on page 143 tab to view any comments added to the job, or to add a comment.


Details
Comments 1
alau Verified on version 2014.2/841040
about a year ago
Add a comment
Drop files here to attach them
Emoji codes are supported
Post

Adding a comment sends a notification. The **Comments** tab displays the number of open comments associated with the job. If you hover your mouse over the comment count, a tooltip is displayed showing how many comments are archived:



For more information on customizing job specifications, see "Job Specifications" in *Helix Versioning Engine Administrator Guide: Fundamentals*.

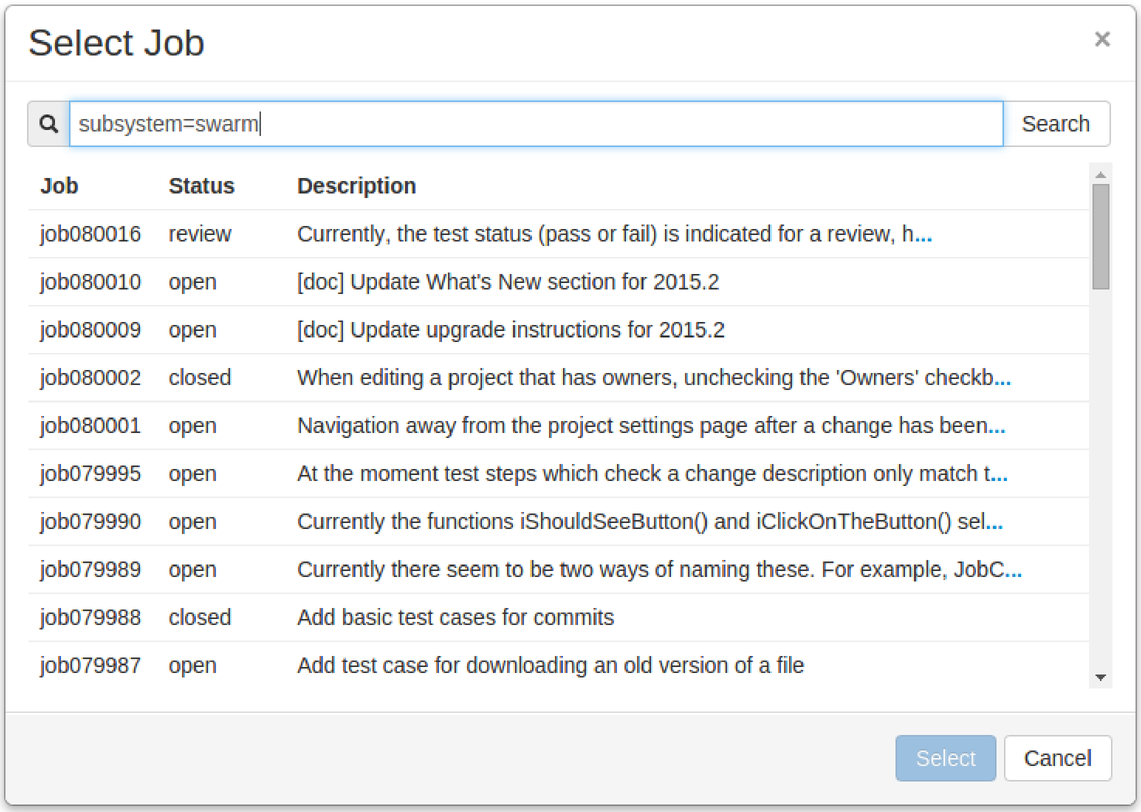
Note
The default Helix Core job specification contains very few fields. Adding fields to record additional information, such as the modification time and userid, reporting time and userid, can assist Swarm use appropriate terminology when describing the current disposition of a job.

Adding jobs

Swarm does not provide the ability to create new jobs in the Helix server, but jobs can be added to changelists or reviews:

1. Navigate to a changelist or review.
2. Click the **Add Job** link ➕ Add Job.

3. Scroll through the available jobs, or enter job search criteria to search available jobs.


Select Job

subsystem=swarm

Job	Status	Description
job080016	review	Currently, the test status (pass or fail) is indicated for a review, h...
job080010	open	[doc] Update What's New section for 2015.2
job080009	open	[doc] Update upgrade instructions for 2015.2
job080002	closed	When editing a project that has owners, unchecking the 'Owners' checkb...
job080001	open	Navigation away from the project settings page after a change has been...
job079995	open	At the moment test steps which check a change description only match t...
job079990	open	Currently the functions iShouldSeeButton() and iClickOnTheButton() sel...
job079989	open	Currently there seem to be two ways of naming these. For example, JobC...
job079988	closed	Add basic test cases for commits
job079987	open	Add test case for downloading an old version of a file

For more information on job search criteria, see "Jobs" in *Helix Versioning Engine User Guide*.

4. If you find the job you want to add, click its row to highlight it and then click **Select**. Or, double-click the desired job to add it.
5. If you do not find the appropriate job, click **Cancel**.

Note
If you attempt to add a job to a review that affects a single project, Swarm applies the project's job view filter to display only jobs that affect the project. It is not currently possible to expand the filter to include jobs outside of the project.

Unlinking jobs

Swarm does not provide the ability to delete jobs from the Helix server, but jobs can be unlinked from changelists or reviews:

1. Navigate to a changelist or review that has an associated job.
2. Click the **X** button beside the job.

3. When prompted for confirmation, click **Unlink** to unlink the job.

Changelists

Changelists are the basic unit of versioning work in Helix Core. A changelist is a list of files, their revision numbers, and the changes made to those files. Commits is a shorter synonym used throughout Swarm.

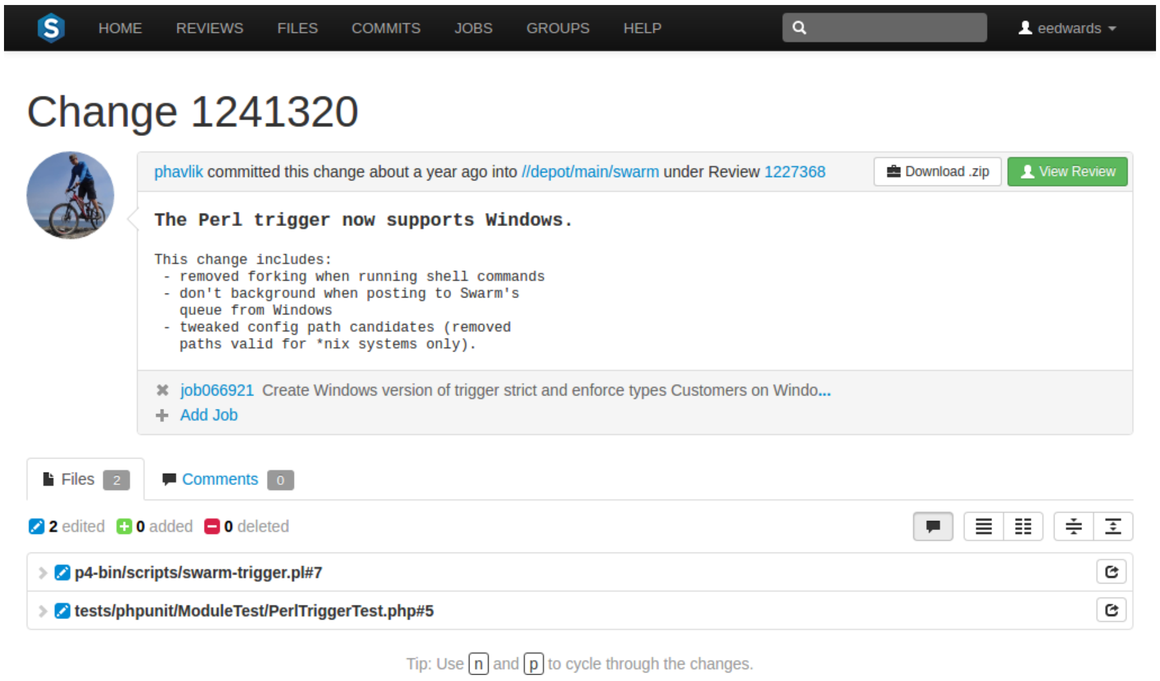
More information is available here:

- Browsing changelists
- Browsing a user's shelved changelists
- Swarm's internal use of changelists to facilitate code reviews

Changelist Display

View a specific *changelist* by clicking on a linked changelist number, or by visiting the URL: `https://myswarm.url/changes/changelist number`

When Swarm displays a change, the presentation is similar to:



Change 1241320

phavlik committed this change about a year ago into //depot/main/swarm under Review 1227368

```
The Perl trigger now supports Windows.

This change includes:
 - removed forking when running shell commands
 - don't background when posting to Swarm's
   queue from Windows
 - tweaked config path candidates (removed
   paths valid for *nix systems only).
```

job066921 Create Windows version of trigger strict and enforce types Customers on Windo...

2 edited 0 added 0 deleted

p4-bin/scripts/swarm-trigger.pl#7

tests/phpunit/ModuleTest/PerlTriggerTest.php#5

Tip: Use n and p to cycle through the changes.

The changelist display includes:

- The avatar and userid of the user who made the change
- The time the change was made
- The common depot location containing all the files included in the change

- When the `zip` command-line tool is available, Swarm can provide a ZIP archive containing all of the files in a changelist.

  Note
  The **Download .zip** button does not appear if the `zip` command-line tool is not available. Similarly, the **Download .zip** button is disabled if the changelist contains only deleted files.

  When you click the **Download .zip** button, Swarm performs the following steps:

  1. Scan the changelist's files, to determine if you have permission to access their contents (according to the Helix server protections), and if the total file size is small enough to be processed by Swarm.
  2. Sync the file contents to the Swarm server from the Helix server.
  3. Create the ZIP archive by compressing the file content.
  4. Start a download of the generated ZIP archive.

  You might not see all of the above steps; Swarm caches the resulting ZIP archives so that repeated requests to download the same changelist's files can skip the sync/compress steps whenever possible.

  If an error occurs while scanning, syncing, or compressing, Swarm indicates the error.

  For information on the configuration for ZIP archives, see "Archives configuration" on page 240.

- If the change was involved in a code review, a link to the review along with a **View Review** button.
- The description of the change
- A list of jobs that this change *fixes*, if any. You can add jobs and unlink jobs here.
- The list of files included in the change, including any folders between the common depot location and the file, and the file's version number.
- A tab to review any comments made regarding the change, or any of its files.

Each file is presented in a diff display, showing you whether the file was added, modified, or deleted. For text-based and image files, Swarm can display any changes made within the file. For changes with only a single file, the diff display is the default; otherwise each file is listed. Click the filename to see the diff display. See "Diffs" on the facing page for more information.

The **Request Review** button indicates the current state of this change; no Review record has been created. Clicking **Request Review** starts a code review for this change. For more information, see "Start a review" on page 221.

Important
If your Helix server is configured as a commit-edge deployment, and your normal connection is to an edge server, Swarm refuses to start reviews for shelved changes that have not been promoted to the commit server.

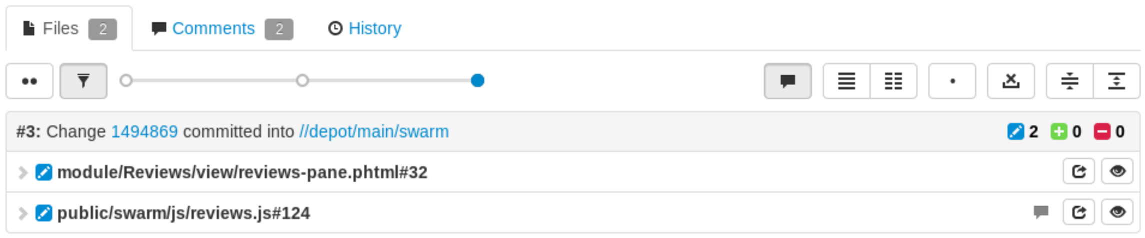
Within Swarm, this means that the **Request Review** button does not appear for unpromoted shelved changes. Outside of Swarm, attempts to start reviews for unpromoted shelved changelists appear to do nothing. Ask your Helix server administrator for assistance if you cannot start a review.

An administrator of the Helix server can automatically promote shelved changes to the commit server by setting the configurable `dm.shelve.promote` to 1.

When a file in a changelist has one or more associated comments, an icon appears near the far right of the file's entry.

Diffs

When you view a changelist or code review, the associated files are presented as *diffs*, short for differences, showing you how they have changed.

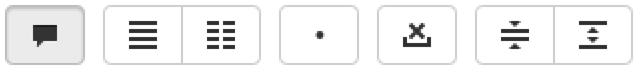


#3: Change 1494869 committed into //depot/main/swarm 2 0 0

module/Reviews/view/reviews-pane.phtml#32

public/swarm/js/reviews.js#124

The first row of buttons above the files allow you to (left to right):

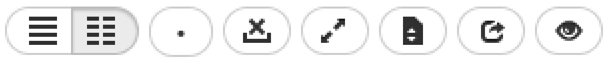


- Toggle the display of comments
- Show all diffs as inline
- Show all diffs as side-by-side
- Toggle show whitespace for all files
- Toggle ignore whitespace in diffs
- Collapse all files
- Expand all files. By default this button is disabled if there are more than 10 files in the review. For details, see Expand All Limit

Each file is presented with an icon indicating whether the file was:

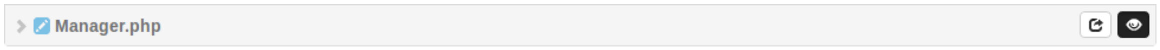
- Added/branched/imported
- Edited/integrated
- Deleted

The file's presentation can be controlled with (left to right):



- **Show In-line** button, which highlights line additions, modifications, and removals in a single pane.
- **Show Side-by-Side** button, which highlights additions, modifications, and removals in two panes, with the older version of the file on the left, and the newer version on the right.
- **Show Whitespace** button, which makes whitespace characters more visible; spaces show up as dots, tabs show up as arrows that point to a bar, and line endings show up as down-pointing arrows.
- **Ignore Whitespace** button, which toggles the highlighting of whitespace changes in a file, making it easier to identify non-formatting changes. Normally, whitespace is not ignored; an administrator can change the default. See Diff configuration for details.
- **Show all diffs** button toggles between displaying all the diffs for the file and only the first few. The limit of what are shown by default is configurable by the administrator (see Max Diffs) and defaults to 1500 and if there are fewer than that then this button is hidden (edited and integrated files only).
- **Show Full Context** button toggles between displaying only the portions of the file that have changed and the full file (edited and integrated files only).
- **Show File** button, displayed only for edited or integrated files, opens a new browser tab/window display the full file (where possible), provide access to its history, and a button to download the file.
- **Mark as Read** button, displayed only for code reviews, helps you (and others) keep track of which files have been reviewed. This is particularly useful when a code review consists of many files.

  When clicked, the button's colors invert and the associated file is visually muted, to make it easy to distinguish read files from unread files:

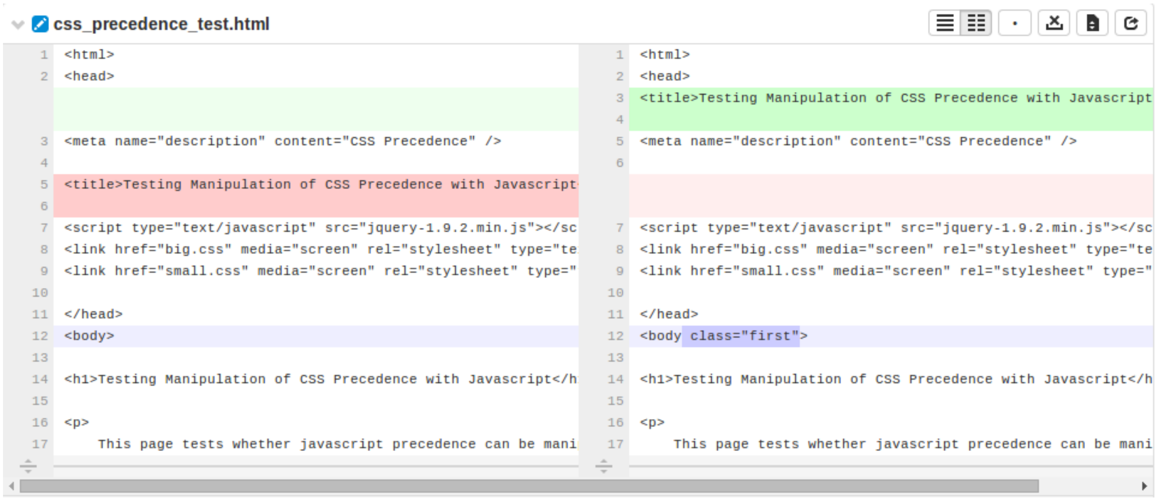
  

  If a file has been marked as read, click the button a second time to reset the status to unread.

If the file has comments associated, its entry shows a comments exist icon.

Viewing a diff

When you view a diff, the changes are highlighted:


css_precedence_test.html

```
<html>
<head>

<meta name="description" content="CSS Precedence" />

<title>Testing Manipulation of CSS Precedence with Javascript

<script type="text/javascript" src="jquery-1.9.2.min.js"></sc
<link href="big.css" media="screen" rel="stylesheet" type="te
<link href="small.css" media="screen" rel="stylesheet" type="

</head>
<body>

<h1>Testing Manipulation of CSS Precedence with Javascript</h

<p>
    This page tests whether javascript precedence can be mani
```

```
<html>
<head>
<title>Testing Manipulation of CSS Precedence with Javascript

<meta name="description" content="CSS Precedence" />


<script type="text/javascript" src="jquery-1.9.2.min.js"></sc
<link href="big.css" media="screen" rel="stylesheet" type="te
<link href="small.css" media="screen" rel="stylesheet" type="

</head>
<body class="first">

<h1>Testing Manipulation of CSS Precedence with Javascript</h

<p>
    This page tests whether javascript precedence can be mani
```

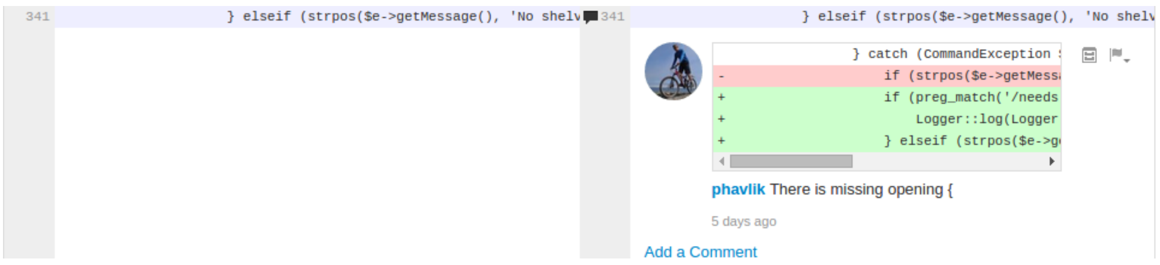
- Red indicates lines that have been removed
- Blue indicates lines that have been modified
- Green indicates lines that have been added

The diff presentation displays a concise view of where changes are made within a file, showing the changed lines and only a few lines before and after each revision. Sometimes, this concise view is insufficient to understand the context of the change. The **Show Full Context** displays the entire file, but this can be too much detail.

When there are more lines in a file, either above or below the currently displayed diff, a **Show More Context** row appears.

Clicking this row attempts to display 10 additional lines adjacent to the row (which may appear between two modifications), allowing up to 20 lines of additional context to be displayed. The row disappears when no additional content is available.

Comments may be presented within the body of a file and appear immediately below the line the commenter targeted for comment. See "Comments" on the next page for more details.



```
341             } elseif (strpos($e->getMessage(), 'No shelv
```

```
341             } elseif (strpos($e->getMessage(), 'No shelv
```

```
          } catch (CommandException :
-             if (strpos($e->getMessa
+             if (preg_match('/needs
+                 Logger::log(Logger
+             } elseif (strpos($e->g
```

phavlik There is missing opening {

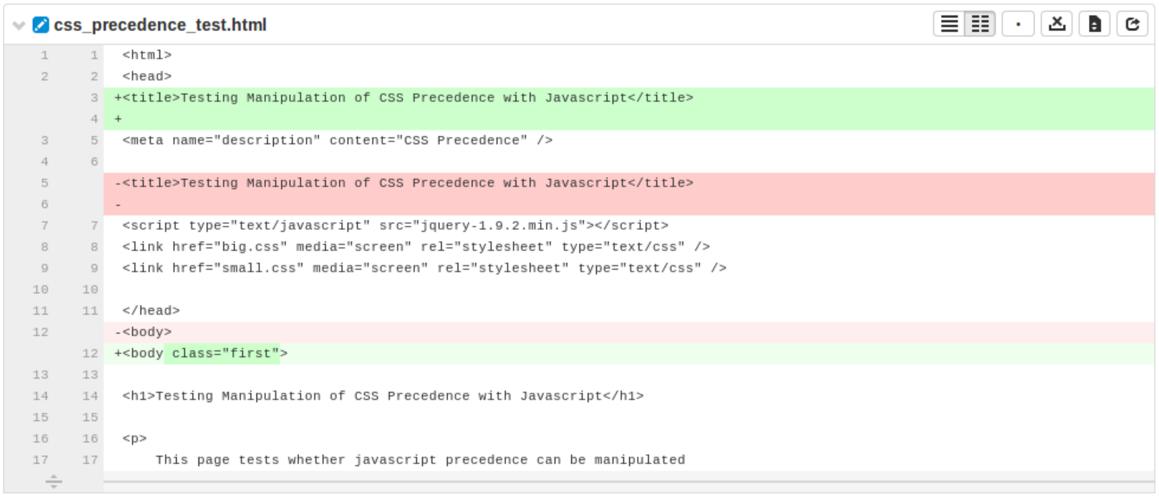
5 days ago

Add a Comment

Note

The *comment(s) exist here* icon appears in the line number column whenever comments exist. This is useful when the comment display is toggled off.

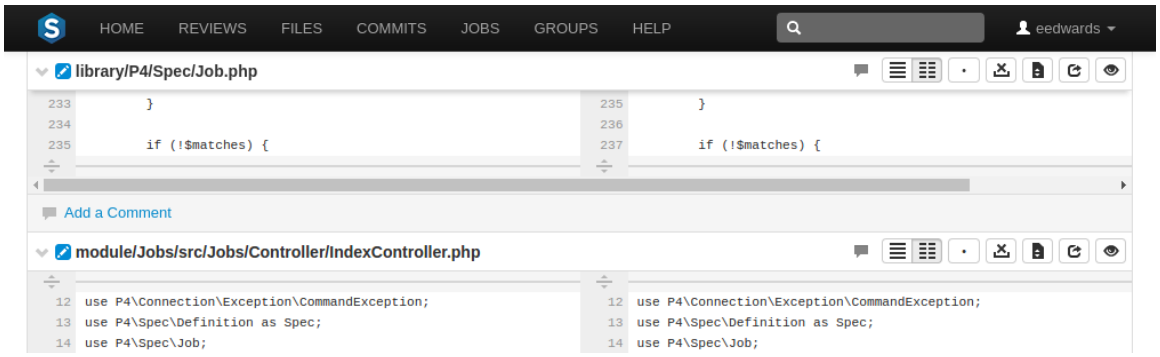
When viewing a diff in-line, the line numbers for the old version are first, and the line numbers for the new version are second, followed by the file content. Some users find this view easier to use when locating an area that has changed, but then switch to side-by-side view to help them understand the change better. Swarm maintains the scroll position; you do not lose your place in the file after toggling the diff view.



```
 <html>
 <head>
+<title>Testing Manipulation of CSS Precedence with Javascript</title>
+
 <meta name="description" content="CSS Precedence" />

-<title>Testing Manipulation of CSS Precedence with Javascript</title>
-
 <script type="text/javascript" src="jquery-1.9.2.min.js"></script>
 <link href="big.css" media="screen" rel="stylesheet" type="text/css" />
 <link href="small.css" media="screen" rel="stylesheet" type="text/css" />

 </head>
-<body>
+<body class="first">

 <h1>Testing Manipulation of CSS Precedence with Javascript</h1>

 <p>
     This page tests whether javascript precedence can be manipulated
```

Note
Press **n** on your keyboard to scroll to the next changed area within a file. Press **p** to scroll to the previous change.

When a diff contains multiple files, the changes may be taller than your browser window. Swarm keeps the per-file toolbar in view for each file as you scroll, so that you can continue to identify the file at the top of the screen and control its presentation.



```
        }                                   }

        if (!$matches) {                    if (!$matches) {
```

```
use P4\Connection\Exception\CommandException;      use P4\Connection\Exception\CommandException;
use P4\Spec\Definition as Spec;                     use P4\Spec\Definition as Spec;
use P4\Spec\Job;                                    use P4\Spec\Job;
```

Comments

Comments are the primary feedback mechanism provided by Swarm. Comments can be made on files in a review, on any lines of any text file in a review, and on jobs.

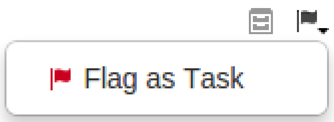
Access the comments for a review, or job, by clicking the **Comments** tab. The number of open (non-archived comments) is displayed in the tab.

If you hover your mouse over the comment count, a tooltip is displayed showing how many comments are archived. See "Archiving comments" on page 151 for details.

Tasks

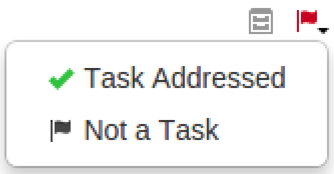
Flagging comments as *tasks* is a lightweight workflow within a review that helps authors and reviewers prioritize review feedback. Any comment on a code review can be flagged as a task, indicating to a code review's author that the described issue needs to be addressed, and that the review is unlikely to be approved without a fix.

To flag a comment as a task, check the **Flag as Task** checkbox when posting a comment, or click the flag icon in the upper right of an existing comment and select **Flag as Task** in the drop-down menu.



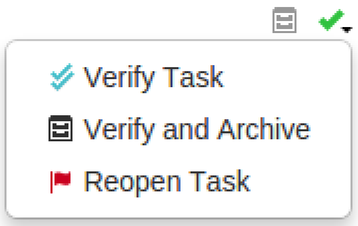
Note
If you do not have permission to archive comments, you do not have permission to flag comments as tasks. Anonymous users never have permission to archive comments, and can only view current task states.

Once a comment is flagged as a task, it is considered to be an *open task*. Clicking the red flag icon displays a drop-down menu with the following options:



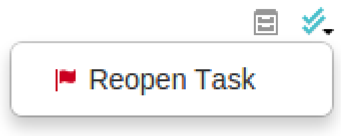
- **Task Addressed**: usually used by the review's author to indicate that the noted issue has been fixed.
- **Not a Task**: used to correct comments that have inappropriately been flagged as tasks.

A comment with a green check indicates that the task has been addressed. Clicking the green check displays a drop-down menu with the following options:



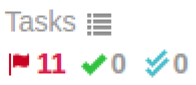
- **Verify Task**: usually used by the comment author, or other reviewer, after checking that the issue is indeed fixed.
- **Verify and Archive**: used to both indicate that the issue has been fixed, and to archive the comment so it is hidden from view. Archived tasks, whether they are open, addressed, or verified, are not included in the task counts for the code review.
- **Reopen Task**: used when an update to the review does not resolve the noted issue, or to correct an inadvertent task verification.

A comment with a blue double-check indicates that the task has been verified. Clicking the blue double-check displays a drop-down menu with the following option:

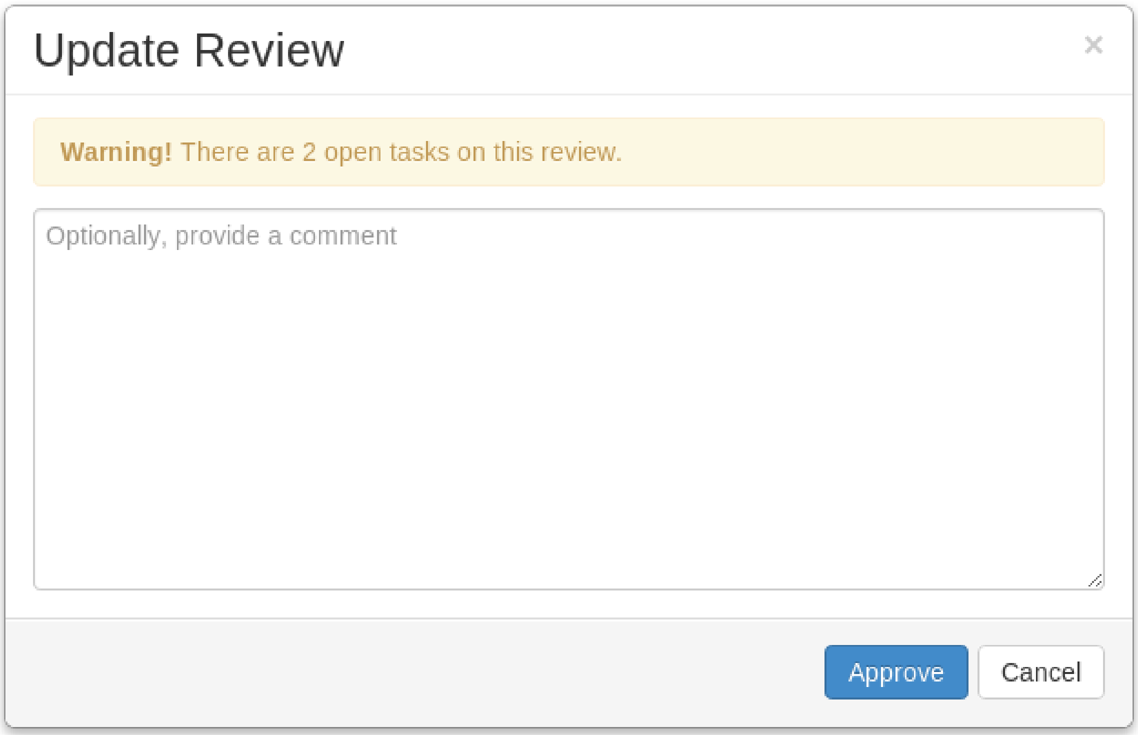


- **Reopen Task**: used if the issue needs further work post-verification, or if verification was made inadvertently.

A summary of comments flagged as tasks and their various states appears below the code review's description, beside the reviewers area. Archived comments that are flagged as tasks are not included in the summary. See "Tasks" on page 213 for more details.



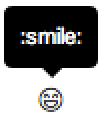
Like many of Swarm's features, tasks are merely advisory. Flagging a comment as a task provides visual indication that there is an identified issue, but Swarm does not restrict any operations for tasks in any state. A warning is displayed should you attempt to approve or commit a review via the Swarm UI that has any open tasks (that are not archived):


Update Review
×
Warning! There are 2 open tasks on this review.
Optionally, provide a comment
Approve
Cancel

Comment features

Emoji

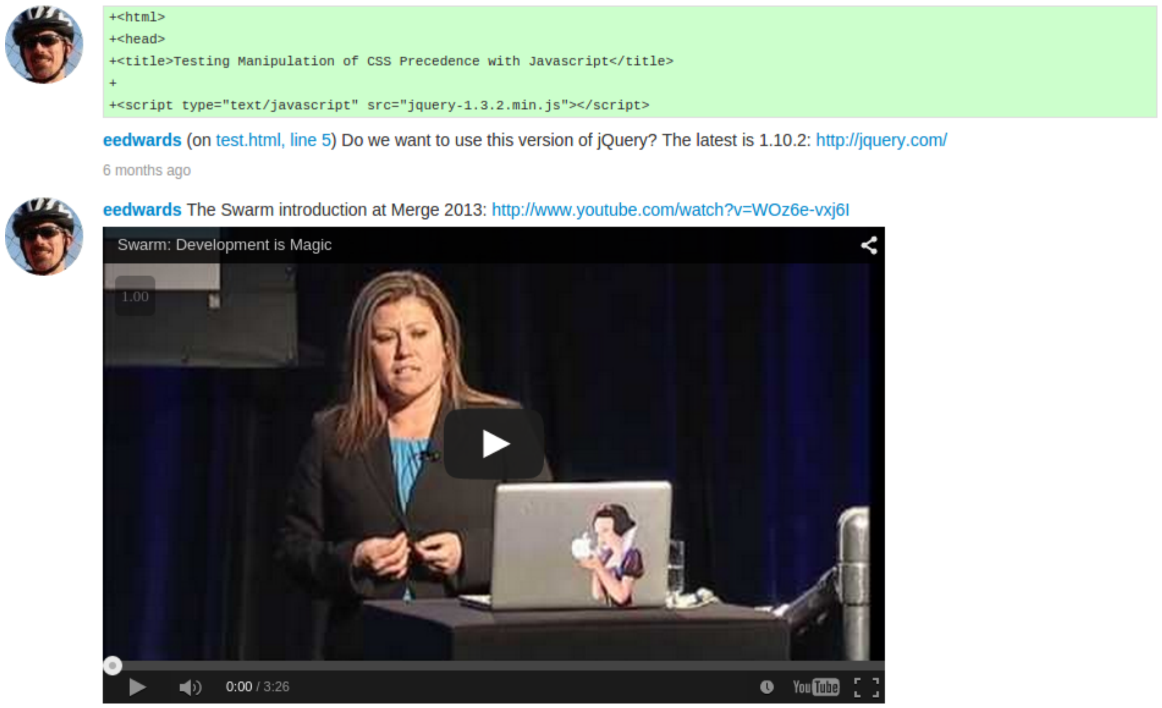
Swarm comments support Emoji short-hand; when you save a comment, emoticon text like `:smile:` is displayed as: 😄. Hover your mouse over an Emoji emoticon to see a tooltip displaying the text short-hand for the emoticon.



For more information on Emoji, see Emoji at Wikipedia. Emoji emoticons are listed in the Emoji Cheat Sheet.

Links in comments

Whenever you include a URL in a comment, it is automatically made into a link. If the link points to an image, or a YouTube video, that resource is displayed at the end of the comment.



```
+<html>
+<head>
+<title>Testing Manipulation of CSS Precedence with Javascript</title>
+
+<script type="text/javascript" src="jquery-1.3.2.min.js"></script>
```

eedwards (on test.html, line 5) Do we want to use this version of jQuery? The latest is 1.10.2: http://jquery.com/

6 months ago

eedwards The Swarm introduction at Merge 2013: http://www.youtube.com/watch?v=WOz6e-vxj6I

Liking comments

As an authenticated user, you can *like* a comment by clicking the muted heart icon beneath a comment about 16 hours ago · ♡.

When you like a comment, a notification is sent to the comment's author, and the heart icon becomes red to indicate that you have liked the comment about 16 hours ago · 1 ♥.

If a comment has one or more likes, a count of the likes appears before the heart icon. A tooltip appears when you hover the mouse over the like count displaying the usernames of everyone that has liked the comment.

Click the heart icon again to unlike a comment.

Note
You cannot like/unlike a comment that has been archived.

Comment attachments

Arbitrary files can be attached to comments. This is useful for sharing documents that are helpful in code review, such as screenshots of error conditions, reference code, etc.

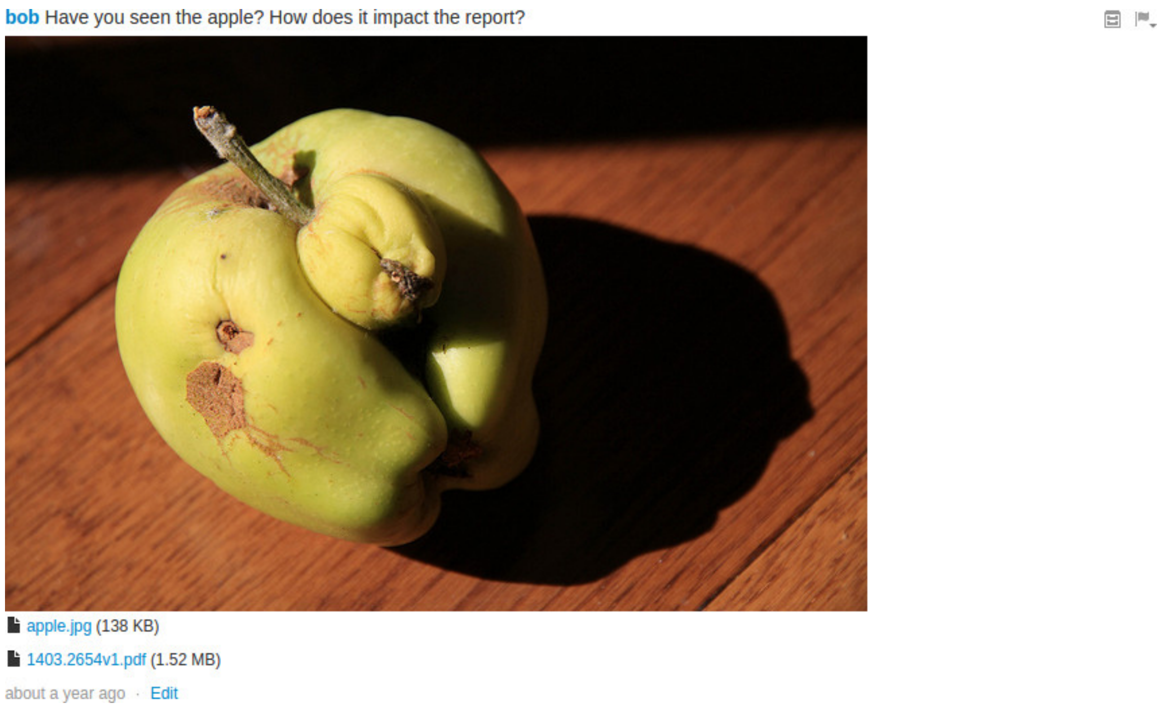
Swarm must be configured to enable comment attachments. Once the configuration is complete, the comment area adds a note **Drop files here to attach them**.

To attach a file, simply drag it from your desktop or file browser and drop the file onto the comment area. Multiple files can be attached to a comment, either one at a time, or by dragging a group of files in one go. However, uploading a folder of files is not supported.

When you let go of the file (or files), the upload to Swarm starts immediately, and a progress bar for each file is displayed below the comment area with green bars indicating completed uploads and blue bars indicating uploads in progress:

Before you post your comment, if any files already attached should be removed, click the **X** to the right of the attached file you wish to remove.

After you post your comment, a list of attached files appears below the comment text. The list includes the filename and file sizes for each attached file. When you view your comment on the Comments tab, Swarm displays an image preview for any attached images.


bob Have you seen the apple? How does it impact the report?

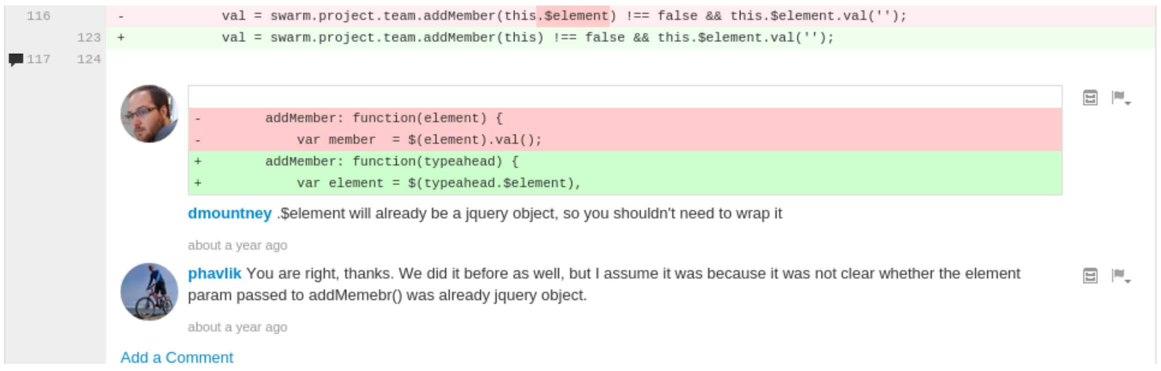
apple.jpg (138 KB)

1403.2654v1.pdf (1.52 MB)

about a year ago · Edit

Comment context

When comments are added to files in a review, on lines that have been changed, Swarm records several lines of context before the line receiving the comment. This helps makes sense of the comments should later changes remove those lines.



```
-        val = swarm.project.team.addMember(this.$element) !== false && this.$element.val('');
+        val = swarm.project.team.addMember(this) !== false && this.$element.val('');
```

```
-        addMember: function(element) {
-            var member  = $(element).val();
+        addMember: function(typeahead) {
+            var element = $(typeahead.$element),
```

dmountney .$element will already be a jquery object, so you shouldn't need to wrap it

about a year ago

phavlik You are right, thanks. We did it before as well, but I assume it was because it was not clear whether the element param passed to addMemebr() was already jquery object.

about a year ago

Add a Comment

Each comment associated with that line has a record of the context, but only the first comment displays that context.

Delayed notifications

Delayed notifications allow reviewers to add or edit comments as they progress through a review, but prevent notifications from being sent until their review effort is complete. Any delayed notifications are rolled up into a single email notification when the reviewer turns off delayed notifications.

Whenever **Delay Notification** is checked, *batch* mode is enabled and the notification for any new or edited comment is delayed; no email is sent. Swarm remembers that you have clicked **Delay Notification**; you do not need to click it for each comment you post.

When there are delayed notifications, Swarm displays their count.

Batch mode ends when a new or edited comment is saved with **Delay Notification** unchecked. Swarm then sends a single email notification containing all previously delayed comment notifications plus the most recent new or edited comment.

Note
There is no *expiry* mechanism; delayed notifications are delayed indefinitely until **Delay Notification** is unchecked.

Hover your mouse over **Delay Notification** to see a tooltip that indicates the current state.

Note
Delayed notifications are available only for comments on reviews; comments on commits or jobs produce notifications immediately.

Commenting on a changelist or review

1. Visit the changelist's or review's page.
2. Click **Comments** to view the **Comments** tab.

3. Add your comment in the provided text area.

   Also see "Comment attachments" on page 147.
4. Click **Post**.

Commenting on a specific line in a file

1. Visit the changelist's or review's page.
2. Click **Files** to view the **Files** tab.
3. Click a line you want to comment on. The comment text area appears.
4. Add your comment in the provided text area.
5. Click **Post**.

Commenting on a file in a changelist or code review

1. Visit the changelist's or review's page.
2. Click **Files** to view the **Files** tab.
3. If there are multiple files, click the file you want to comment on to expand its view.
4. Click the **Add a Comment** link in the footer of the file's display.
5. Add your comment in the provided text area.
6. Click **Post**.

Note
You can use "@mention" on page 171 in comments. An @mention includes the specified user in the review, and they will receive a notification whenever there is an update to the review.

Editing comments

For any comment that you have created, you can edit its contents:

1. Click the **Edit** link.
2. Adjust the comment content, including adding new attachments or removing existing attachments.

   When you attempt to remove an attachment, it is not immediately removed but marked for removal; the attachment's presentation becomes muted and the removal **X** icon is replaced with a **Restore** icon.

   Before completing your edits to the comment, if you do not wish to remove attachments that are marked for removal, click the **Restore** icon; the **Restore** icon is replaced with the removal **X** icon, and the attachment's presentation is no longer muted.
3. Click **Save**. The edits to the comment text are saved, any new attachments are saved, and any attachments marked for removal are removed. Swarm also adds (edited) to the comment's timestamp.

   At any time, you can click **Cancel** to cancel editing of the comment. The original comment text is reinstated and any existing attachments are retained. Any text edits or uploaded attachments are lost.

Whenever a comment is edited, Swarm sends a notification to everyone involved in the review, including the author and reviewers, but not to the editor of the comment.

Note
Swarm does not provide a mechanism to see older versions of edited comments.

Archiving comments

As a code review progresses, comments made on earlier versions of a file may no longer be pertinent. If left unattended, comments may accumulate until it is difficult to determine which comments are still relevant and which comments have been addressed.

- Click the *file drawer* icon at the top right of a posted comment to archive it.

Archived comments are normally hidden from view. When archived comments exist, a box appears showing the number of archived comments. Click the box to toggle the presentation of all archived comments.

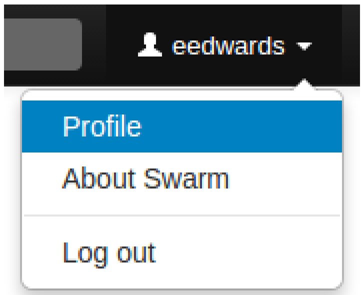
Restore comments

A closed comment may be restored.

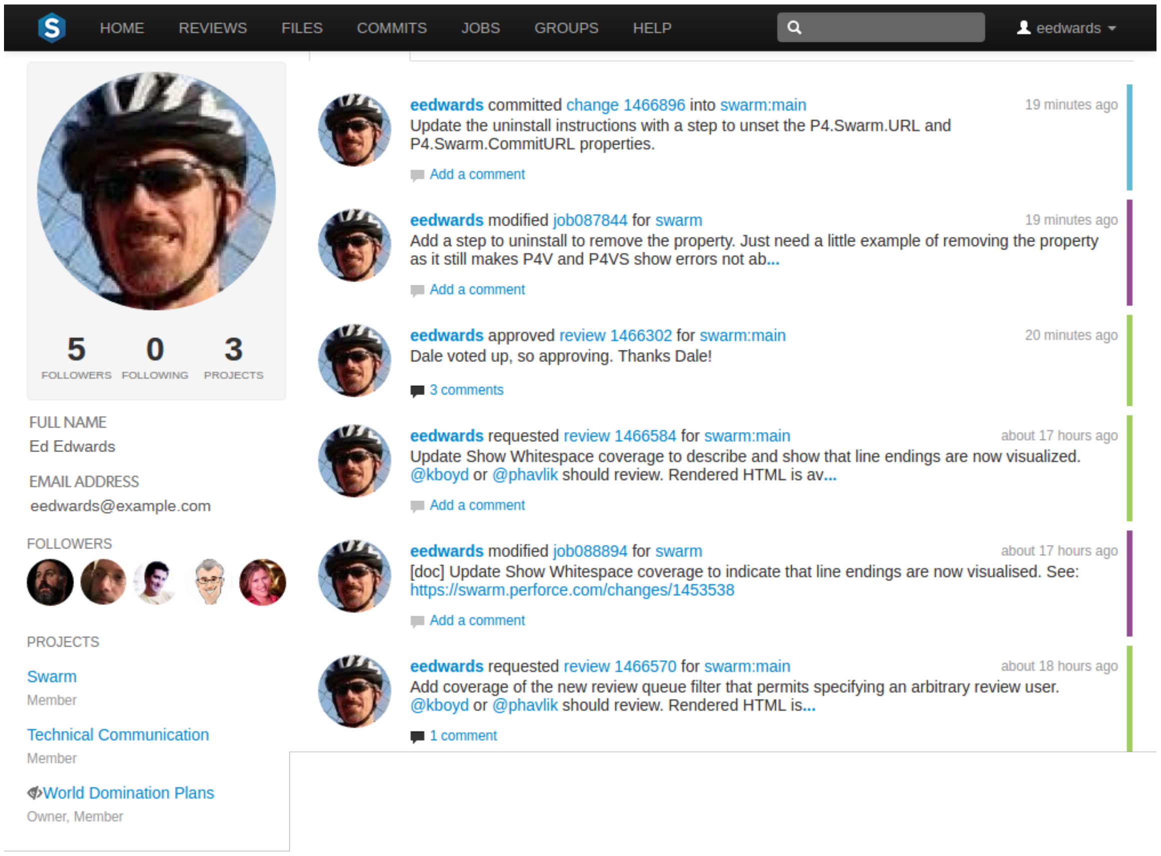
1. Click the *archived comments* box to display all archived comments.
2. Locate the comment you want to restore.
3. Click the arrow icon in the top right of the comment to restore it.

Users

Swarm's users are based on the users configured in the Helix server. Display your own user profile when you are logged in by clicking on your userid at the top-right of the main toolbar and selecting **Profile**.



The user profile page displays your avatar, your full name, your email address, the list of users following your activity, the users or projects that you are following, and the list of projects of which you are an owner or member, including an *eye* icon ( ) beside any private projects you belong to. An activity stream for events you have created is also presented.



5 FOLLOWERS 0 FOLLOWING 3 PROJECTS

FULL NAME
Ed Edwards

EMAIL ADDRESS
eedwards@example.com

FOLLOWERS

PROJECTS
Swarm
Member
Technical Communication
Member
World Domination Plans
Owner, Member

eedwards committed change 1466896 into swarm:main — 19 minutes ago
Update the uninstall instructions with a step to unset the P4.Swarm.URL and P4.Swarm.CommitURL properties.
Add a comment

eedwards modified job087844 for swarm — 19 minutes ago
Add a step to uninstall to remove the property. Just need a little example of removing the property as it still makes P4V and P4VS show errors not ab...
Add a comment

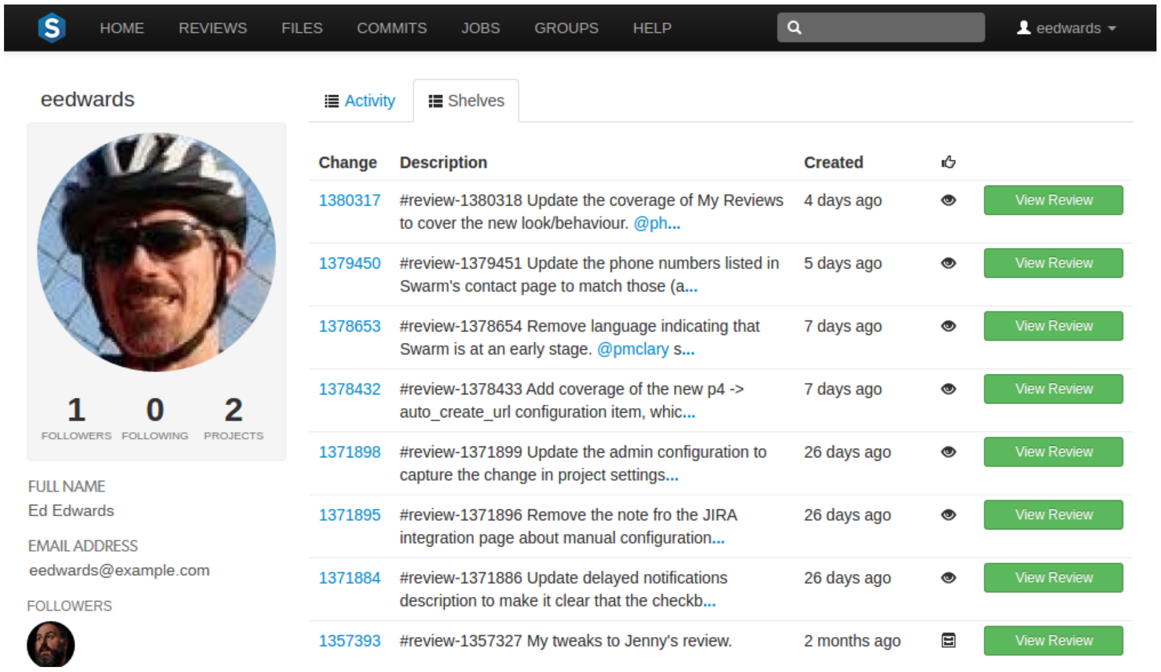
eedwards approved review 1466302 for swarm:main — 20 minutes ago
Dale voted up, so approving. Thanks Dale!
3 comments

eedwards requested review 1466584 for swarm:main — about 17 hours ago
Update Show Whitespace coverage to describe and show that line endings are now visualized. @kboyd or @phavlik should review. Rendered HTML is av...
Add a comment

eedwards modified job088894 for swarm — about 17 hours ago
[doc] Update Show Whitespace coverage to indicate that line endings are now visualised. See: https://swarm.perforce.com/changes/1453538
Add a comment

eedwards requested review 1466570 for swarm:main — about 18 hours ago
Add coverage of the new review queue filter that permits specifying an arbitrary review user. @kboyd or @phavlik should review. Rendered HTML is...
1 comment

Click the **Shelves** tab to display a list of the user's shelved changelists.



Change	Description	Created		
1380317	#review-1380318 Update the coverage of My Reviews to cover the new look/behaviour. @ph...	4 days ago		View Review
1379450	#review-1379451 Update the phone numbers listed in Swarm's contact page to match those (a...	5 days ago		View Review
1378653	#review-1378654 Remove language indicating that Swarm is at an early stage. @pmclary s...	7 days ago		View Review
1378432	#review-1378433 Add coverage of the new p4 -> auto_create_url configuration item, whic...	7 days ago		View Review
1371898	#review-1371899 Update the admin configuration to capture the change in project settings...	26 days ago		View Review
1371895	#review-1371896 Remove the note fro the JIRA integration page about manual configuration...	26 days ago		View Review
1371884	#review-1371886 Update delayed notifications description to make it clear that the checkb...	26 days ago		View Review
1357393	#review-1357327 My tweaks to Jenny's review.	2 months ago		View Review

A *shelved* changelist is a pending changelist that has a copy of one or more files from within the changelist stored on the server. Shelved files are not versioned: if you update the shelved files, the update replaces any existing files on the changelist's shelf.

Note
Swarm can use multiple shelved changes to record the history of reviews. See "Internal representation" on page 200 for details.

Swarm uses shelved changelists as the basis of its code review feature. However, not all shelved changelists are reviews. Users may shelve files for other reasons, including ensuring that the Helix server has a copy of work in progress, or as a way to move temporary work from one workspace to another.

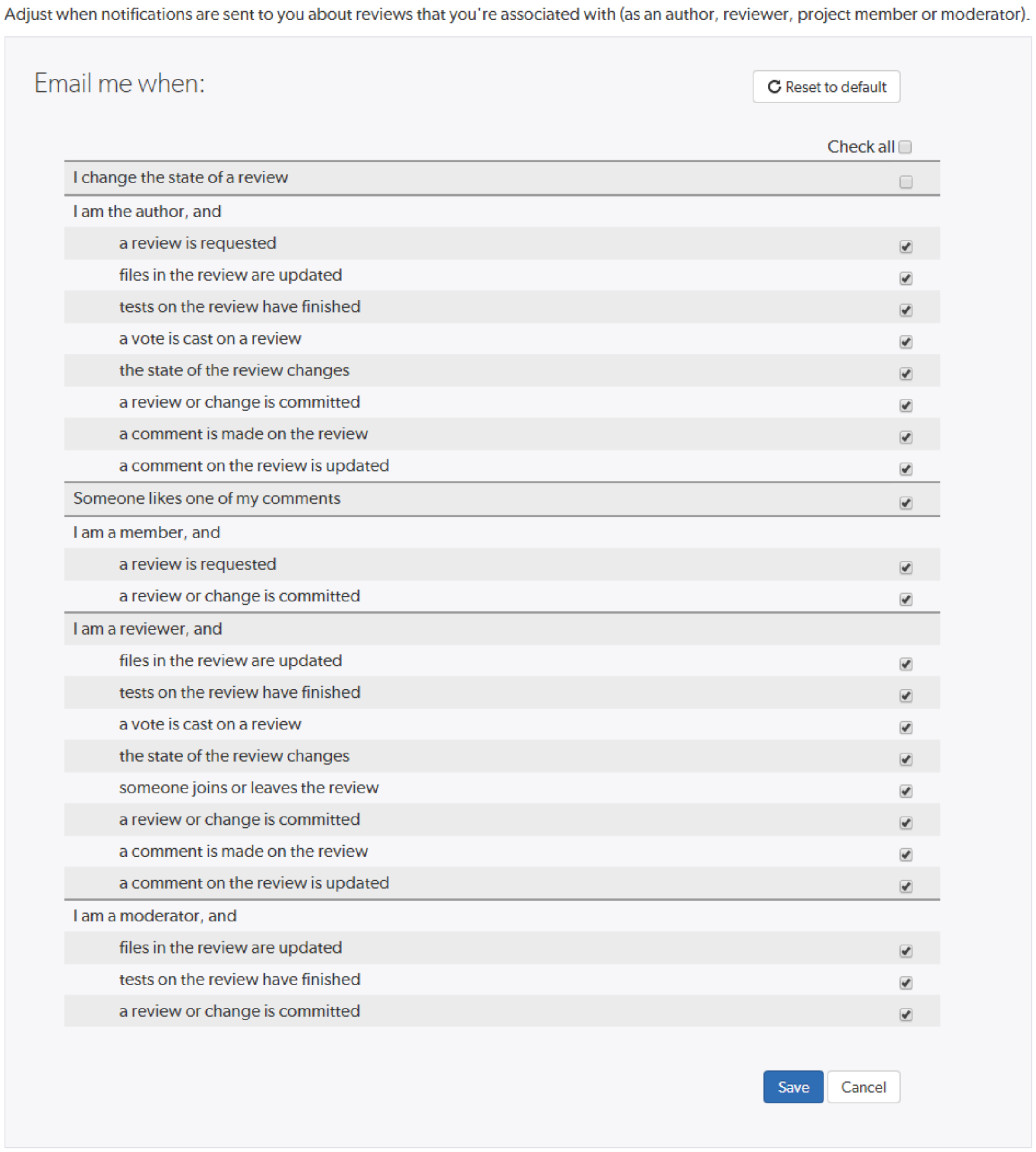
Click **Request Review** to start a Swarm review for any shelved changelist that is not already involved in a review.

Click **View Review** to view the Swarm review associated with shelved changelists when a review has already started.

Settings

The **Settings** tab allows you to configure which notifications you receive when events occur within Swarm. This allows you to limit the number of emails you receive to just those you are interested in. The settings apply across all projects.

Note
Defaults for these options are configured by the Swarm administrator, and they may force some of these options to on or off. See "Global settings" on page 270 for how this is configured.



Adjust when notifications are sent to you about reviews that you're associated with (as an author, reviewer, project member or moderator).

Email me when:

I change the state of a review	☐
I am the author, and	
a review is requested	✓
files in the review are updated	✓
tests on the review have finished	✓
a vote is cast on a review	✓
the state of the review changes	✓
a review or change is committed	✓
a comment is made on the review	✓
a comment on the review is updated	✓
Someone likes one of my comments	✓
I am a member, and	
a review is requested	✓
a review or change is committed	✓
I am a reviewer, and	
files in the review are updated	✓
tests on the review have finished	✓
a vote is cast on a review	✓
the state of the review changes	✓
someone joins or leaves the review	✓
a review or change is committed	✓
a comment is made on the review	✓
a comment on the review is updated	✓
I am a moderator, and	
files in the review are updated	✓
tests on the review have finished	✓
a review or change is committed	✓

Toggle notifications for each event on or off to control whether you receive an email when that event occurs.

Clicking the **Save** button saves these settings. Clicking **Reset to default** resets the options back to system defaults.

Viewing users

View the profile pages of other users displayed anywhere by Swarm by clicking on their avatar or userid, or by visiting the URL: `https://myswarm.url/users/userid`

Note
Currently, Swarm does not provide an overall list of users.

Groups

Groups are a feature of Helix server that makes it easier to manage permissions for users.

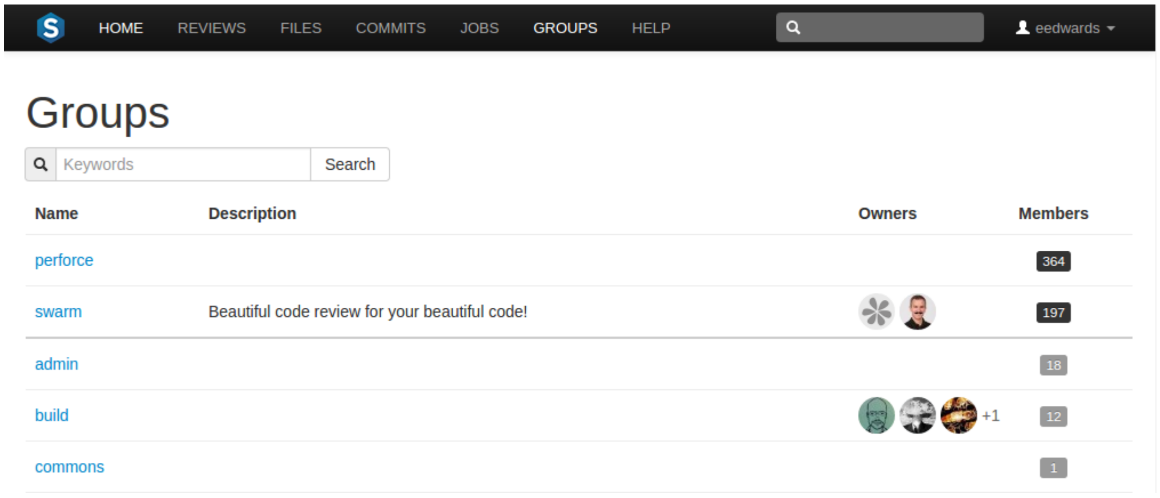
It is important to be aware of certain aspects of Helix server groups:

- Groups can have both users and sub-groups. A user that is a member of a sub-group is also a member of the parent group.
- Groups have owners, but owners are not members of their groups. A group owner can be added as a member of a group.
- Groups can be created by users with *super* privileges in Helix server (`p4d`). If p4d is at version 2012.1 or newer, users with *admin* privileges can also add groups.
- Groups can be edited by their owners, or by users with *super* privileges in Helix server.
- Groups have a separate namespace from users; you can have a user named `fish` that is a member of a group named `fish`.
- Groups can be project moderators
- Groups can be reviewers, see Review display and @@mention for details.
- You can `@@mention` a group in a code review comment to ensure the mentioned group receives notifications of code review events. This also creates a link that displays the group's details when clicked. See @@mention for details.
- Groups cannot be project owners.

More information on adding, editing, and removing groups is included in the "Groups" on page 180 chapter.

Listing groups

Begin browsing groups by clicking the **Groups** link in the main toolbar.



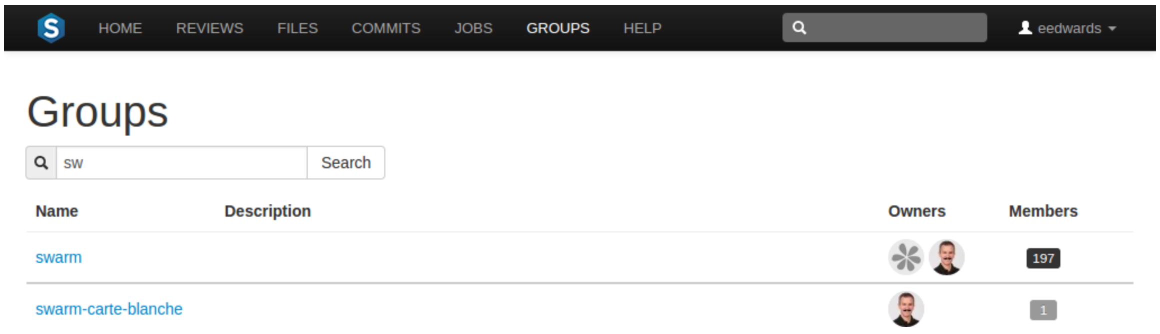
Name	Description	Owners	Members
perforce			364
swarm	Beautiful code review for your beautiful code!		197
admin			18
build		+1	12
commons			1

For each group, the group's name, description, list of owner avatars, and a membership count (which has a darker background when you are a member). Click on a group name to display details for that group.

The groups listing is sorted by multiple criteria:

- Groups that you own, or belong to, are listed first. A thicker row border indicates where your group ownership/membership ends; you are not an owner/member of any group below the thicker border.
- Groups are then sorted by name.

You can search available groups by entering some text in the **Search** field and clicking **Search**. Any group names or descriptions that match the entered text are displayed.



Name	Description	Owners	Members
swarm			197
swarm-carte-blanche			1

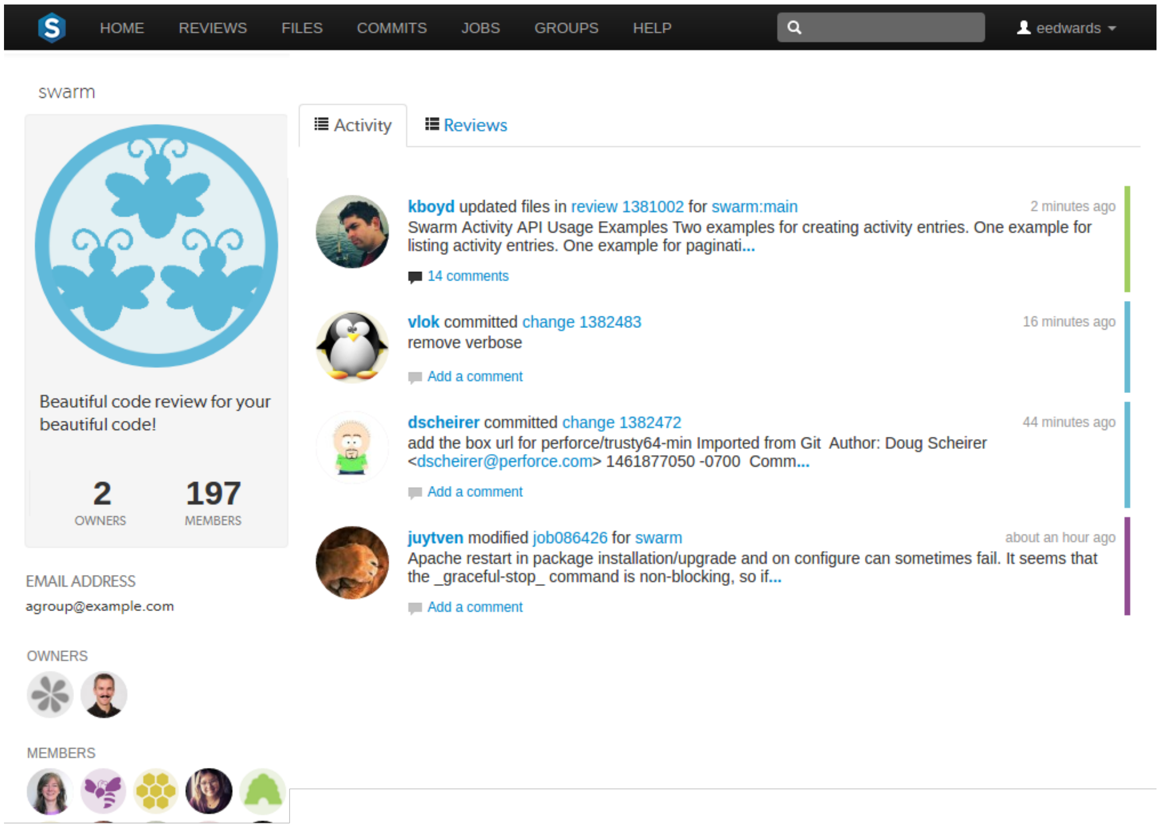
If you have *super* privileges in Helix server (**p4d**), or have admin privileges in **p4d** version 2012.1 or newer, Swarm displays the **Add Group** button. Click **Add Group** to add a new group.

Note
If your Helix server has a large number of groups, the time required to display the list of groups might be notable.

Viewing a group

View a specific group by clicking on a linked group name, or by visiting the URL:
`https://myswarm.url/groups/group-id`

When Swarm displays a group, the presentation is similar to:



Beautiful code review for your beautiful code!

agroup@example.com

kboyd updated files in review 1381002 for swarm:main — 2 minutes ago
Swarm Activity API Usage Examples Two examples for creating activity entries. One example for listing activity entries. One example for paginati...
14 comments

vlok committed change 1382483 — 16 minutes ago
remove verbose
Add a comment

dscheirer committed change 1382472 — 44 minutes ago
add the box url for perforce/trusty64-min Imported from Git Author: Doug Scheirer <dscheirer@perforce.com> 1461877050 -0700 Comm...
Add a comment

juytven modified job086426 for swarm — about an hour ago
Apache restart in package installation/upgrade and on configure can sometimes fail. It seems that the _graceful-stop_ command is non-blocking, so if...
Add a comment

The information presented in the group display includes:

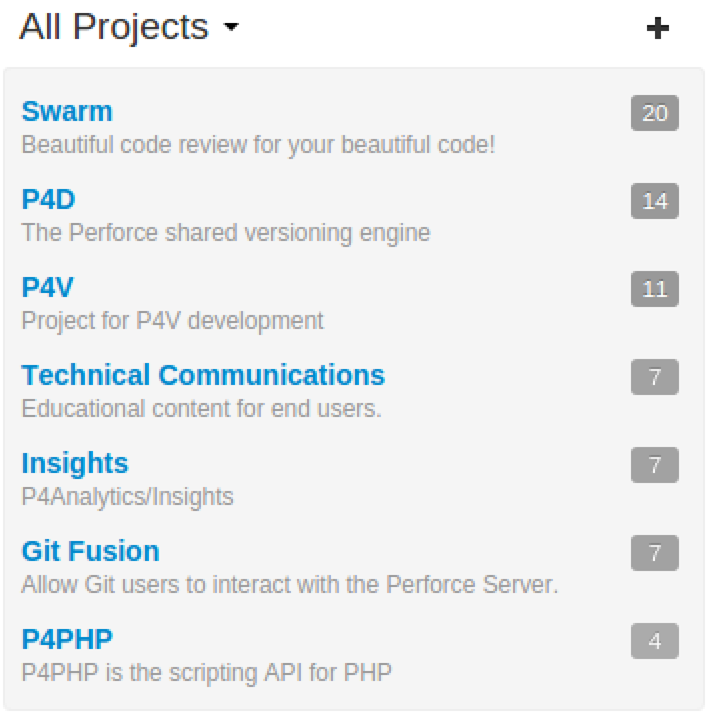
- In the left sidebar:
  - The group's name.
  - The group's avatar, if it has one.
  - The group's description, if it has one.
  - The count of the group's owners and members.
  - The group's email address, if it has one
  - The avatars of the group's owners.
  - The avatars of the group's members.
- Any events generated by a group member is associated with the group, and appears in the right-hand area as an activity stream.

Click the **Reviews** link in the group toolbar to display the review queue associated with the group, which includes all reviews authored by any of the group's members.

Projects

A Swarm project is a group of Helix Core users who are working together on one or more codelines within the Helix server. A project's definition includes one or more branches of code, and optionally a job filter, automated test integration, or automated deployment. This section provides an introduction to the interactions users have with projects. See the "Projects" on page 188 chapter for details on managing projects.

Projects are listed on the Swarm home page. Anonymous users see a list of all public projects:



All Projects

Swarm 20
Beautiful code review for your beautiful code!

P4D 14
The Perforce shared versioning engine

P4V 11
Project for P4V development

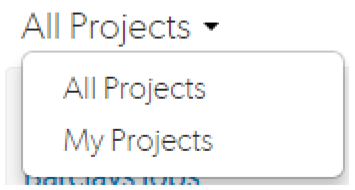
Technical Communications 7
Educational content for end users.

Insights 7
P4Analytics/Insights

Git Fusion 7
Allow Git users to interact with the Perforce Server.

P4PHP 4
P4PHP is the scripting API for PHP

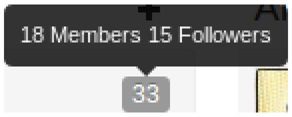
Logged-in users can choose to display all projects or just the projects they are a member of by clicking on the dropdown above the projects list:



- **All Projects:** lists all of the available projects.
- **My Projects:** list all the projects that you are a member of.

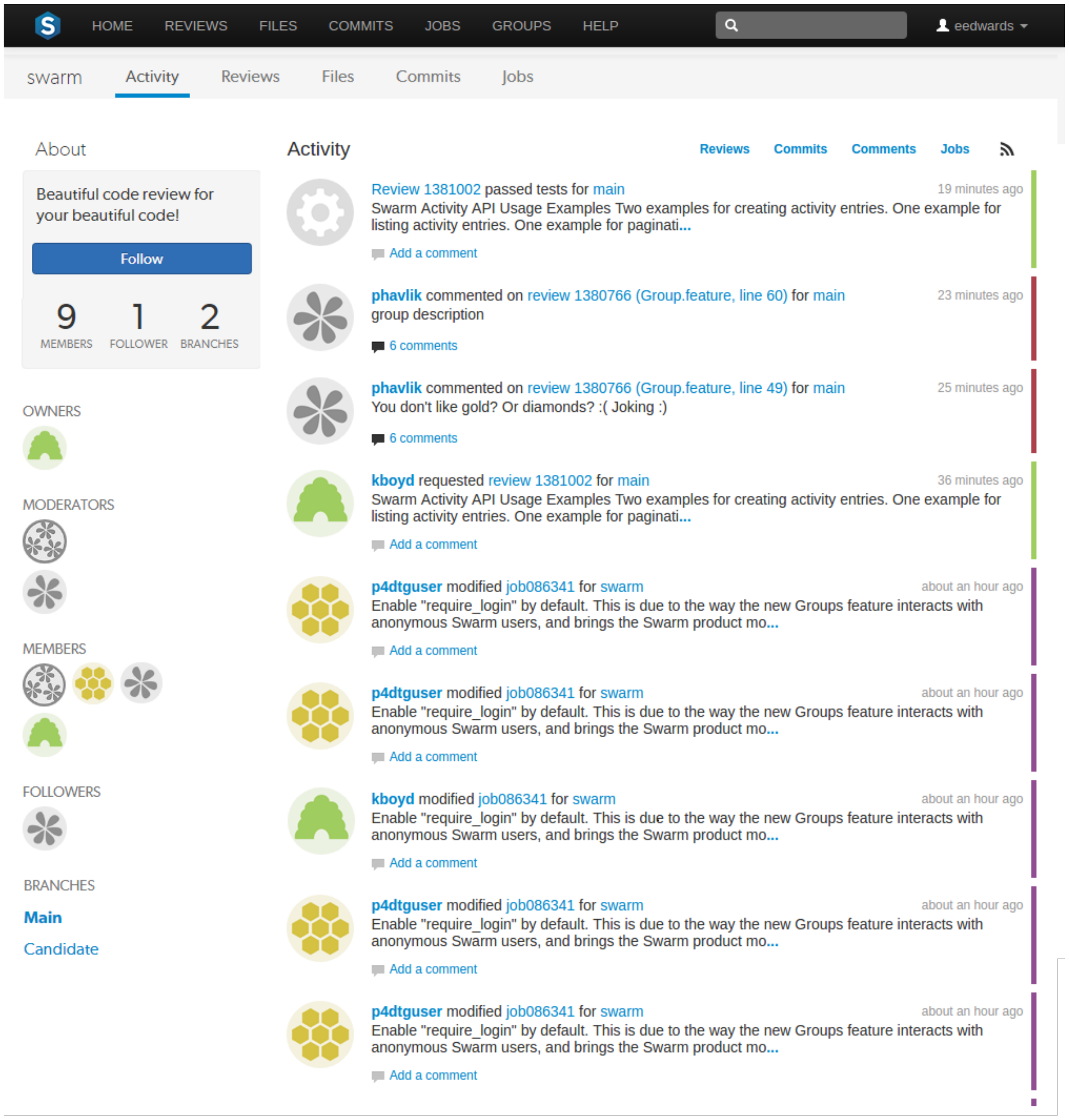
Note
If you are an owner of a project but not a member of that project, the project will not appear in your **My Projects** list. This allows you to administer a large number of projects but only be a member of the projects you want to see in your **My Projects** list.

The number in the icon to the right of the project name displays the total number of members and followers for the project. Hover your mouse over the icon to display a tooltip with the number of members and the number of followers for the project.



Viewing a project

View a project by clicking on its project name or branch identifier in an activity stream, or by visiting the URL: `https://`*`myswarm.url`*`/projects/`*`project-name`*



Beautiful code review for your beautiful code!

Review 1381002 passed tests for main — 19 minutes ago
Swarm Activity API Usage Examples Two examples for creating activity entries. One example for listing activity entries. One example for paginati...
Add a comment

phavlik commented on review 1380766 (Group.feature, line 60) for main — 23 minutes ago
group description
6 comments

phavlik commented on review 1380766 (Group.feature, line 49) for main — 25 minutes ago
You don't like gold? Or diamonds? :(Joking :)
6 comments

kboyd requested review 1381002 for main — 36 minutes ago
Swarm Activity API Usage Examples Two examples for creating activity entries. One example for listing activity entries. One example for paginati...
Add a comment

p4dtguser modified job086341 for swarm — about an hour ago
Enable "require_login" by default. This is due to the way the new Groups feature interacts with anonymous Swarm users, and brings the Swarm product mo...
Add a comment

p4dtguser modified job086341 for swarm — about an hour ago
Enable "require_login" by default. This is due to the way the new Groups feature interacts with anonymous Swarm users, and brings the Swarm product mo...
Add a comment

kboyd modified job086341 for swarm — about an hour ago
Enable "require_login" by default. This is due to the way the new Groups feature interacts with anonymous Swarm users, and brings the Swarm product mo...
Add a comment

p4dtguser modified job086341 for swarm — about an hour ago
Enable "require_login" by default. This is due to the way the new Groups feature interacts with anonymous Swarm users, and brings the Swarm product mo...
Add a comment

p4dtguser modified job086341 for swarm — about an hour ago
Enable "require_login" by default. This is due to the way the new Groups feature interacts with anonymous Swarm users, and brings the Swarm product mo...
Add a comment

The project's **Activity** tab provides the following information:

- A short description of the project
- The project's activity stream
- A **Follow** button, click to follow the project. This button is not available if you are a member of the project.
- A list of project owners
- A list of project moderators
- A list of project members

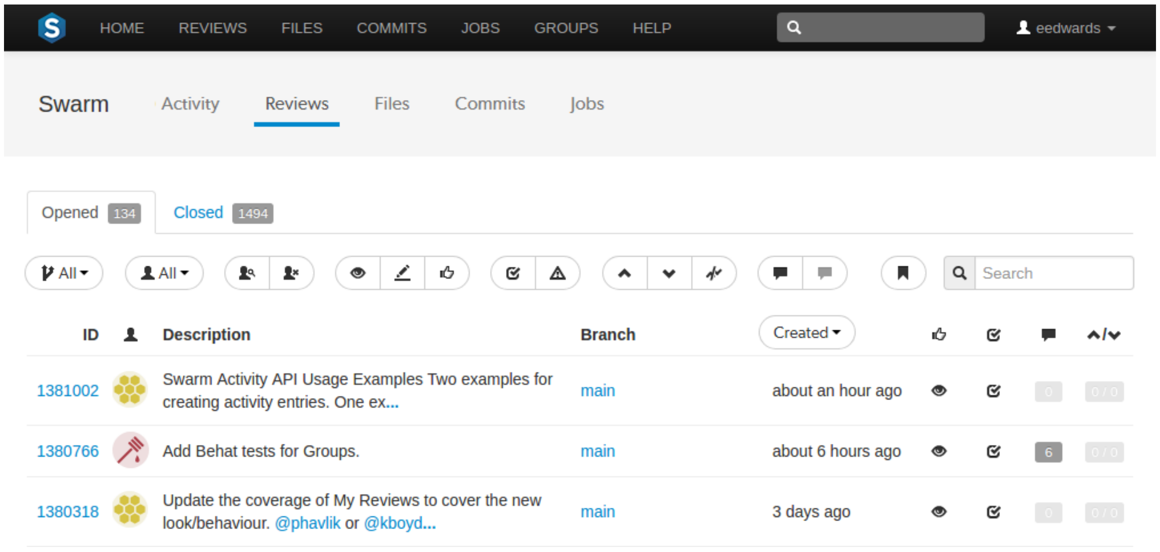
- A list of project followers
- A list of the branches defined for the project

If there is a **`README.md`** in the project's mainline, then this will be displayed in an **Overview** tab before the **Activity** tab. This file can contain Markdown text, allowing a formatted description of the project to be provided. See "Markdown in projects" on page 177 for a description of the type of formatting that is supported.

See "Mainline branch identification" on page 267 for details on how to configure the mainline of a project.

Reviews

The project's **Reviews** tab shows a list of code reviews specific to the project.

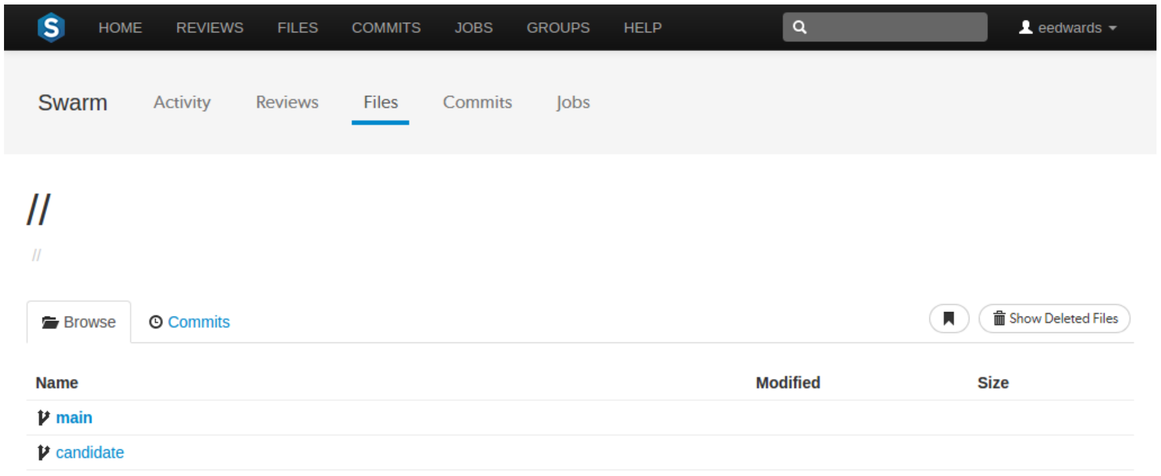


ID		Description	Branch	Created				
1381002		Swarm Activity API Usage Examples Two examples for creating activity entries. One ex...	main	about an hour ago			0	0 / 0
1380766		Add Behat tests for Groups.	main	about 6 hours ago			6	0 / 0
1380318		Update the coverage of My Reviews to cover the new look/behaviour. @phavlik or @kboyd...	main	3 days ago			0	0 / 0

For more details on browsing, filtering, and searching reviews, see "Review queues" on page 204.

Files

The project's **Files** tab shows a list of the project's files, starting with a folder view representing each branch. Branches are designated with the *branch* icon.



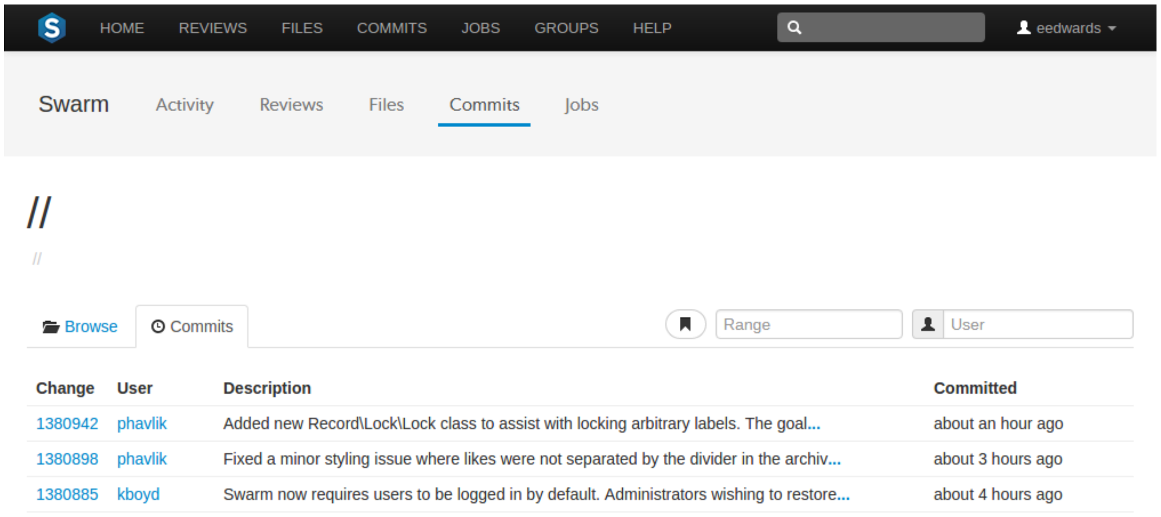
Name	Modified	Size
main		
candidate		

The project's *main* branch, identified by using a name such as *main*, *mainline*, *master*, *trunk*, is sorted to the top of the list of branches and appears in bold. The list of names can be configured; see "Mainline branch identification" on page 267 for details.

For more information on browsing files, see "Files" on page 124.

Commits

The project's **Commits** tab shows a list of changes made to the project.

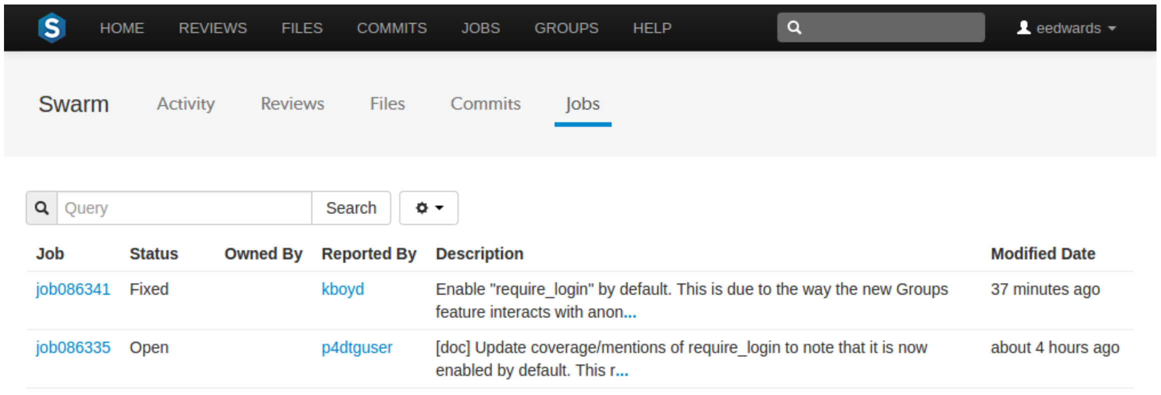


Change	User	Description	Committed
1380942	phavlik	Added new Record\Lock\Lock class to assist with locking arbitrary labels. The goal...	about an hour ago
1380898	phavlik	Fixed a minor styling issue where likes were not separated by the divider in the archiv...	about 3 hours ago
1380885	kboyd	Swarm now requires users to be logged in by default. Administrators wishing to restore...	about 4 hours ago

For more details on history browsing, see "Commits" on page 130.

Jobs

The project's **Jobs** tab shows a list of jobs associated with the project. This only works properly when the project configuration includes a *job filter*. See "Add a project" on page 188 for details.



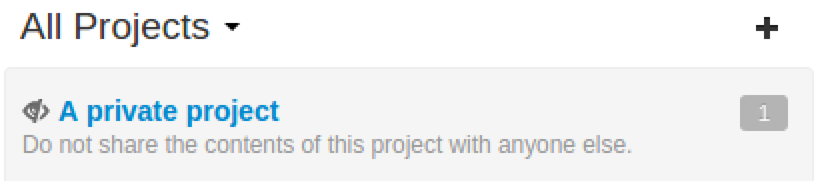
Job	Status	Owned By	Reported By	Description	Modified Date
job086341	Fixed		kboyd	Enable "require_login" by default. This is due to the way the new Groups feature interacts with anon...	37 minutes ago
job086335	Open		p4dtguser	[doc] Update coverage/mentions of require_login to note that it is now enabled by default. This r...	about 4 hours ago

For more details on browsing and searching jobs, see "Jobs" on page 133.

Private projects

Private projects, introduced in Swarm 2016.2, provide a way to make specific projects and their activity less visible to Swarm users. When a project is made private, only the projects owners, moderators, and members, plus users with *super* privileges in the Helix server, can see the project, its activity streams, and ongoing reviews.

If you are logged in as an owner, moderator, or member of a private Swarm project, that project appears on the Swarm home page with an *eye* icon to indicate that it is private and has limited visibility:



Similarly, the *eye* icon appears beside the project's title when viewing the project. When you hover your mouse over the *eye* icon, a tooltip appears indicating that this project is indeed private.

Caveats

The following are important caveats regarding private projects:

- While Swarm can mask the existence of projects, their activity, and reviews, Swarm honors each user's access to files within the Helix server. This means that have access to the files in a private project's branches can browse to those branch paths within the depot and see the files and any committed changes.
- If you need to prevent access to important files, your Helix server administrator is required to manage the protections table accordingly. For more information, see the section "Authorizing Access" in the *Helix Versioning Engine Administrator Guide: Fundamentals*.
- It is possible for a user to start a review that touches files that belong to a private project. If the user is not a member, owner, or moderator of the private project, or does not have super privileges in the Helix server, they cannot participate in the review.

- If a user is not a member, owner, or moderator of a private project, or does not have super privileges in the Helix server, and they are added as a review participant (via an "@mention" on page 171 or by editing a review's participants) to a review containing files within the private project's branches, that user cannot participate in the review: the user cannot see the review, its files, or comments. Due to limitations in how Swarm sends email notifications, such users could still receive notifications for reviews they cannot participate in.
- Notifications usually include links and mentions of the associated projects. Notifications involving private projects are filtered to remove any references to those private projects.

Notifications

Provided you have entered a working email address for your userid in Helix server, Swarm sends email notifications to you when various events take place. The table below shows the notifications sent if all of the project, group, and user notifications are enabled, this is the default behavior.

Note
The system-wide default notifications can be changed by the system owner. See "Global settings" on page 270 for details.

Tip
Many of the notifications can be disabled, this allows project, group, and user notifications to be customized to limit the number and type of notifications sent.

- Project notifications are configured on the project settings tab.
- Group notifications are configured on the group Notifications tab.
- User notifications are configured on the user Settings tab.

Legend:

A = Author	**PM** = Project member or project member group	✓ = role user/group receives notification
R = Reviewer or reviewer group	**PF** = Project follower	✗ = role user/group does not receive notification
M = Moderator or moderator group	**AF** = Author follower	⊘ = role not included for this event
GM = Group member	**CA** = Comment author	

Event	Notification sent to one of these roles...								
	Event by you?	A	R	M	GM	PM	PF	AF	CA
A review is committed in Swarm	✗	✓	✓	✓ 3	✓ 4	✓ 1	✓ 1	✗	⊘
A review is committed outside of Swarm	✗	✗	✓	✓ 3	✓ 4	✓ 1	✓ 1	✗	⊘
A change is committed outside of Swarm (see "(1) Committed change notifications" on page 167)	✗	✗	⊘	✓ 3	✓ 4	✓ 1	✓ 1	✓ 1	⊘
A review is started (see "(2) Review start notifications" on page 168)	✓	✓	✓	✓ 3	✓ 4	✓ 2	✗	✗	⊘
A reviewer casts a vote	✓	✓	✓ 5	✗	✓ 4	✗	✗	✗	⊘
A review's state changes	✗	✓	✓ 5	✗	✓ 4	✗	✗	✗	⊘
A review's reviewers are changed	✗	✓	✓ 5	✗	✓ 4	✗	✗	✗	⊘
A review's files are updated	✓	✓	✓ 5	✗	✓ 4	✗	✗	✗	⊘
A review's automated tests have finished (see "(7) Tests have finished" on page 169)	✗	✓	✓	✓	✓ 4	✗	✗	✗	⊘
A review's description is changed	✗	✗	✗	✗	✗	✗	✗	✗	⊘

Event	Notification sent to one of these roles...								
	Event by you?	A	R	M	GM	PM	PF	AF	CA
A review comment is created	✗	✓	✓ 5	✗	✓4	✗	✗	✗	✗ 6
A review comment is edited	✗	✓	✓ 5	✗	✓4	✗	✗	✗	✗ 6
A committed change comment is created	✗	✓	⊘	✗	✓4	✗	✗	✗	⊘6
A committed change comment is edited	✗	✓	⊘	✗	✓4	✗	✗	✗	⊘6
Someone joins or leaves a review	✗	✓	✓ 5	✗	✓4	✗	✗	✗	⊘
Someone likes a comment you wrote	✗	✗	✗	✗	✗	✗	✗	✗	✓5
You like a comment you wrote	✓	✗	✗	✗	✗	✗	✗	✗	✗

Any custom modules added to Swarm may also send notifications.

Important
Email delivery for events related to restricted changes is disabled by default. See "Restricted Changes" on page 289 for details on how to enable restricted change notifications.

@mention notifications

Users: @mention

Using an @mention in a review, changelist, or comment causes the referenced userid to receive a notification and be included in any future notifications regarding the associated file or review.

When a comment is added to a job, Swarm sends a notification to users listed in user fields in the job, users @mentioned in the job description, and the authors of any associated changes.

Note
Typically, you would not receive a notification when you @mention yourself. @mentions themselves do not trigger notifications; they inform who receives notifications. See "notify_self" on page 256 for details on how to enable self notification.

Groups: @@mention

Using an @@mention in a review, changelist, or comment causes the referenced groupid members to receive a notification and be included in any future notifications regarding the associated file or review.

When a comment is added to a job, Swarm sends a notification to groups listed in group fields in the job, groups @@mentioned in the job description, and the authors of any associated changes.

Note

- **Group mailing list enabled:** notifications are sent to the group email address.
- **Group mailing list disabled:** notifications are sent to the group members individual email addresses.

(1) Committed change notifications

Notifications for committed changes are sent by default, but can be disabled or require users to opt-in. Please see the notification configuration and adding a project for more details.

When committed change notifications are configured for opt-in, you need to copy the configuration's special depot path to the `Reviews:` field in your user spec within Helix server. Once you have done so, Swarm can send you committed change notifications for any change that matches a depot path specified in the `Reviews:` field.

Update Reviews with p4

1. Begin editing your user spec:

   ```
   $ p4 user
   ```

2. Edit the `Reviews:` field to include the special depot path, plus any other paths you want to receive notifications for.
3. Save the spec.

Update Reviews with P4V

1. Select **Connection > Edit Current User**.
2. Edit the `Reviews:` field to include the special depot path, plus any other paths you want to

receive notifications for.

3. Click **OK**.

(2) Review start notifications

By default, notifications are sent when a review is started, but can be disabled for a project, group, or user.

Note
If a user or group is added as a reviewer when a new review is created, the reviewer will be notified that the review has been started. This initial notification cannot be muted.

(3) Moderator notifications

Moderators and moderator groups are associated with specific project branches. Moderators receive notifications for reviews and commits against the branches they are associated with.

(4) Group member notifications

Groups can be members of a project, moderators of a project branch, and reviewers of a review. The notifications that group members receive will depend on, the role the group has and the group notification settings that are configured for that group. Notifications received by group members also depend on whether the group has a group mailing list enabled, in this case additional notification settings are available. See *Group mailing list enabled* below for details.

Group mailing list disabled

Group members can be notified when a member of the group starts a review. Group members can also be notified when a change is committed by, or on behalf of, a changelist owner who is also a member of the group. These two notifications can be individually selected as required. See "Add a group" on page 180 for details.

Notifications are sent to the group members individual email addresses.

Group mailing list enabled

Group members can be notified when a member of the group starts a review. Group members can also be notified when a change is committed by, or on behalf of, a changelist owner who is also a member of the group. These two notifications can be individually selected as required.

Additional notifications are available if the group mailing list address is enabled. See "Add a group" on page 180 for details.

Notifications are sent to the group email address.

(5) Disable notifications

By default, notifications for events in a review are sent to all reviewers. Reviewers can disable notifications during a review to avoid further email notifications. Once notifications are disabled for a reviewer, they can be re-enabled when they are specifically @mentioned; reviewers can disable notifications again after they have been re-enabled. See "Disable notifications" on page 217 for details.

(6) Comment author notifications

By default, notifications are not sent to comment authors, but Swarm can be configured to send notifications of comments to comment authors. See "notify_self" on page 256 for details.

(7) Tests have finished

By default, notifications are sent when:

- Automated tests have failed for a review.
- The first time automated tests pass for a review after a test failure for that review.

Log in / Log out

When you are not logged into Swarm, certain features are unavailable to you, such as providing comments to changes or reviews, adding projects, and more.

To log in:

1. Click the **Log in** link found at the right of the main toolbar.
2. Type in your username and password, appropriate for the Helix server that Swarm is configured to use.
3. Select the **Remember Me** check box if you prefer to stay logged in between browser restarts.

   Note
   The Helix server can enforce maximum login times. You may become logged out even if **Remember Me** is checked.

4. Click the **Login** button.

Note
Whenever you log in, Swarm continues displaying the same page. Any features available to logged-in users appear automatically.

To log out:

1. Click your userid, found at the right of the main toolbar.
2. Select **Log out** from the drop-down menu.

require_login

By default, Swarm requires users to login, which prevents anonymous users from accessing a Helix server via Swarm. Users who have not logged in see a login page immediately when visiting Swarm:

The steps to log in are identical to using the **Login** dialog.

Swarm administrators can disable require_login to allow anonymous users to see commits, reviews, etc.

Note
service and *operator* users are not permitted to login.

For more information on these user types, see the *Helix Versioning Engine Administrator Guide: Fundamentals*.

Notable minor features

Quick URLs

Many developers like to type the least amount of information to locate the data they're looking for. Swarm tries to assist by handling URLs intelligently. If you visit a URL like:

```
https://myswarm.url/identifier
```

Swarm attempts to locate `identifier` as a review, changelist, depot path, project, job, user, group, depot name, and Git Fusion SHA1 (or fragment). If you visit the URL **`https://myswarm.url/123`**, Swarm redirects to **`https://myswarm.url/changes/123`**, provided that **`changelist 123`** exists. If you visit the URL **`https://myswarm.url/bob`**, Swarm tries each of the following URLs and redirects to the first match:

```
https://myswarm.url/changes/bob
https://myswarm.url/files/bob
https://myswarm.url/projects/bob
https://myswarm.url/jobs/bob
https://myswarm.url/users/bob
https://myswarm.url/groups/bob
```

If you enter an identifier that does not exist for any type of resource, Swarm displays a **`Page Not Found`** error.

Note
Changelist identifiers must be numeric, so there could never be a changelist called bob.

Note
Git Fusion SHA1 identifiers (or a fragment of one) work when the SHA1 refers to a single changelist. If the SHA1 refers to more than one changelist, which can occur for Git branches, Swarm reports a 404 error.

@mention

When you write a changelist description, job description, comment, or review, use `@mention` to refer to projects, changelists, jobs, and users. Use `@@mention` to refer to groups. Swarm automatically creates a link for each @mention and @@mention, so it's easy to navigate to the specified resource. For example, when you include `@job12345` in a comment, Swarm turns that text into a link that, when clicked, displays the Jobs page for `job12345`.

Note
For jobs, the `@` character is optional: Swarm creates job links for any text that looks like a job identifier, such as `job012345`.

Similarly, for changes and reviews, the `@` character is optional: Swarm creates change links for `change 1234`, or review links for `review 2345`.

When you start a code review, including a user `@mention`, or a group `@@mention` in the changelist description automatically makes them reviewers for that code review. During a code review, including a user `@mention` or group `@@mention` in a comment causes the mentioned user or group to receive notifications of code review events, even if they are not a member of your project or following you or your project.

Additional option for a user `@mention`:

- `@*mention`: include an asterisk (`*`) before the userid to make the user a required reviewer. See "Required reviewers" on page 229 for details.

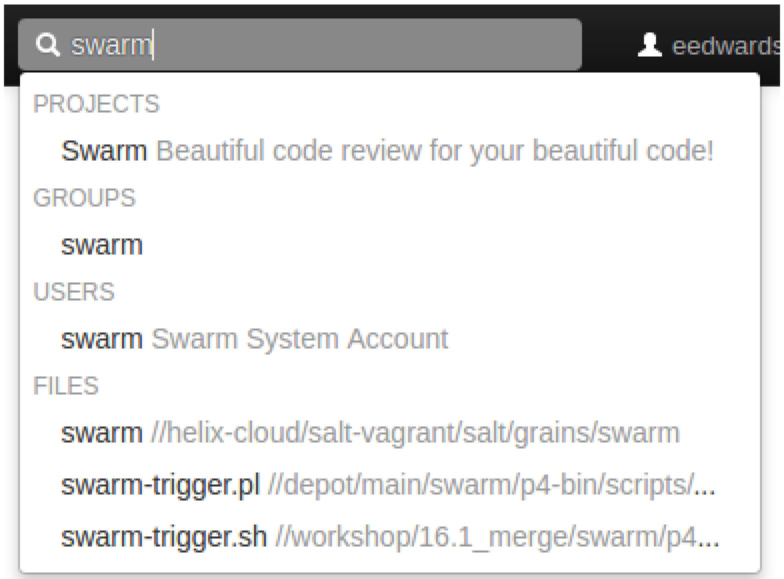
Additional options for a group `@@mention`:

- `@@*mention`: include an asterisk `*` before the groupid to make the group a required reviewer, **All votes required**. See "Required reviewers" on page 229 for details.
- `@@!mention`: include an exclamation mark `!` before the groupid to make the group a required reviewer, **One vote required**. See "Required reviewers" on page 229 for details.

Note
- **Group mailing list enabled:** notifications are sent to the group email address.

- **Group mailing list disabled:** notifications are sent to the group members individual email addresses.

Search

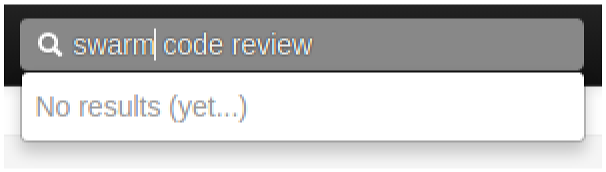


swarm
eedwards
PROJECTS
Swarm Beautiful code review for your beautiful code!
GROUPS
swarm
USERS
swarm Swarm System Account
FILES
swarm //helix-cloud/salt-vagrant/salt/grains/swarm
swarm-trigger.pl //depot/main/swarm/p4-bin/scripts/...
swarm-trigger.sh //workshop/16.1_merge/swarm/p4...

Swarm can search for users, groups, projects, and file paths. Enter keywords or path elements into the search field, which appears at the right of the Swarm toolbar, to display any matching results.

Navigate the results with the ↑ (up arrow) or ↓ (down arrow) keys, and display the details for the result by pressing **Enter**. Or, click a search result with your mouse.

Full-content searching is only available if your Swarm administrator installs the Helix Core Search Tool. See "Search" on page 285 for details.



Swarm updates the search results as you type. Some results should appear within a second or two. You may have to wait a few seconds for final results to be incorporated into the results list. When Swarm does not yet have any results, it indicates such.

JIRA integration

Swarm ships with a JIRA module that can:

- create links to JIRA when Swarm displays JIRA issue identifiers in changelists, comments, jobs, reviews, etc.
- add links within JIRA back to Swarm, for JIRA issues associated with reviews or committed changes. These links reflect the current status of associated code reviews.

By default, the JIRA module is disabled. To enable JIRA integration, see "Enabling the JIRA module" on page 236.

Avatars

Each event in an activity stream includes an *avatar*, an image that represents the user responsible for the event. Avatars help to visually tie together various events and personalize the history presented in the stream.

Based on the email address entered in a user's or group's Helix Core account, Swarm attempts to fetch an avatar from gravatar.com. Otherwise, Swarm selects from its collection of default avatars and color palettes to provide pleasing variations.

Hover your mouse over an avatar to display a tooltip with the user's or groups full name.

Following

Whenever you see a **Follow** button, for example when you are viewing a project page or user profile, clicking the button causes Swarm to send you notifications whenever there is activity generated by the current resource.

This is useful if, in the case of a project, you are not a project member but want to know what's happening in the project. Or, in the case of a user, you want to see what activity that user generates.

To stop receiving notifications, visit the project page or user profile and click the **Unfollow** button.

Time

Swarm typically displays the time of an event, such as when a file was created, as about *X units ago*. Hover your mouse pointer over a time display to see a tooltip displaying the exact date and time of the event.

Keyboard shortcuts

Swarm provides the following keyboard shortcuts:

- **n** - While viewing "Diffs" on page 140, pressing **n** scrolls to the next difference.
- **p** - While viewing "Diffs" on page 140, pressing **p** scrolls to the previous difference.

- **ESC** - While viewing a dialog, pressing **ESC** closes the dialog.

  While entering text into a text area, pressing **ESC** stops text entry.

About Swarm

You can discover the version of Swarm you are using:

1. Log in to Swarm.
2. Click your userid, found at the right of the main toolbar, and select **About Swarm**.

   A dialog appears displaying the Swarm version.

Custom error pages

If Swarm encounters an error during processing, such as when a "Quick URLs" on page 170 is used that points to a non-existent resource, Swarm presents a custom error page featuring its mascot, Bizzy Heisenbug. Bizzy's friends swarm your mouse pointer while you are moving it.

Short links

Swarm provides a *short link* feature that creates shorter URLs than normal to make sharing specific views within Swarm easier. It is also possible to register or configure an alternate, shorter hostname to have even shorter URLs. See "Short links" on page 295 for details.

Conceptually, this is identical to http://tinyurl.com, or the Twitter feature http://t.co, but is restricted to Swarm URLs on a hostname you control.

Swarm displays a *bookmark* button 🔖 when viewing files or folders in the depot.

Click the button to display a popup containing the short link. Press **CTRL+C** (on Windows and Linux), or **Command+C** (on Mac OSX), to copy the short link. You can then paste the short link anywhere you'd like to share the current file or folder view in Swarm.

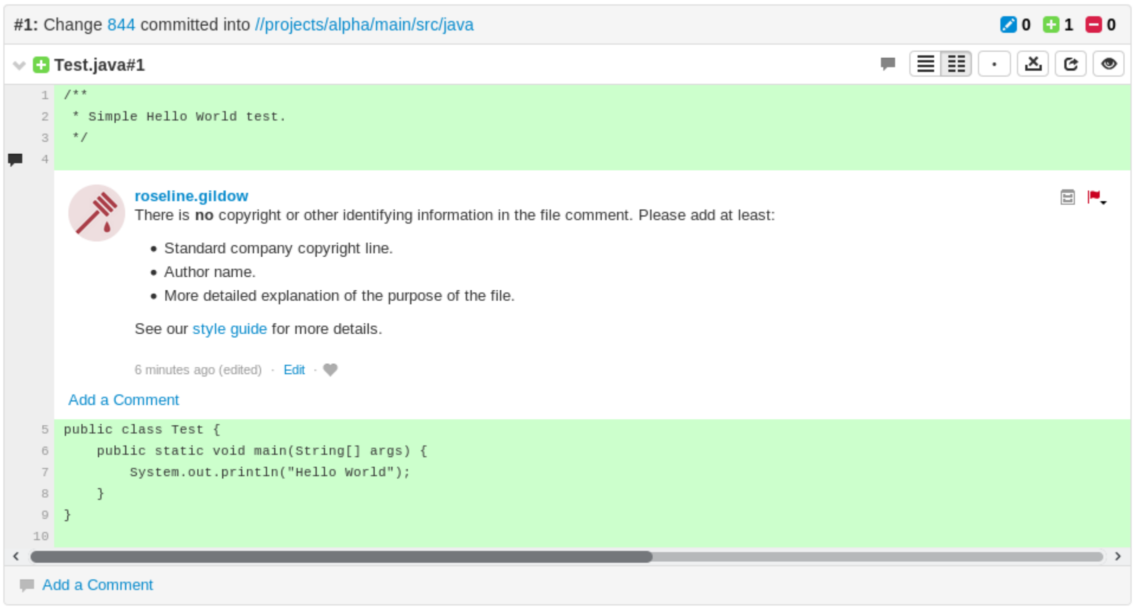
Mobile browser compatibility

Swarm is intended to be used from any modern browser, including most mobile device browsers. For our initial release, mobile browser testing has been quite limited. We anticipate that you may encounter various issues, but we certainly look forward to hearing about your mobile device experiences using Swarm.

Markdown

It is possible to use *Markdown* within Swarm to add text styles to comments and project overview pages. It is not supported in reviews.

Markdown in comments


#1: Change 844 committed into //projects/alpha/main/src/java
0 1 0
Test.java#1
Add a Comment
Add a Comment

```
/**
 * Simple Hello World test.
 */

public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

roseline.gildow
There is **no** copyright or other identifying information in the file comment. Please add at least:

- Standard company copyright line.
- Author name.
- More detailed explanation of the purpose of the file.

See our style guide for more details.

6 minutes ago (edited) · Edit ·

The supported set of Markdown styles in review comments is limited to inline styles, so headers and other structural syntax are ignored.

- **bold** and *italics* can be specified with `**bold**` or `*italics*`.
- Unordered lists can be specified with asterisk (`*`) markers. Plus (`+`) and minus (`-`) signs also work:

  ```
  * This
  * That
  * Another
  ```

- Ordered lists can be specified with numbered markers:

  ```
  1. First
  2. Second
  3. Third
  ```

- Hypertext links can be specified with [Link text](http://address/page). If you don't need to add anchor text, then a URL in the text without any markup is linkified.
- You can mark inline text as code using `backticks`.

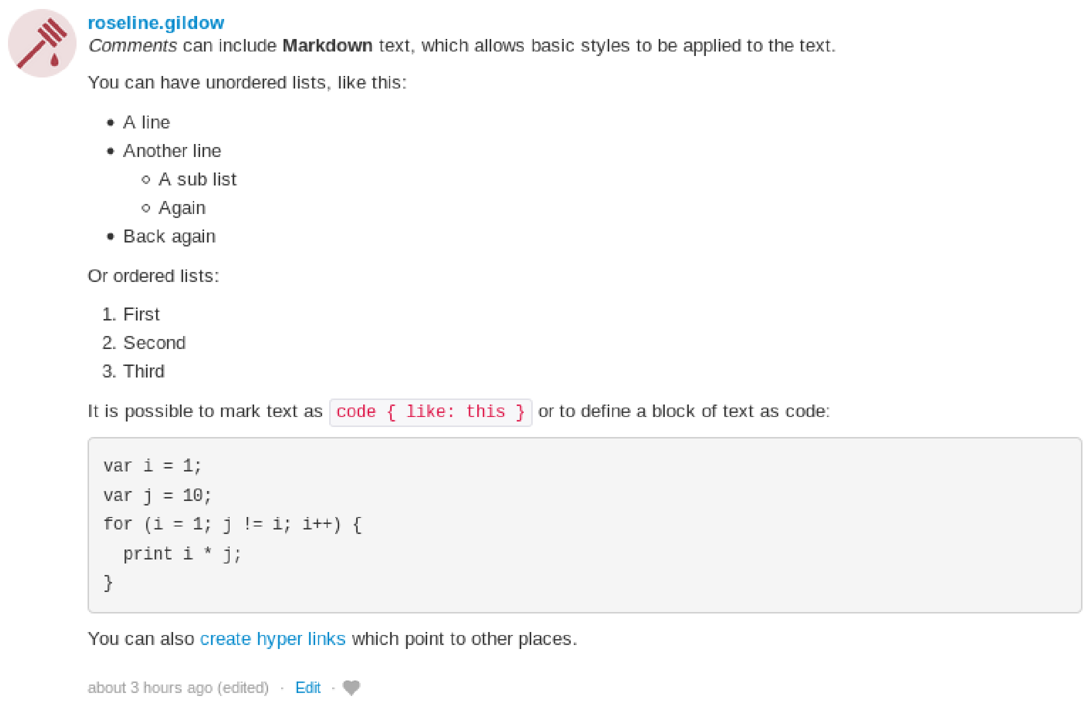
- You can also mark blocks of code using three backticks on a line, like:

  ~~~
  ```
  var text = "Some code text";
  alert(text);
  ```
  ~~~

Markdown is only supported in comments, not in the review description.

The following example shows all of the above syntax in a single example, with both the source and the final result.

~~~
*Comments* can include **Markdown** text, which allows basic styles to be
applied to the text.

You can have unordered lists, like this:
* A line
* Another line
  * A sub list
  * Again
* Back again

Or ordered lists:
1. First
2. Second
3. Third

It is possible to mark text as `code { like: this }` or to define a block
of text as code:
```
var i = 1;
var j = 10;
for (i = 1; j != i; i++) {
  print i * j;
}
```

You can also [create hyper links](http://www.perforce.com) which point to
other places.
~~~



roseline.gildow
Comments can include **Markdown** text, which allows basic styles to be applied to the text.

You can have unordered lists, like this:

- A line
- Another line
  - A sub list
  - Again
- Back again

Or ordered lists:

1. First
2. Second
3. Third

It is possible to mark text as `code { like: this }` or to define a block of text as code:

```
var i = 1;
var j = 10;
for (i = 1; j != i; i++) {
  print i * j;
}
```

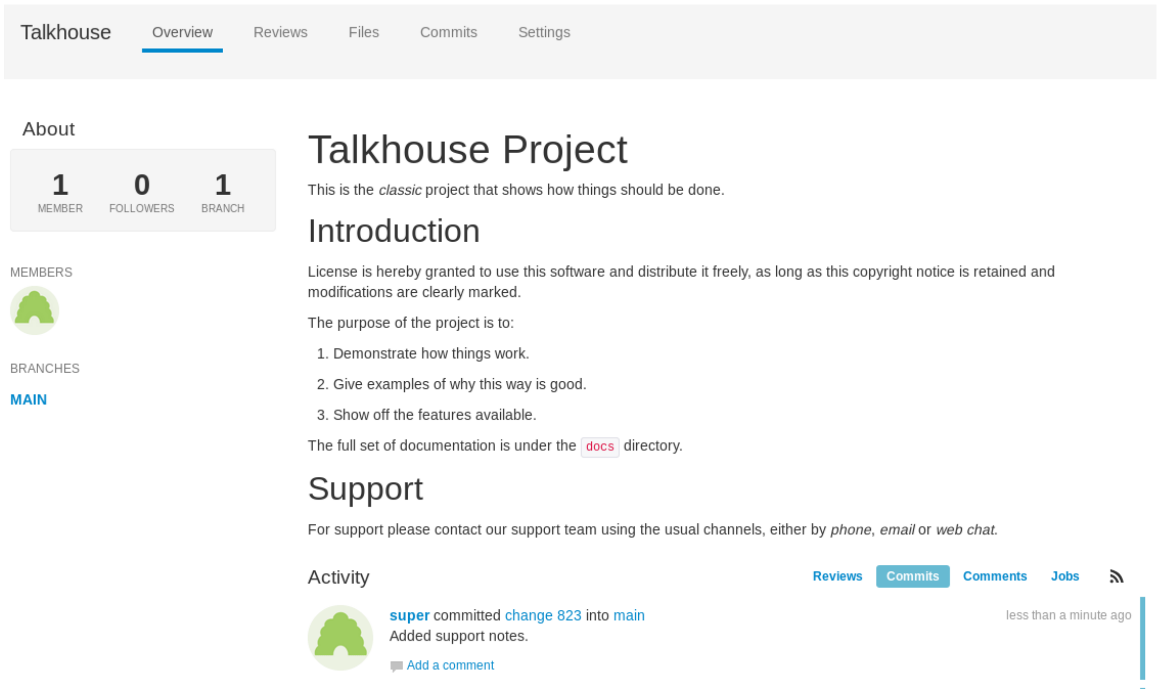
You can also create hyper links which point to other places.

about 3 hours ago (edited) · Edit ·

Markdown in projects

Project pages can have a `README.md` file in the root of their `MAIN` branch, which if present will be displayed on the project's overview page.

If you need to change which branch is considered MAIN, and therefore from where the README.md is read from, see "Mainline branch identification" on page 267.



Talkhouse Project

This is the *classic* project that shows how things should be done.

Introduction

License is hereby granted to use this software and distribute it freely, as long as this copyright notice is retained and modifications are clearly marked.

The purpose of the project is to:

1. Demonstrate how things work.
2. Give examples of why this way is good.
3. Show off the features available.

The full set of documentation is under the `docs` directory.

Support

For support please contact our support team using the usual channels, either by *phone*, *email* or *web chat*.

super committed change 823 into main
Added support notes.

As well as the *Markdown* syntax that is supported in comments and described above, the README pages support a more extensive range of syntax, the most notable of which are described below. For a more thorough list of syntax, see here.

- Headers can be defined using hash (#) marks.

```
# Main heading
## Sub heading
### Lesser heading
```

- Images can be added as well. You can either provide the full URL, or use a relative URL to reference something in Swarm.

```
![alt text](https://p4swarm/view/depot/www/dev/images/jamgraph-
jam.gif "Title Text")
![alt text](/view/depot/www/dev/images/jamgraph-jam.gif "Title Text")
```

- Tables are supported by using pipe (|) separators between columns and colons (:) for justification.

```
| Tables   | Look   | Like this |
| -------- | -----  | --------- |
| Left     | right: | :center:  |
```

- It is also possible to blockquote paragraphs using the greater than symbol (>).

  ```
  > Blockquotes can be displayed like this, using the
  > the greater than sign at the start of the line.

  Normal text resumes here.
  ```

If a `README.md` is displayed on the project overview page, then it appears above the activity stream.

5 | Groups

Groups are a feature of Helix server that makes it easier to manage permissions for users. Swarm can use groups to coordinate review activities and responsibilities. This chapter covers how to manage groups in Swarm, including how to:

- "Add a group" below
- "Edit a group" on page 187
- "Remove a group" on page 187

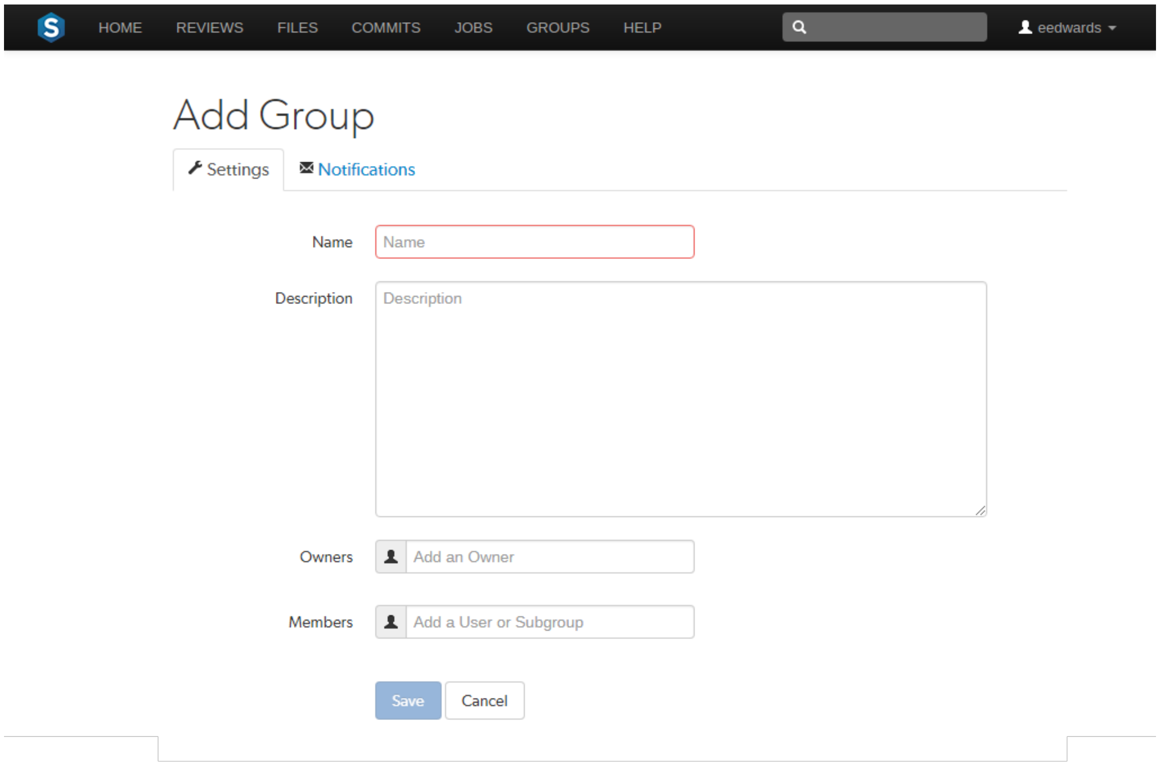
See "Groups" on page 155 for an introduction to Swarm groups.

Add a group

Important
You must have *super* privileges in Helix server (**p4d**), or have *admin* privileges in **p4d** version 2012.1 or later, to create a group. If you do not have sufficient permissions, Swarm does not display the **Add Group** button.

1. Click the **Groups** link in the main toolbar to display the groups listing page.
2. Click the **Add Group** button.

   The Add Group Settings tab appears.



3. Provide a name for the group.
4. **Optional:** provide a description.
5. **Optional:** specify an owner. This field auto-suggests users within Helix server as you type.

   Once specified, modifying the group's definition is restricted to group owners and users with *super* privileges in Helix server.

   If you do not specify an owner, you must specify at least one member (below).
6. **Optional:** specify group members. This field auto-suggests projects, groups, and users within Helix server as you type (up to a combined limit of 20 entries).

   If you specify a project, the project's members become members of the group. If you specify a group, that group becomes a sub-group of your new group, and all of its members (and members of any of its sub-groups) become members of your new group.

   If you do not specify any members, you must specify at least one owner (above).

7. You now have two options, either:

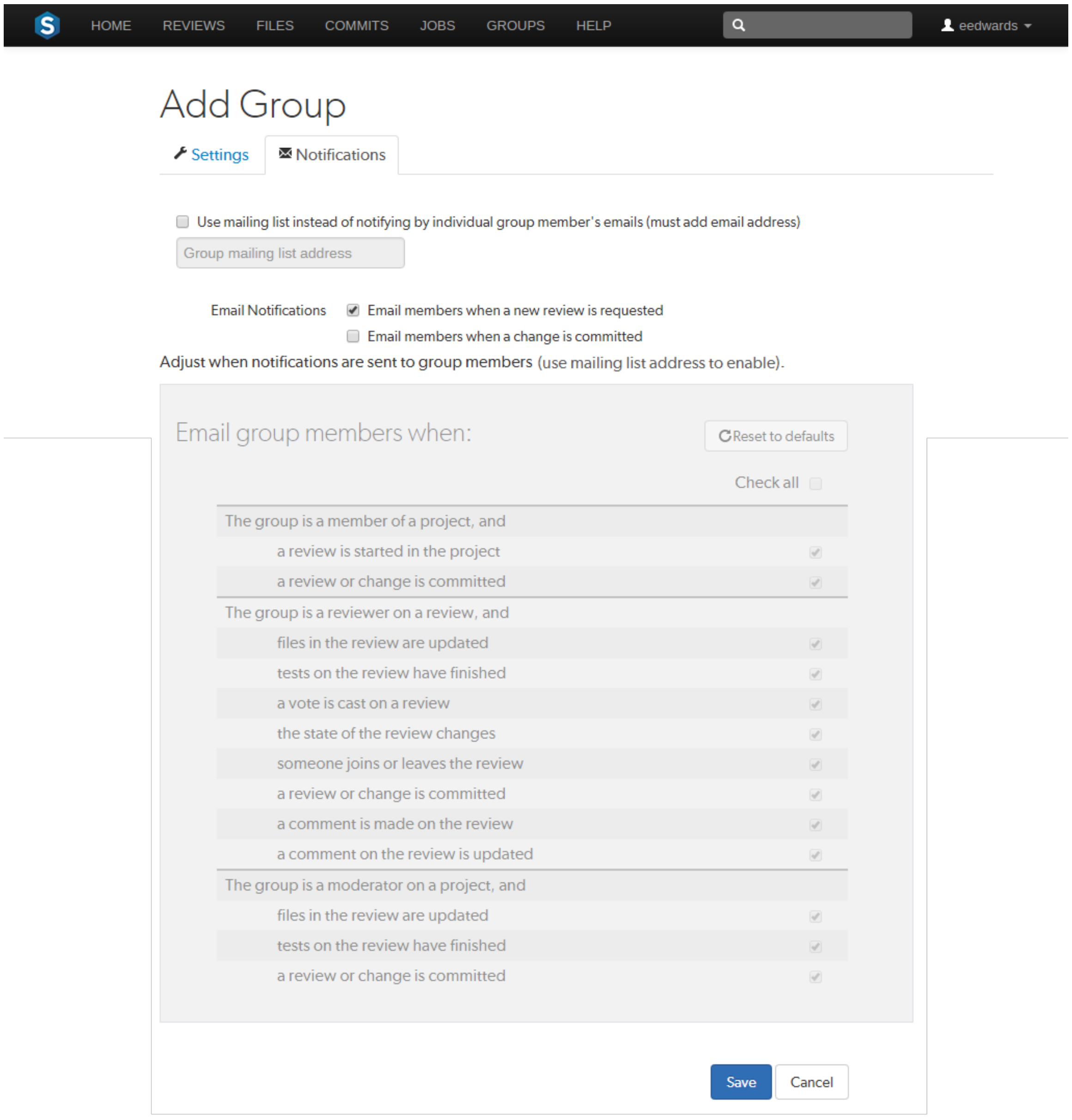
   - Click **Save** to finish adding the group. By default group members will be emailed when a new review is requested.

     > **Note**
     > The **Save** button is disabled if any required fields are empty.

     or

   - **Optional:** configure email notifications for the group in the Add Group, Notifications tab. See the next step for details.

8. **Optional:** click the **Notifications** tab to configure group email notifications.



Add Group

Settings | Notifications

[ ] Use mailing list instead of notifying by individual group member's emails (must add email address)

Group mailing list address

Email Notifications [✓] Email members when a new review is requested
[ ] Email members when a change is committed

Adjust when notifications are sent to group members (use mailing list address to enable).

Email group members when:

Reset to defaults

Check all [ ]

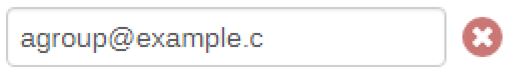
The group is a member of a project, and	
a review is started in the project	✓
a review or change is committed	✓
The group is a reviewer on a review, and	
files in the review are updated	✓
tests on the review have finished	✓
a vote is cast on a review	✓
the state of the review changes	✓
someone joins or leaves the review	✓
a review or change is committed	✓
a comment is made on the review	✓
a comment on the review is updated	✓
The group is a moderator on a project, and	
files in the review are updated	✓
tests on the review have finished	✓
a review or change is committed	✓

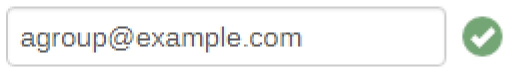
Save | Cancel

9. **Optional:** add a group mailing list address by selecting **Use mailing list instead of notifying by individual group member's emails (must add email address)** and entering a valid group email address.

   Note
   - **Group mailing list enabled:** notifications are sent to the group email address.
   - **Group mailing list disabled:** notifications are sent to the group members individual email addresses.

   The format of the email address is validated as you type.

   
   agroup@example.c

   
   agroup@example.com

10. Group members can be notified when a member of the group starts a review. Group members can be notified when a change is committed by, or on behalf of, a changelist owner who is also a member of this group. These settings are always available even if the group mailing list is not enabled.

    Select which actions send a notification to the group:

    - **Email members when a review is requested**: When any member of this group creates a review, this group will be notified.
    - **Email members when a change is committed**: When a change is committed into Perforce, if the owner of the changelist is a member of this group, this group will be notified.

      Tip
      When a user commits a changelist in Swarm, it is committed on behalf of the changelist owner. If the changelist owner is a member of this group, this group will be notified.

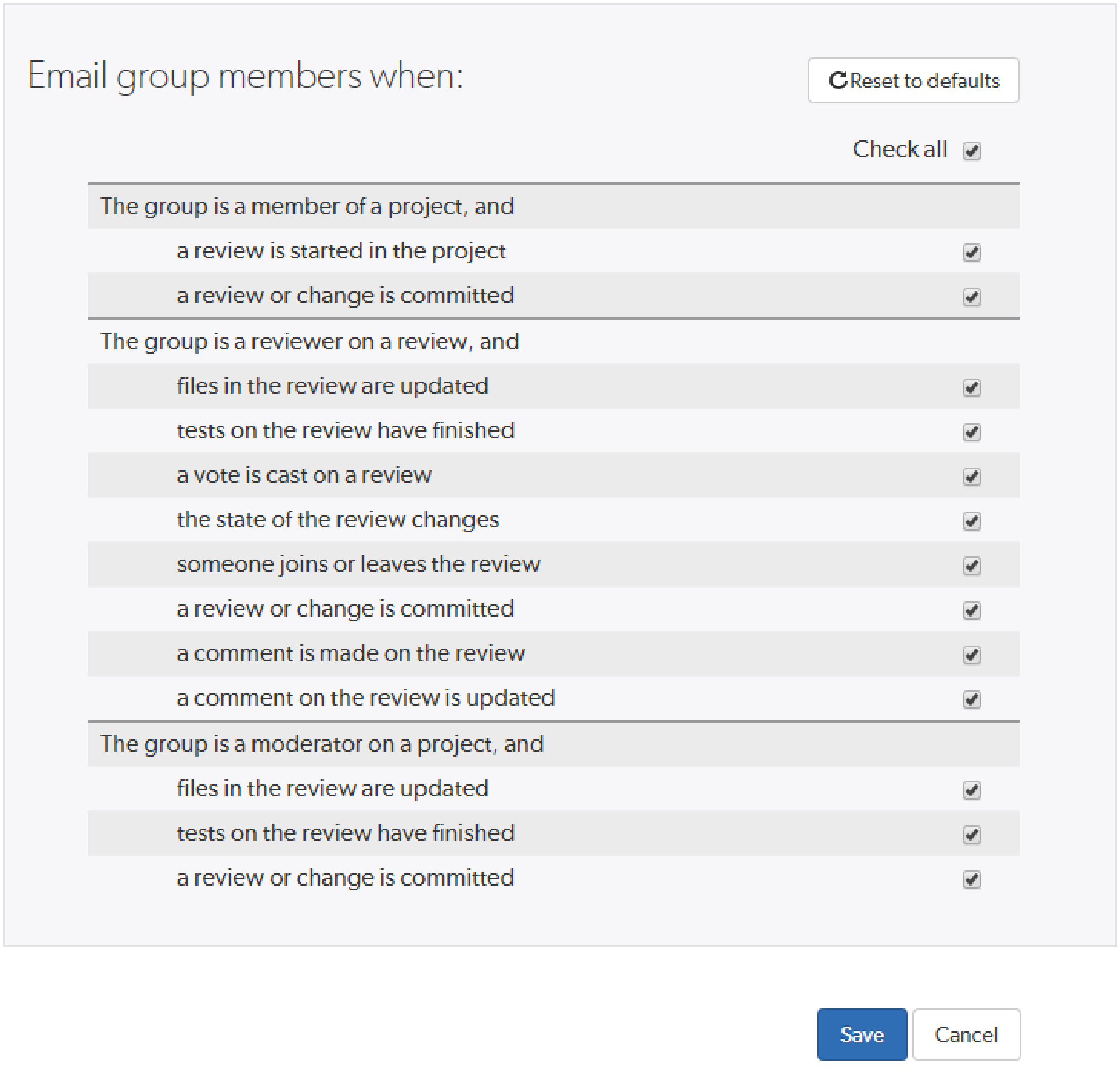
    Note
    Members of your group may receive notification emails even if group notifications are disabled as they may be members of a project, or follow a project or user, or Swarm's review daemon functionality may be enabled. See the notifications overview for details.

11. Group notification settings allow you to configure which notifications are sent to the group mailing list when events occur within Swarm (the group mailing list must be enabled). This allows you to limit the number of emails sent to the group mailing list. These settings apply across all projects.

    Note
    The following group notification settings are only available if the group mailing list address is configured.

    Note
    Defaults for these options are configured by the Swarm administrator, and they may force some of these options to on or off. See "Global settings" on page 270 for how this is configured.

Adjust when notifications are sent to group members.



Email group members when:

Check all	✓
The group is a member of a project, and	
a review is started in the project	✓
a review or change is committed	✓
The group is a reviewer on a review, and	
files in the review are updated	✓
tests on the review have finished	✓
a vote is cast on a review	✓
the state of the review changes	✓
someone joins or leaves the review	✓
a review or change is committed	✓
a comment is made on the review	✓
a comment on the review is updated	✓
The group is a moderator on a project, and	
files in the review are updated	✓
tests on the review have finished	✓
a review or change is committed	✓

Toggle notifications for each event on or off to control whether the group receives an email when that event occurs.

Clicking **Reset to default** resets the options back to system defaults.

12. Click **Save**.

Note
The **Save** button is disabled if any required fields are empty.

Edit a group

Important
You must be an owner of a group, or be a user with *super* privileges in Helix server , to edit a group.

1. Visit the group page you want to edit.
2. Click the **Settings** tab for the group to edit group details. If you do not have permission to edit a group, this tab does not appear.
3. Edit group details as required. See "Add a group" on page 180 for more details.
4. Click the **Notifications** tab for the group to edit group notifications. If you do not have permission to edit a group, this tab does not appear.
5. Edit group notifications as required. See "Add a group" on page 180 for more details.
6. Click **Save**.

Note
When a group is edited, the list of associated reviews is not immediately updated to match any changes in membership. The association with a review does not change until the review is updated.

Remove a group

Important
To remove a group, you must either have *super* privileges in Helix server, or be the group owner.

1. Visit the group page you want to remove.
2. Click the **Settings** tab for the group. If you do not have permission to edit a group, this tab does not appear.
3. Click **Delete**.

   A tooltip appears to confirm that you want to delete this group.
4. Click **Delete** to confirm.

6 | Projects

A Swarm project is a group of Helix Core users who are working together on a specific codebase, defined by one or more *branches* of code, along with a *job* filter, and automated test integration. This chapter covers Swarm's project management capabilities, including how to:

- "Add a project" below
- "Edit a project" on page 194
- "Membership" on page 194
- "Remove a project" on page 197

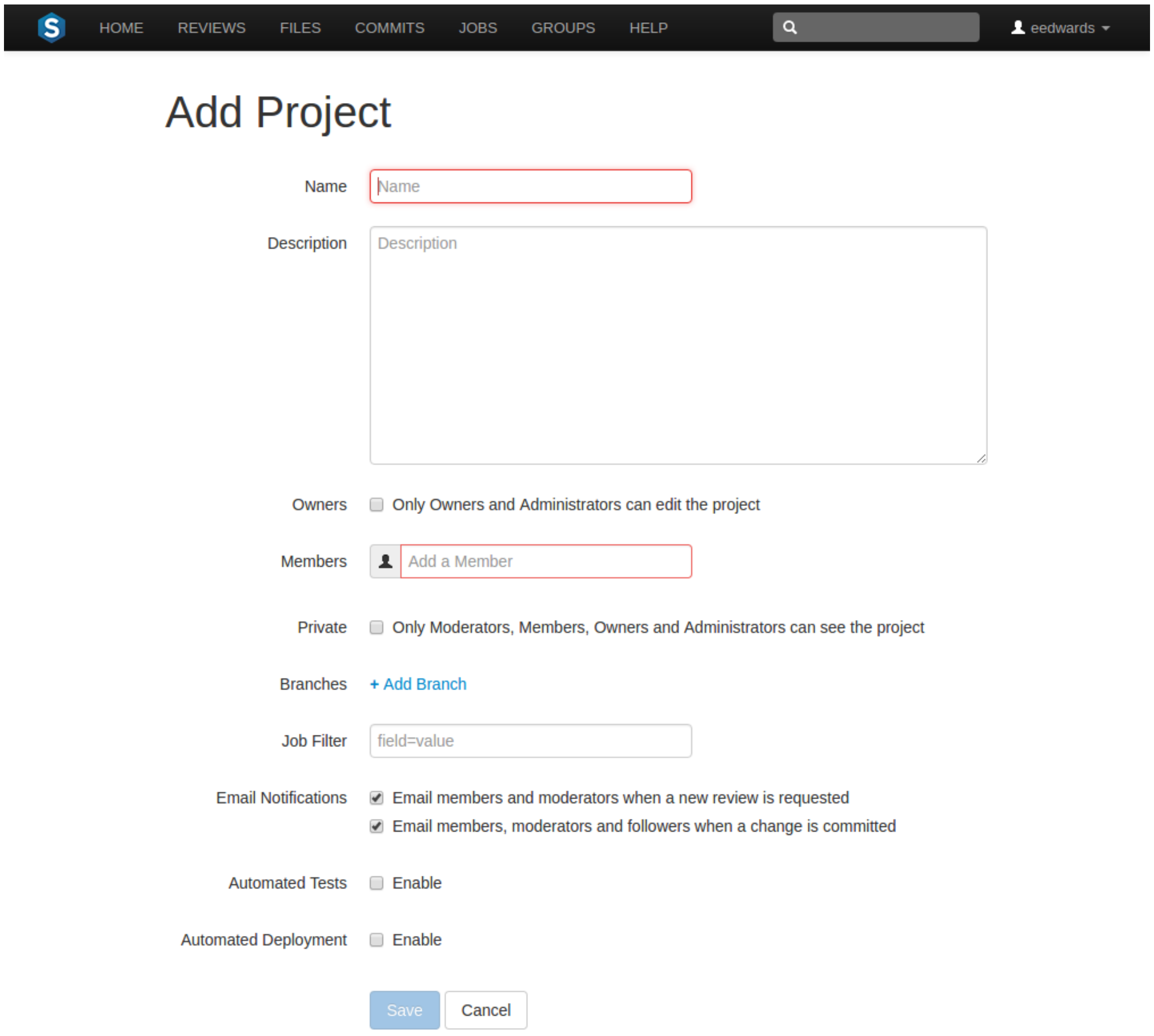
See "Projects" on page 158 for an introduction to Swarm projects.

Add a project

1. On the Swarm home page, click the **+** icon at the top-right of the projects sidebar.

   Note
   The ability to add projects can be limited to administrators only, or limited to members of specific groups. When limited, users who are not administrators, or a member of the specified group, will not see the **+** icon at the top-right of the projects sidebar.

   The **Add Project** page appears:



HOME REVIEWS FILES COMMITS JOBS GROUPS HELP eedwards

Add Project

Name	Name
Description	Description
Owners	☐ Only Owners and Administrators can edit the project
Members	Add a Member
Private	☐ Only Moderators, Members, Owners and Administrators can see the project
Branches	+ Add Branch
Job Filter	field=value
Email Notifications	☑ Email members and moderators when a new review is requested
	☑ Email members, moderators and followers when a change is committed
Automated Tests	☐ Enable
Automated Deployment	☐ Enable

2. Provide a name for the project.
3. Optionally provide a description.
4. Optionally select the **Only Owners and Administrators can edit the project** checkbox. When checked, a field is displayed allowing you to add a new owner. The field auto-suggests users within Helix server as you type.

   Note
   You cannot specify a group as an owner.

   Once specified, modifying the project's definition is restricted to project owners and administrators (users with *admin*-level or *super*-level privileges in Helix server).

5. Specify at least one team member. This field auto-suggests projects, groups, and users within Helix server as you type (up to a combined limit of 20 entries).

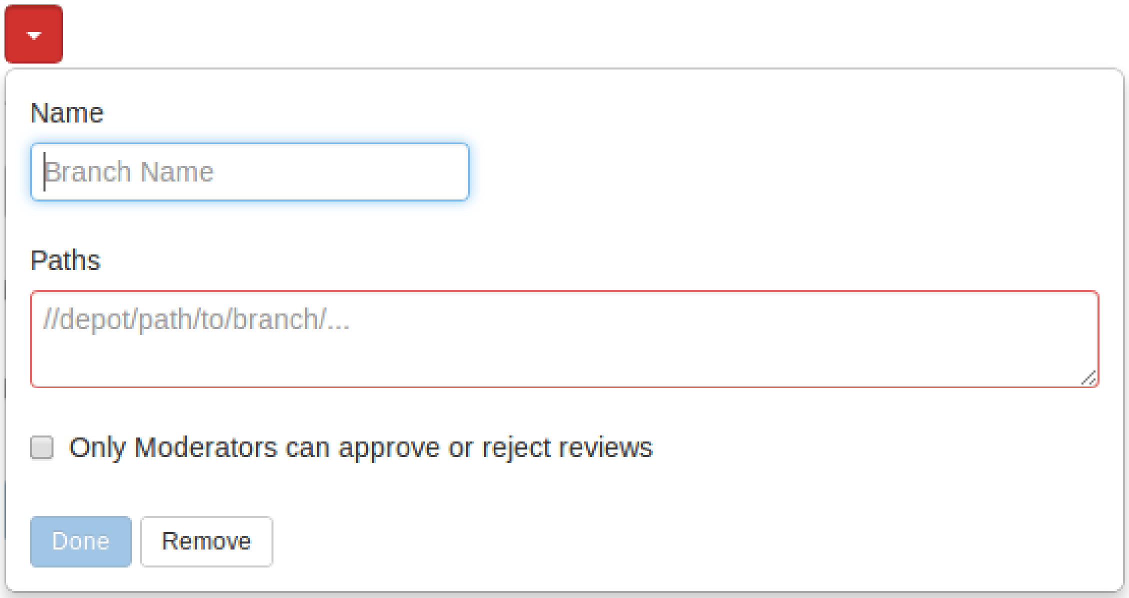
   Important
   During project creation, if you do not have *admin* privileges and you do not add yourself as a member or as an owner, you cannot edit this project's configuration later.

   See "Membership" on page 194 for more details.

6. Optionally click the **Private** checkbox to make this project private. Private projects and their associated reviews are only visible to project owners, moderators, and members, plus users with *super* privileges in Helix server.

   For more information, see "Private projects" on page 163.

7. Optionally click the **Add Branch** link to display the branch drop-down dialog.


Name
Branch Name
Paths
//depot/path/to/branch/...
Only Moderators can approve or reject reviews
Done
Remove

   a. Enter a short **Name** for the branch.
   b. Enter one or more branch paths, one per line.

> **Note**
> Each branch path should be expressed in depot syntax. Wildcards should not be used; the only exception is that the branch path can end with the Helix Core wildcard ...
>
> For more information, see "File Specifications" in *P4 Command Reference*.

c. Optionally check the **Only Moderators can approve or reject reviews** checkbox. When checked, a field is displayed allowing you to add a new moderator. The field auto-suggests groups and users within Helix server Engine as you type.

   If a group is specified as a moderator, all of the members of that group have the same moderator privileges for that project branch as if they were added individually.

   Once the branch specification is complete and the project has been saved, changing the state of any review associated with this moderated branch is restricted as follows:

   - Only moderators can approve or reject the review. Moderators can also transition a review to any other state.
   - The review's author, when not a moderator, can change the review's state to **Needs Review**, **Needs Revision**, **Archived**, and can attach committed changelists.

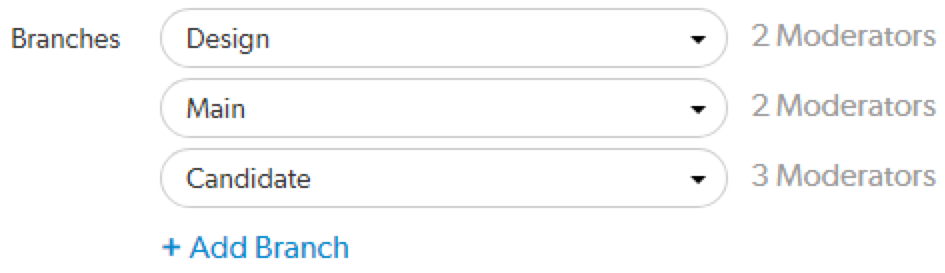
     Normally, the review's author cannot change the review's state to **Approved** or **Rejected** on moderated branches. However, authors that are also moderators have moderator privileges, and may approve or reject their own review.

     When `disable_self_approve` is enabled, authors who are moderators (or even users with *admin* privileges) cannot approve their own reviews.
   - Project members can change the review's state to **Needs Review** or **Needs Revision**, and can attach committed changelists. Project members cannot change the review's state to **Approved**, **Rejected**, or **Archived**.
   - Users that are not project members, moderators, or the review's author cannot transition the review's state.
   - For the review's author and project members, if a review is not in one of their permitted states, for example if the review's state is **Rejected**, they cannot transition the review to another state.

     These restrictions have no effect on who can start a review.

d. Click the **Done** button to accept your branch specification.

Once the branch definition has completed, if any moderators were specified, the number of moderators for that branch is displayed in the list of branches:



8. Optionally specify a job filter. The job filter allows you to specify criteria that are used to associate jobs with projects. For example, entering `Subsystem=ProjectA` associates jobs whose **subsystem** field is set to `ProjectA` with the current project.

   Note
   This job filter is simpler than the filters available in other Helix Core clients. The filter must be expressed as `field=value` pairs; bare keywords are not permitted. The asterisk for wildcard matching is permitted, but no other filter expression syntax is permitted.

9. By default project members and moderators are notified when a new review is started. Project members, moderators, and followers are notified when a change is committed.

   Select which actions send a notification:

   - **Email members and moderators when a new review is requested:** When a new review is requested for the project, all project members and moderators of the project are added to the email notification list.
   - **Email members, moderators and followers when a change is committed:** When a change is committed for the project, all project members, project moderators and project followers of this project are added to the email notification list.

   Note
   - When a group is a project member or project moderator, all of the members of that group are notified using the same logic as for individual project members and moderators.
   - Any "@mention" on page 171 users and groups, or users and groups who are explicitly added to a review or changelist, will receive notifications even if new review/committed review notifications are disabled.

10. Optionally click the **Enable** checkbox beside **Automated Tests** to display the automated tests configuration fields.

    Specify a URL that triggers a test execution. Use the special arguments described in the dialog to help compose a URL that informs your test suite with important details. For more details, see "How can I integrate my test suite to inform review acceptance or rejection?" on page 31

11. Optionally click the **Enable** checkbox beside **Automated Deployment** to display the automated deployment configuration fields.

    Specify a URL that triggers a deployment of the project's code. Use the special arguments described in the dialog to help compose a URL that informs your deployment program with important details. For more details, see "How can I automatically deploy code within a review?" on page 35

12. Click **Save**.

    Note
    The **Save** button is disabled if any required fields are empty.

    Important
    It is possible to create a project that you cannot edit. This can happen if you have specified owners but not yourself as an owner, or if you have not specified yourself as a member. Swarm can detect some (but not all) such situations when you save a project; when it does detect such a situation, a warning dialog is displayed.

    If you see this dialog, click **Continue** to save the project without your ownership/membership, or click **Cancel** within the dialog to continue editing the project. The project page's **Save** and **Cancel** buttons are disabled while this dialog is visible.

Edit a project

1. Visit the project page you want to edit.
2. Click **Settings** in the project's toolbar.
3. Adjust the project's details as required. See "Add a project" on page 188 for more details.
4. Click **Save**.

Note
By default, any member of a project can edit the project's configuration. Administrators can configure Swarm to prevent changes to the project's name and branch definition(s). See "Projects" on page 274 for details.

Membership

Membership in a Swarm project identifies users as belonging to the project, making them part of the team.

The are only a few notable differences between project members and non-members:

Difference	Member	Non-Member	Description
"Notifications" on page 164	✓	✗	Members receive project notifications; non-members do not.

Difference	Member	Non-Member	Description
"Avatars" on page 173	✓	✗	The project's home page features member's avatars.
"States" on page 231	✓	✗	Members can transition code review states; non-members cannot.

There are two ways to become a member of a project in Swarm:

1. Add a project and make yourself a member.
2. Ask a member of an existing project to add you as a member.

> **Note**
> If the project has any owners specified, you need to ask a project owner to add you as a member.

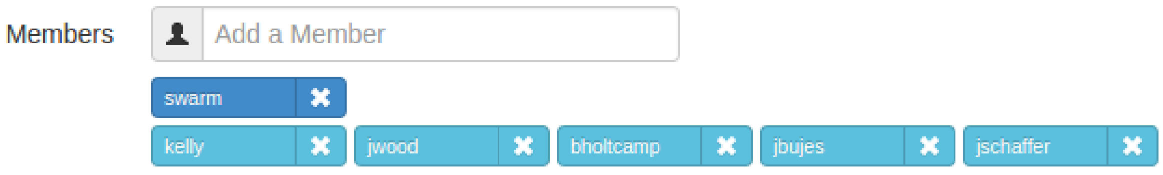
> **Note**
> Users with *super* privileges in Helix server can always adjust the settings for any project, including adjusting membership.

Add a member

If you are an owner of a project, or a member of a project without specified owners:

1. Visit the project page that needs the new member.
2. Click **Settings** in the project's toolbar.
3. The **Members** text field lets you specify a Swarm project, Helix Core group, or Helix Core user to add to the members for this project. The field auto-suggests project ids, group ids, and userids by matching what you have typed so far against the list of users in the Helix server.

   When you specify a project or group, all of the members of that project or group become members of this project. Swarm does not display all of the individual users, but it does provide a visual separation: project or group names are displayed first, with a darker blue background.

   When you hover your mouse over a member project or group, a tooltip appears displaying up to 100 of the userids of the project's or group's users.
4. Click **Save**.

Remove a member

If you are an owner of a project, or a member of a project without specified owners:

1. Visit the project page that has a member you want to remove.
2. Click **Settings** in the project's toolbar.

   Known members of the project are displayed beneath the **Members** text field, with a medium blue button representing projects or groups and a light blue button representing individual users.
3. Click the **X** next to the project id, group id, or userid you want to remove.
4. Click **Save**.

> **Warning**
> You are able to remove your own membership or ownership. Doing so could prevent you from managing the project.

Owners

A project *owner* is a Helix Core user that controls the configuration for a project. An owner does not need to be a member of a project, but once **Only Owners and Super Users can edit the project** has been set, only an owner or user with *super* privileges in Helix server can edit any project settings.

Moderators

A project *moderator* is a user assigned to moderate reviews for a specific branch associated with a project. See how to specify moderators.

When **Only Moderators can approve or reject reviews** is set for a project branch, changing the state of any review associated with the moderated branch is restricted as follows:

- Only moderators can approve or reject the review. Moderators can also transition a review to any other state.
- The review's author, when she is not a moderator, can change the review's state to **Needs Review**, **Needs Revision**, **Archived**, and can attach committed changelists.

  Normally, the review's author cannot change the review's state to **Approved** or **Rejected** on moderated branches. However, authors that are also moderators have moderator privileges, and may approve or reject their own review.

  When `disable_self_approve` is enabled, authors who are moderators (or even users with *admin* privileges) cannot approve their own reviews.
- Project members can change the review's state to **Needs Review** or **Needs Revision**, and can attach committed changelists. Project members cannot change the review's state to **Approved**, **Rejected**, or **Archived**.

- Users that are not project members, moderators, or the review's author cannot transition the review's state.
- For the review's author and project members, if a review is not in one of their permitted states, for example if the review's state is **Rejected**, they cannot transition the review to another state.

  These restrictions have no effect on who can start a review.

Remove a project

Note
Users with *super* or *admin* privileges in Helix server can always remove projects.

When a project has owners assigned, owners can remove any projects they own. Projects without assigned owners can be removed by any of their members.

Use the following steps to remove a project:

1. Visit the project page you want to remove.
2. Click **Settings** in the project's toolbar.
3. Click **Delete**.

   A tooltip appears to confirm whether you want to delete this project.
4. Click **Delete** to confirm.

7 | Code reviews

A code review is a process in which other developers can see your code and provide feedback that can suggest ways to improve the code's structure, performance, maintainability, and interaction with other code.

Benefits

Some of the benefits of code review are:

- Enforcing coding standards: code reviews can catch code that does not meet your team's coding standards. This improves the readability and consistency of your codebase.
- Knowledge and experience sharing: your team can help you learn to code better. This is particularly useful for developers new to the team.
- Early defect detection: small errors can be caught before they become problems later on.
- Code sharing: code reviews spread knowledge of the current codebase, which helps both with maintaining a mental model of the overall project, as well as defending against developer absences.
- Better personal review: knowing that someone might catch a simple coding error often increases the review developers perform of their own code.

Swarm attempts to provide these benefits without adding onerous overhead for developers.

Facilities

Swarm provides the following code review facilities. In the list, the term *author* refers to the person who creates a change to be reviewed, *reviewer* refers to any authenticated Swarm user performing code review tasks, and *required reviewer* refers to a reviewer whose up-vote is required before a review can be approved.

- Authors can request reviews, and can designate reviewers and required reviewers.
- Reviewers can start a code review on existing changes.
- Reviewers can add themselves a review to indicate that they are participating in the review and sharing responsibility for the review.
- Reviewers can provide comments to overall changes or specific lines of files, using "Markdown" on page 174 text.
- Reviewers can vote on a review, to indicate their approval or disapproval.
- Required reviewers can prevent a review from becoming approved until they up-vote the review.
- Projects with assigned moderators limit review approval to one of the moderators.

- Reviewers can mark changes as needing revision, approved, rejected, or to be archived for future consideration.
- Reviewers can commit approved changes if necessary.

Advisory nature

Currently, Swarm's default code review workflow would best be described as advisory. Swarm provides only a few mechanisms to structure or restrict code review workflows, such as:

- A required reviewer, which is any user designated by a review author, project member or moderator, or is any authenticated user that joins a review and makes their vote required, can prevent reviews from being approved until they up-vote the review. See "Required reviewers" on page 229 for details.
- Branch moderators, when configured for one or more branches in a project, prevent reviews from being approved (or rejected) without their involvement. See "Add a project" on page 188 for details.
- Administrators can optionally enable triggers in the Helix server that can *enforce* that submitted changes are associated with approved reviews, and can enable *strict* matching of the contents of a changelist to the contents of an associated, approved code review. See "Helix Core configuration for Swarm" on page 81 and "Trigger options" on page 299 for details.

Agile development teams should find sufficient capability within Swarm to make code reviews a regular part of their workflow. Swarm's development team has been using it regularly during development of Swarm. If you have ideas and suggestions for improvement, please contact us.

Models

There are three code review models: pre-commit, post-commit, and the Git Fusion model. Which model you use for code reviews with Swarm is up to you.

Pre-commit model

The pre-commit model is possible due to Helix server's *shelving* feature. Shelving enables you to temporarily make copies of your files available to other users without committing the changes into the depot. Shelving can be a very handy way for developers to create a backup, or to handle local workspace changes that might otherwise lose work in progress, without having to commit code that might destabilize a codebase.

Swarm uses the shelving feature in Helix server to manage code reviews. Shelving allows reviewers to easily acquire a copy of the code to be reviewed, and allows updates to the reviewed code prior to submission.

For more information on shelving, see "Shelving work in progress" in *Helix Versioning Engine User Guide*.

Post-commit model

The post-commit model can be used if your team's development processes preclude the use of shelving. Code must be committed to the Helix server before code review can begin, which reduces the opportunity to fix problems before, for example, a continuous integration system notices problems. However, code reviews can be started for any existing code regardless of how long it has been committed.

Git Fusion model

Perforce Git Fusion provides repo management for Git repositories, and provides workflows that enable Git and Helix Core users to collaborate on the same projects using their preferred tools.

The Git Fusion model is similar to the pre-commit model; changes in your local repo can be pushed for review to a named Helix Core branch in the Git fusion repo configuration, making your proposed changes available so that others can review and comment on them prior to committing them to the target branch. Git Fusion and Swarm work together to create a review branch and container for the pre-commit collaboration.

The Git Fusion model has several limitations that you should be aware of:

- The target branch for Git Fusion-created reviews must be a fully populated branch, and must be listed in the repo-specific Git Fusion configuration.

  See "Setting up Repos" in the *Git Fusion Guide* for details on converting a lightweight branch into a fully populated Helix Core branch.
- Reviews created with Git Fusion can only be updated from Git Fusion.
- You cannot clean up history and then push your changes to the same review. If you perform a Git rebase, you should push your changes as a new review.
- A Git Fusion review does not currently display the individual task branch commits that make up the review. Only the merged commit diffs are shown.

For more information on Git Fusion, see *Git Fusion Guide*.

Internal representation

Swarm-managed changelists

A code review consists of one or more shelved changelists that Swarm manages. A shelved changelist is a pending changelist that has a snapshot of its files on a shelf associated with the changelist.

When a review is started, Swarm creates a new changelist that becomes the review changelist. What happens afterwards varies:

- If the review contains uncommitted work (the pre-commit model), Swarm copies the shelved files from the user's changelist that initiated the review into the review's changelist.

- Any time that a user's changelist associated with the review has its shelved files updated, Swarm copies the shelved files into its review changelist and creates an archive changelist. An archive changelist is no different from any other pending changelist with shelved files, but it allows Swarm to provide versioning and diffs within a review.
- If the head version of a review is committed (the post-commit model), the review's changelist is emptied of files.

The review's changelist is never actually committed; this allows the review to be opened later with additional shelved changes.

Important
Swarm's managed review changelists should only be deleted if you are uninstalling Swarm.

Swarm's review changelists maintain the history of a review and all of its feedback. The deletion of a Swarm shelved changelist causes instability and potentially data loss, and represents a scenario that can be very challenging to recover from, even with the engagement of Perforce consultants.

You can display a list of all of the Swarm-managed changelists using the `p4 changelists` command:

```
$ p4 changelists -u swarm
Change 1212285 on 2015/07/31 by swarm@swarm-96017af4-5615-9819-7af1-
6fc1fa537214 *pending* 'Add requirements and instructions'
Change 1212284 on 2015/07/31 by swarm@swarm-96017af4-5615-9819-7af1-
6fc1fa537214 *pending* 'Add requirements and instructions'
...
```

swarm is the userid with *admin*-level privileges within the Helix server that Swarm is configured to use. Use the appropriate userid when you run the `p4 changelists` command.

Swarm-managed workspaces

Whenever Swarm creates a changelist for a review, it uses a client workspace (or just workspace) associated with the configured Helix Core userid that has *admin* privileges. Whenever a user commits a change via Swarm's user interface, Swarm uses a workspace associated with that user.

To learn more about workspaces, see the section "Helix Core as a version control implementation" in *Solutions Overview: Helix Version Control System*.

The workspaces that Swarm creates and uses live in the *SWARM_ROOT*`/data/clients` folder.

Inside the clients folder, Swarm maintains a user-specific folder that contains any workspace folders that may be required. Each user-specific folder is named by converting their Helix Core userid into hexadecimal to avoid any characters that would be problematic in the filesystem, such as slashes, accents, UTF-8 characters, etc. For example, the folder for the user eedwards would be named `6565647761726473`.

Within the user-specific folder are the folders that become the root of each workspace. Each of these folders is named with a globally-unique identifier (GUID) prefixed with `swarm-`, for example `swarm-438d482b-f107-9a35-c06c-86ac68136b00`. Accompanying each folder is a lock file with the same name plus the .lock extension. Finally, the user-specific clients folder contains a management lock file called `manage.lock`.

Here is an example of the folder structure:

```
SWARM_ROOT/
        data/
                clients/
                  6565647761726473/
                        manage.lock
                        swarm-438d482b-f107-9a35-c06c-86ac68136b00/
                        swarm-438d482b-f107-9a35-c06c-86ac68136b00.lock
                        swarm-8388362a-233d-0cb9-3e90-895eaaa99f6c/
                        swarm-8388362a-233d-0cb9-3e90-895eaaa99f6c.lock
                  736c6f7264/
                        manage.lock
                        swarm-da7de4b4-0ecb-12c8-1b35-f3e32bb18033/
                        swarm-da7de4b4-0ecb-12c8-1b35-f3e32bb18033.lock
```

Here are the steps Swarm takes when it needs to use a client:

1. Convert the current connection's userid to hexadecimal.
2. Check to see whether a user-specific folder exists within *SWARM_ROOT*`/data/clients`; if not, create the folder.

3. Within the user-specific folder, loop over any existing workspace folders and attempt to lock each in turn:

   If a lock is acquired skip to the next step. Otherwise, perform the following procedure.

   Create workspace procedure:

   a. Check if the max number of clients for the current user has been reached:
      - If so, wait a short amount of time (50 milliseconds), and start step 3 again.
      - If not, proceed to the next step.
   b. Take a lock on `manage.lock`.
   c. Check if the max number of clients for the current user has been reached:
      - If so, release the `manage.lock`, wait a short amount of time (50 milliseconds), and start step 3 again.
      - If not, proceed to the next step.
   d. Create a new workspace folder using a GUID-based filename, and take a lock on the folder.
   e. Release the `manage.lock` lock.
4. Perform the necessary file operations using the locked workspace folder.
5. Revert the file content within the workspace folder to avoid having constantly growing disk space use.

   Note
   There may occasionally be stray files left; Swarm is not aggressive about cleaning up.

6. Swarm releases the lock on the workspace folder.

Most users should only require 1-2 workspaces, and those are only required if they commit from Swarm. The *admin* user that Swarm is configured to use should only use one workspace per configured worker.

By default, the number of workspaces that could be active at any given instant is two times the number of configured workers. Since the default worker count is three, Swarm would use at most six workspaces simultaneously.

If the workspace limit is reached, further file processing is blocked until a workspace becomes available. Potentially, this means that users could encounter timeouts. Configuring Swarm to use more workers could solve that issue.

Removal considerations

Administrators might wish to remove Swarm-managed workspaces. There are a few considerations that should be assessed prior to removal:

- Ideally, you should stop the web server (taking Swarm out of service) before removing a Swarm-managed workspace from the Swarm server; this eliminates the risk of removing a workspace that is in use.

If you do not stop the web server first, Swarm may encounter an error during a submit.

- Removal of a Swarm-managed workspace folder does not remove the client spec from the Helix server. Unless the client spec is removed, that workspace effectively becomes orphaned. Orphaned clients are, of themselves, not a big concern as the storage and performance impact is negligible.
- Removal of a Swarm-managed workspace's corresponding client spec in the Helix server can be done. However, **you should never remove a client spec that has associated shelved files**.

  Usually, the only client specs that should have associated shelved files belong to the *admin* account that Swarm is configured to use. All other workspaces that may exist for other users are primarily used for submitting changes, and so should not have shelved files associated.

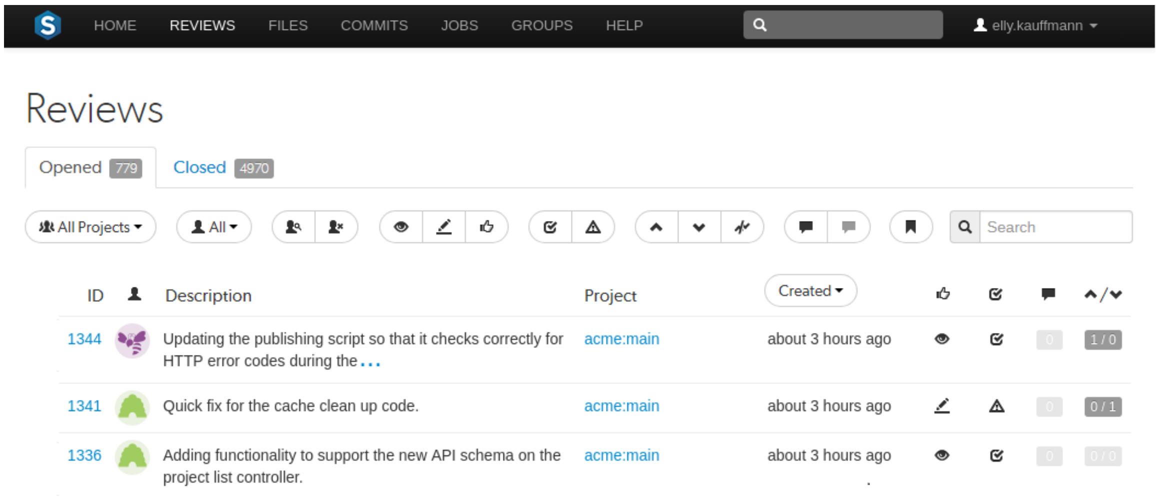
Review queues

Code review queues help you keep track of code reviews that:

- Have been requested and are awaiting review
- Are underway
- Have been accepted, rejected, or archived

To see all available reviews, click the **Reviews** link in the main toolbar.

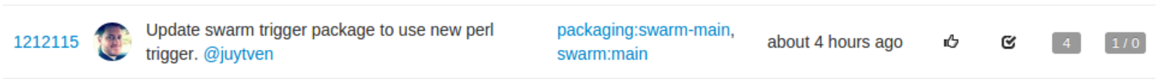
The **Reviews** page lists open and closed reviews for all projects in the Helix server.



ID		Description	Project	Created				
1344		Updating the publishing script so that it checks correctly for HTTP error codes during the ...	acme:main	about 3 hours ago			0	1 / 0
1341		Quick fix for the cache clean up code.	acme:main	about 3 hours ago			0	0 / 1
1336		Adding functionality to support the new API schema on the project list controller.	acme:main	about 3 hours ago			0	0 / 0

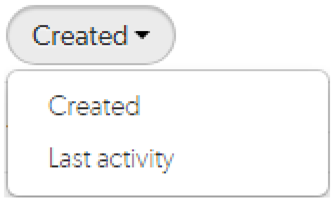
Opened reviews, those that have just begun, are being reviewed, or are awaiting further changes, are displayed on the **Opened** tab. Closed reviews, those that have been completed successfully, have been rejected, or have been archived (where it might be unclear of their benefit), are displayed on the **Closed** tab.

Each review queue entry displays the following information:


1212115 Update swarm trigger package to use new perl trigger. @juytven packaging:swarm-main, swarm:main about 4 hours ago 4 1 / 0

- The review id
- The avatar of the review's author
- The review's description
- The associated project's name and branch
- When the review was created or when the review was last updated depending on the **Result order** button setting.

  When review results are older than 24 hours they are displayed in numerical order within each day.

  To change the sort order, click the **Result order** button and select **Created** or **Last activity** from the dropdown menu:

  - **Created**: Reviews sorted by when they were created
  - **Last activity**: Reviews sorted by when they were last updated

  Note
  - If the **Result order** button is not displayed reviews are sorted by when they were created.
  - **Result order** button display is a global setting controlled by the Swarm administrator. See "Reviews filter" on page 281 for details.

- An icon indicating the current review state
- An icon indicating the test suite state
- A counter for the number of open (non-archived) comments that are associated with the review. Hover your mouse over the comment count to display a tooltip showing the number of archived comments associated with the review.
- An indicator showing the number of up votes and down votes

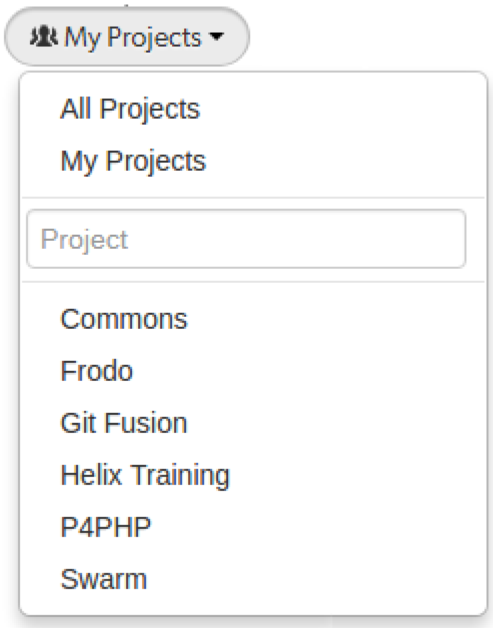
Note
Hover your mouse over any of the icons to see tooltips.

"Projects" on page 158 and "Groups" on page 155 have their own review queues that display reviews created by their members.

Filtering open reviews

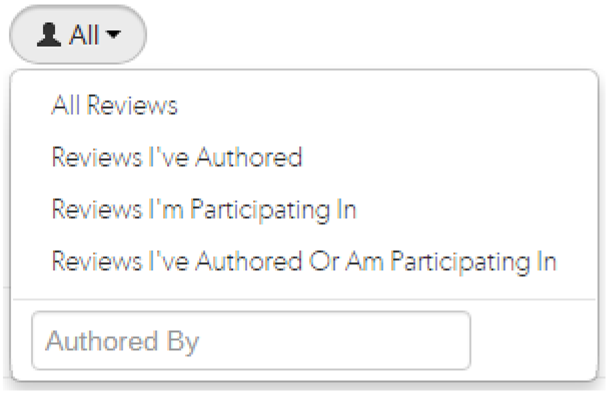
The **Opened** tab presents a list of all code reviews that have started, are being reviewed, are awaiting revisions, or need to be committed. The following filtering options are available for opened code reviews:

- **Project:** a dropdown menu that lets you filter which reviews to display based on project:



  - **All Projects:** all reviews are displayed for all projects.
  - **My Projects:** all reviews for all of the projects you are participating in, as a member, owner, moderator, or follower.
  - **Project Search:** An auto-complete search field that allows you to choose one of the projects defined in Helix server. Once specified, only reviews for the selected project are displayed. Click the **X** button to remove a projectid after it has been specified.
  - **Select Project:** selecting a project name displays only reviews for that project.

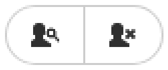
- **Users:** a dropdown menu that lets you filter which reviews to display based on user involvement:


All
All Reviews
Reviews I've Authored
Reviews I'm Participating In
Reviews I've Authored Or Am Participating In
Authored By

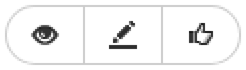
  - **All Reviews:** displays all reviews.
  - **Reviews I've Authored**: displays reviews that you have authored.
  - **Reviews I'm Participating In:** displays reviews that you are a reviewer of, but not an author of.
  - **Review I've Authored Or Am Participating In:** displays reviews that you have authored, or are a reviewer of.
  - **Specific User:** An auto-complete search field that allows you to choose one of the user accounts defined in the Helix server. Once specified, only reviews authored by the user are displayed. Click the **X** button to remove a userid after it has been specified.

  When you select one of the available options, the list of options updates to match the currently selected filter, and the **Users** dropdown indicates the current filter: **All**, **Author**, **Participant**, or ***userid***.

- **Has Reviewers** or **No Reviewers:** displays reviews that have one or more reviewers, or reviews that have no reviewers.

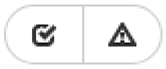


- **Review state:** one of the following (left to right):



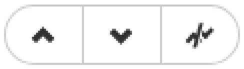
  - **Needs review:** the review's changes need to be reviewed.
  - **Needs revision:** the review's changes have been reviews, but further revisions are required before the review can be accepted.
  - **Approved:** the review's changes have been approved, and should be committed.

- **Test status:** one of the following:

  - **Tests pass:** when automated tests are enabled for the associated project, and the test suite execution succeeds, Swarm updates the review accordingly.
  - **Tests fail:** similar to the Tests pass state, except that the test suite execution has failed. Check with your test suite to determine why the tests failed.

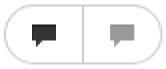
- **Vote status:** one of the following:

  - **Voted up:** I have voted the review up.
  - **Voted down:** I have voted the review down.
  - **Not voted:** I am a participant but have not voted on the review.

    Filters for voting only apply to reviews which you are a participant of. Commenting on or voting on a review will automatically add you as a participant. If you leave the review after commenting on it, then this review will not be included in the list.

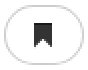
- **Comment status:** one of the following:

  - **Has comments:** I have commented on the review.
  - **Does not have comments:** I have not commented on the review.

    Filters for commenting only apply to reviews which you are a participant of. Commenting on or voting on a review will automatically add you as a participant. If you leave the review after commenting on it, then this review will not be included in the list.

- **Bookmark:** review filters can be remembered by bookmarking the page, and this icon acts as a reminder of that.

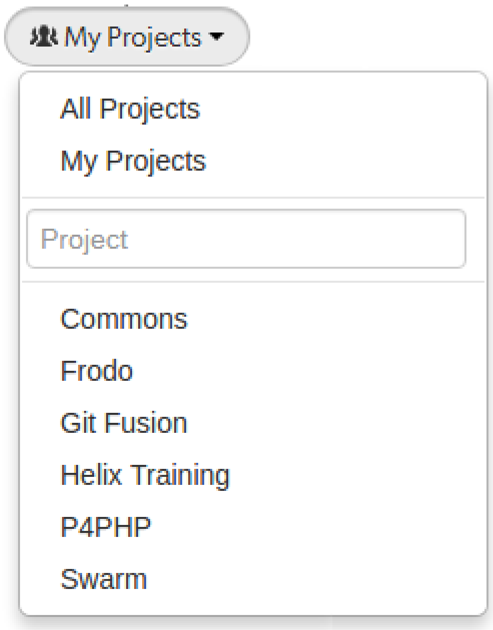
- **Search term:** where review descriptions match your search string.

Swarm updates the URL in your browser to reflect filtering options. This makes it easy to bookmark or share review queue URLs, and it maintains the current filtering if you click on a review and then use your browser's back button to return to the review queue.

Filtering closed reviews

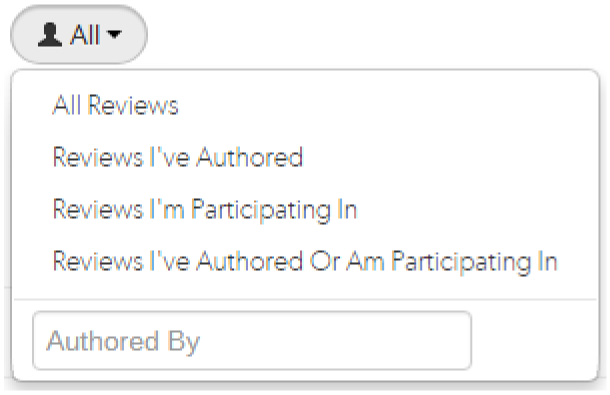
The **Closed** tab presents a list of all code reviews that have been approved and committed, rejected, or archived. The following filtering options are available for closed code reviews:

- `Project:` a dropdown menu that lets you filter which reviews to display based on project:



  - **All Projects:** all reviews are displayed for all projects.
  - **My Projects:** all reviews for all of the projects you are participating in, as a member, owner, moderator, or follower.
  - **Project Search:** An auto-complete search field that allows you to choose one of the projects defined in Helix server. Once specified, only reviews for the selected project are displayed. Click the **X** button to remove a projectid after it has been specified.
  - **Select Project:** selecting a project name displays only reviews for that project.

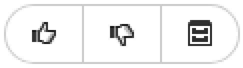
- **Users:** a dropdown menu that lets you filter which reviews to display based on user involvement:


All
All Reviews
Reviews I've Authored
Reviews I'm Participating In
Reviews I've Authored Or Am Participating In
Authored By

  - **All Reviews:** displays all reviews.
  - **Reviews I've Authored**: displays reviews that you have authored.
  - **Reviews I'm Participating In:** displays reviews that you are a reviewer of, but not an author of.
  - **Review I've Authored Or Am Participating In:** displays reviews that you have authored, or are a reviewer of.
  - **Specific User:** An auto-complete search field that allows you to choose one of the user accounts defined in the Helix server. Once specified, only reviews authored by the user are displayed. Click the **X** button to remove a userid after it has been specified.

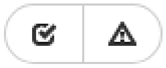
  When you select one of the available options, the list of options updates to match the currently selected filter, and the **Users** dropdown indicates the current filter: **All**, **Author**, **Participant**, or ***userid***.

- **Review state:** one of the following (left to right):



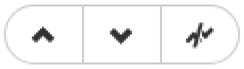
  - `Approved:` the review's changes have been approved, and should be committed.
  - `Rejected:` the review's changes have been rejected.
  - `Archived:` the review's changes have been put aside.

- **Test status:** one of the following:

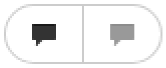


  - `Tests pass:` when automated tests are enabled for the associated project, only reviews where the tests have passed are displayed.
  - `Tests fail:` similar to the Tests pass state, except that only reviews where the tests have failed are displayed.

- **Vote status:** one of the following:

  - `Voted up:` I have voted the review up.
  - `Voted down:` I have voted the review down.
  - **Not voted:** I am a participant but have not voted on the review.
- **Comment status:** one of the following:

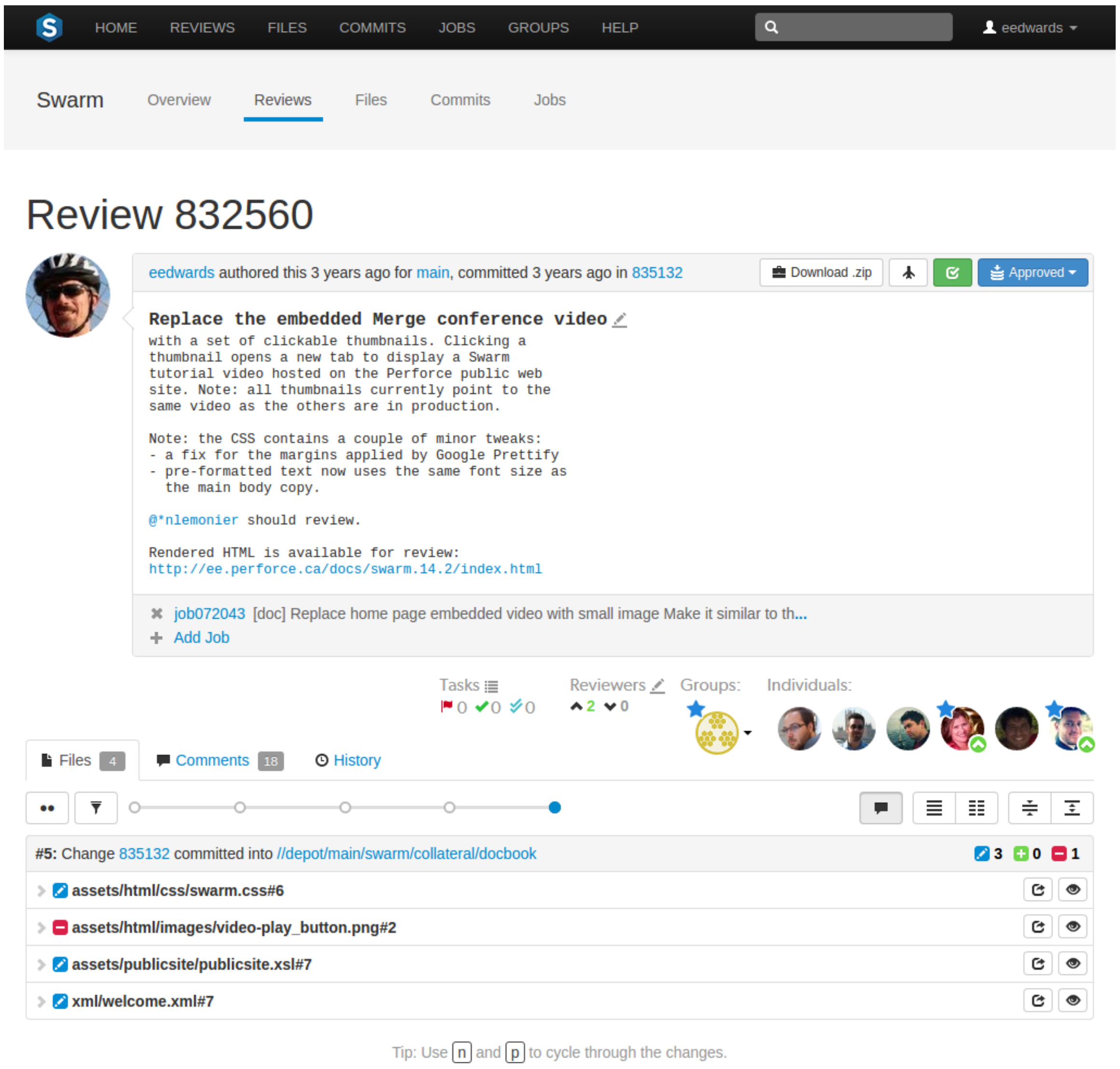
  - **Has comments:** I have commented on the review.
  - **Does not have comments:** I have not commented on the review.
- **Bookmark:** review filters can be remembered by bookmarking the page, and this icon acts as a reminder of that.

- **Search term:** where review descriptions match your search string.

Swarm updates the URL in your browser to reflect filtering options. This makes it easy to bookmark or share review queue URLs, and it maintains the current filtering if you click on a review and then use your browser's back button to return to the review queue.

Review display

During a code review, reviewers spend most of their time using the review interface:



Review 832560

eedwards authored this 3 years ago for main, committed 3 years ago in 835132

```
Replace the embedded Merge conference video
with a set of clickable thumbnails. Clicking a
thumbnail opens a new tab to display a Swarm
tutorial video hosted on the Perforce public web
site. Note: all thumbnails currently point to the
same video as the others are in production.

Note: the CSS contains a couple of minor tweaks:
- a fix for the margins applied by Google Prettify
- pre-formatted text now uses the same font size as
  the main body copy.

@*nlemonier should review.

Rendered HTML is available for review:
http://ee.perforce.ca/docs/swarm.14.2/index.html
```

job072043 [doc] Replace home page embedded video with small image Make it similar to th...

Add Job

#5: Change 835132 committed into //depot/main/swarm/collateral/docbook 3 0 1

- assets/html/css/swarm.css#6
- assets/html/images/video-play_button.png#2
- assets/publicsite/publicsite.xsl#7
- xml/welcome.xml#7

Tip: Use n and p to cycle through the changes.

The review interface is very similar to the changelist interface; and provides largely the same functionality, but has several notable differences that are described in the following sections.

Test status

When continuous integration has been configured for a project, test success or failure is indicated in the review's heading. If your continuous integration tests can provide a URL that provides details of a test run, the indicator becomes linked; click the indicator to see the test details.

When you hover your mouse over the test status indicator, Swarm indicates the test status and how long ago that status was achieved.

Deployment status

When automated deployment has been configured for a project, the deployment success or failure is indicated in the review's heading. If your deployment program can provide a URL that provides details of the deployment, the indicator becomes linked; click the indicator to see the deployment results.

Review state

The **Request Review** button is replaced with the review state drop-down button that indicates the review's current disposition. See "States" on page 231 for more information.

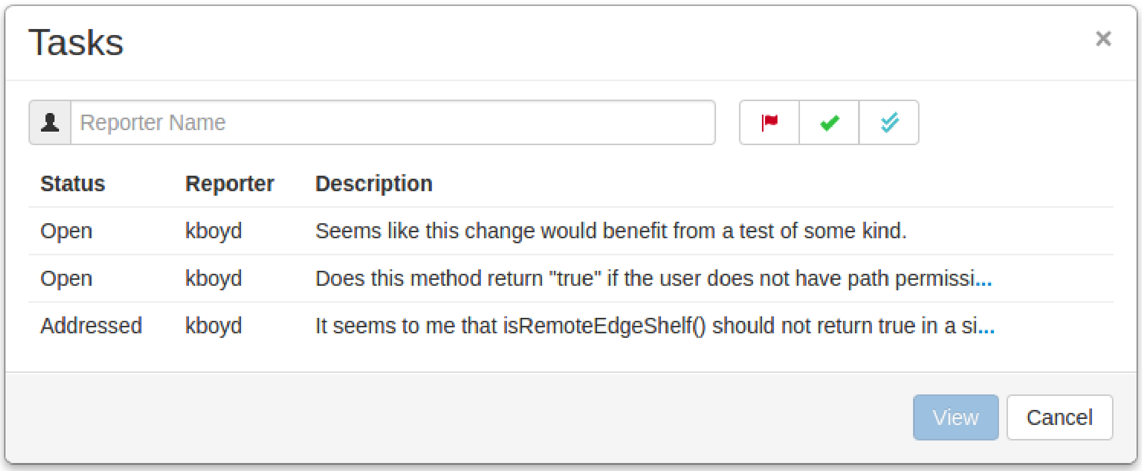
Edit description

An **Edit Description** icon appears in the review's description, which allows you to update the description to reflect any updates have been made during the review.

Tasks

A *Tasks* area appears below the review's description. This area summarizes the number of comments that have been flagged as tasks, with separate counts for open, addressed, and verified tasks. See "Tasks" on page 144 for more details.

Click the Task list button to display a dialog listing all tasks associated with the review:



Status	Reporter	Description
Open	kboyd	Seems like this change would benefit from a test of some kind.
Open	kboyd	Does this method return "true" if the user does not have path permissi...
Addressed	kboyd	It seems to me that isRemoteEdgeShelf() should not return true in a si...

Within the **Tasks** dialog, you can filter the tasks by the reporter (the userid of the user who created the task), and/or by task state; click:

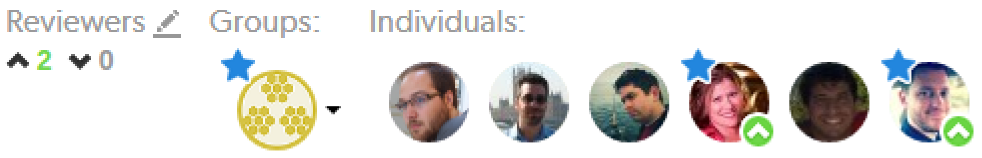
- The **red flag** button to display only open tasks (comments that need to be addressed).
- The **green check mark** button to display only addressed tasks (comments that have been addressed).
- The **blue double-check** mark button to display only verified tasks (comments that have been addressed and verified).

Note
Archived tasks do not appear in the Tasks dialog.

To view a particular task, click once on a task within the **Tasks** dialog to select it, and then click the **View** button. Or, double-click on a task. Either way, the **Tasks** dialog closes and Swarm adjusts the review display so that the chosen task is in view.

Reviewers

A *Reviewers* area appears below the review's description whenever a review has one or more reviewers, or you are logged in.



This area includes, from left to right, the edit reviewers button, the current up and down vote counts, the avatars of current reviewer groups, and the avatars of the current reviewers.

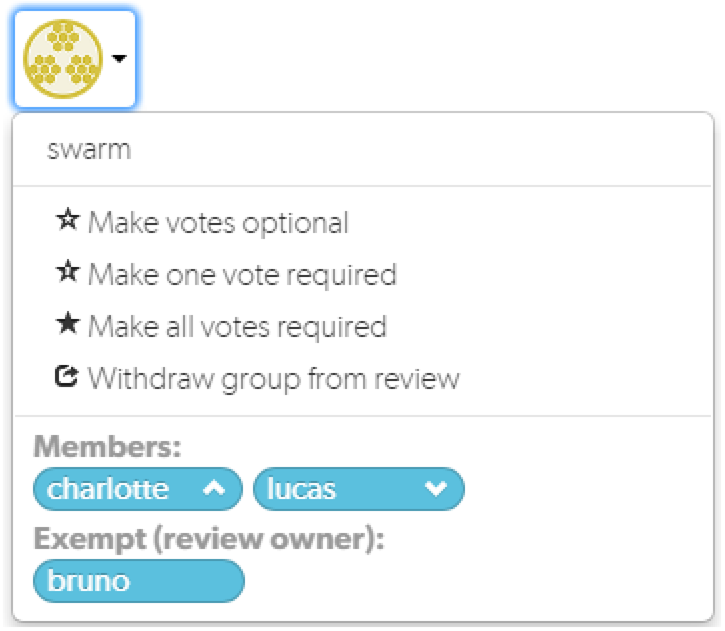
Note
By default, reviewer group members are not displayed in the *Individuals* area of the reviews page when they interact with a review (vote, comment, update, commit, archive, etc.). This avoids overloading the *Individuals* area with individual avatars if you have large reviewer groups.

An exception to this behavior is when a member of a reviewer group is also an individual required reviewer, in this case their avatar will be displayed in the *Individuals* area.

See "Expand group reviewers" on page 285 for details on displaying reviewer group members when they interact with a review.

Group reviewer

If you are a member of a group that is a reviewer on the review, click your individual avatar to vote on the review. See the "Individual reviewer" on page 216 table for details. Your vote is registered for the group, and is also displayed on your individual avatar. Click on the group avatar to see how individual members have voted on the review:



You must be the review author, a project member, a project moderator, or a user with super privileges to change the group settings:

- **Make votes optional:**
  - If any group member votes down the review, the group avatar displays a badge indicating that the group has voted down the review.
  - If at least one group member votes up the review and **no** members of the group vote down the review, the group avatar displays a badge indicating that the group has voted up the review.
- **Make one vote required:** Indicated by a star badge with a 1 over the group avatar.
  - If any group member votes down the review, the group avatar displays a badge indicating that the group has voted down the review.
  - If at least one group member votes up the review and **no** members of the group vote down the review, the group avatar displays a badge indicating that the group has voted up the review. See Required reviewers for details.
- **Make all votes required:** Indicated by a star badge over the group avatar.
  - If any group member votes down the review, the group avatar displays a badge indicating that the group has voted down the review.
  - If all the group members vote up the review and **no** members of the group vote down the review, the group avatar displays a badge indicating that the group has voted up the review. See Required reviewers for details.
- **Withdraw group from review:** Removes the group from the review.

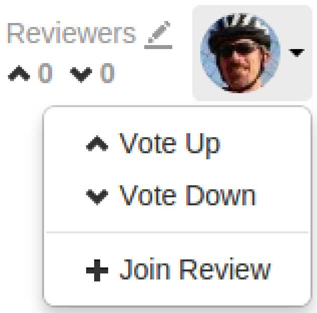
Individual reviewer

When an individual reviewer has voted on a review, their avatar displays a badge indicating whether they voted up or down. Required reviewers have a star badge over their avatar.

If you are logged in and viewing a review you did not author, your avatar appears to the right of any other reviewers and its appearance varies according to the following conditions:

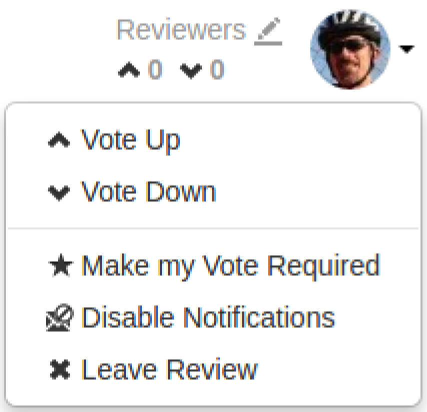
When you are not yet a reviewer, or you are a member of a reviewer group who has not yet voted: Clicking your avatar presents a menu allowing you to vote up or down and thereby become an individual reviewer, or simply join the review as an individual reviewer without voting.

When a member of a reviewer group votes they automatically become an individual reviewer as well.



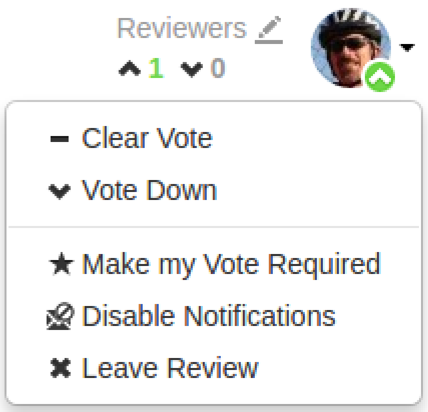
When you are an individual reviewer who has not yet voted: No badge appears on your avatar and clicking your avatar presents a menu allowing you to vote up or down, change whether your individual vote is required or not, or leave the review.

If you are also a member of a reviewer group and you click **Leave Review**, you will remain a member of the reviewer group.



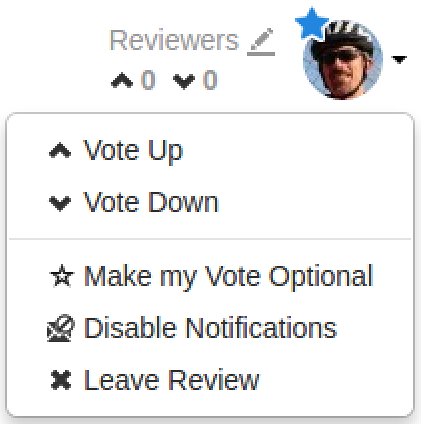
When you are an individual reviewer who has voted, or a member of a reviewer group who has voted: Your avatar displays a badge indicating your vote. Clicking your avatar presents a menu allowing you to clear or change your vote, change whether your individual vote is required or not, or leave the review.

If you are also a member of a reviewer group and you click **Leave Review**, your vote is cleared but you will remain a member of the reviewer group.



When you are a required reviewer, or a member of reviewer group that requires all votes: Your avatar displays a star badge indicating that the review cannot be approved until you vote up. Clicking your avatar presents a menu allowing you to change your vote, make your vote optional, or leave the review.

If you are also a member of a reviewer group and you click **Leave Review**, you will remain a member of the reviewer group. **Make my Vote Optional** is not available for members of "all votes required" reviewer groups.



When a review is updated, if the review's list of files, file content, or file-types changes, any votes cast on the review become *stale*. The vote counts are reset, and the vote indicators become muted.

If you hover your mouse over a reviewer with a stale vote, a tooltip appears displaying the userid, how they voted, and on which version of the review; each version is represented as a point on the "Review timeline" on the facing page.

Note
Stale vote handling is not supported for Git-created reviews.

Disable notifications

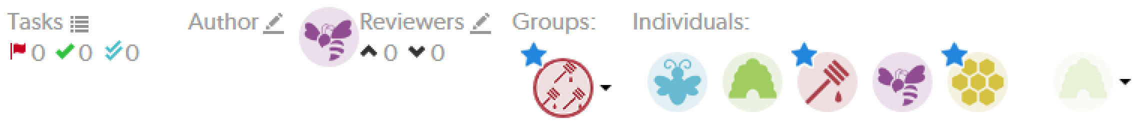
When you become a review participant, by joining the review or being @mentioned in a comment or in the review's description, you receive notifications for any events associated with the review. If you find that the notifications become more of a burden than benefit and you wish to continue being a review participant, you can disable notifications:

1. Click your avatar in the reviewers area to display your reviewer options.
2. Click **Disable Notifications**.

Once notifications are disabled, you no longer receive notifications. However, if you are @mentioned in a subsequent review comment, you do receive a notification for that comment; regular notifications remain disabled. This approach ensures that you don't miss anything that other reviewers or the review author deems important.

Changing the review author

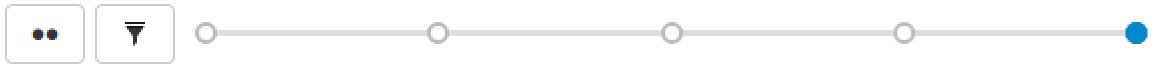
If the configuration option to allow users to change the author of a review is enabled, then an extra icon is displayed to the left of the Tasks and Reviewers icons. By default this option is disabled, so talk to your Swarm Administrator if you require this feature.



If the author icon is displayed, then clicking the edit icon will open a dialog that allows the author of this review to be changed. This is useful if the original author is no longer available, or ownership has passed to a different developer.

Review timeline

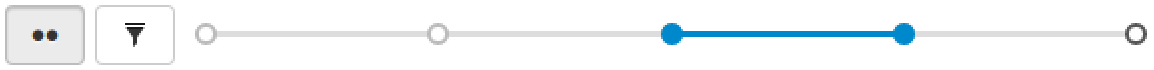
A slider control, called the Review Timeline, appears just above the list of files. If the review's files have been updated at least once, this slider allows you to browse and compare arbitrary versions of the review's files by dragging the version point(s) on the slider.



A Review Timeline tooltip Each point on the slider represents a version of the review's files, with the oldest version on the left and newer versions on the right. Hover your mouse over each point to see a tooltip displaying the version number, who created it and when, plus the changelist containing a copy of that version's files.

If you adjust the slider while files within the review are expanded, Swarm keeps those same files expanded whenever new versions need to be displayed.

The button to the left of the slider with two dots toggles *diff* mode for arbitrary versions of the review's files. In diff mode, the slider shows a bar indicating which two versions are being compared, and the endpoints of the slider can be dragged to any of the available points to compare any earlier version of the review with any later version.



Hovering your mouse over the bar shows a tooltip displaying the versions being compared, and the changelists that contain the files. When a review has many, many versions, Swarm maintains a minimum distance between version dots, which then requires horizontal scrolling.

Dragging one of the version points to the edge of the timeline display area causes the timeline to scroll in the direction of the drag motion, until the end of the timeline is reached. Alternately, click either the ❮ button or ❯ buttons, or drag the scrollbar itself, to adjust the scroll position.

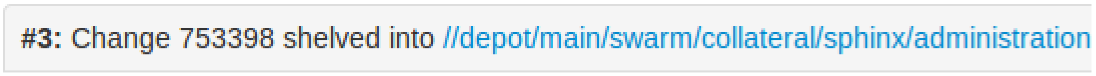
When you release the version point, Swarm adjusts the scroll amount to place the version dots within view, when possible.

Note
It is good to remember that a review consists of one or more Swarm-managed changelists. When comparing versions of a review, Swarm is showing any differences between the selected versions, not the review author's personal changelist. See "Internal representation" on page 200 for details.

For involved reviews that have many revisions and many comments, it can sometimes be difficult to determine whether the points expressed in comments have been addressed. To help with such situations, click the **Filter comments** icon to limit the displayed comments to the current review version.

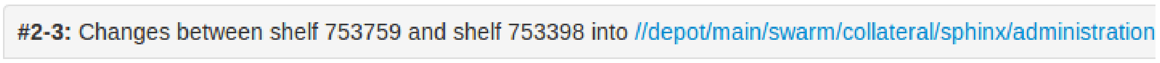
File listing

The file listing header displays:


#3: Change 753398 shelved into //depot/main/swarm/collateral/sphinx/administration

- The current version of the files in the review.
- Which changelist contains a shelved copy of the review's files.
- The common path for all of the files in the review.

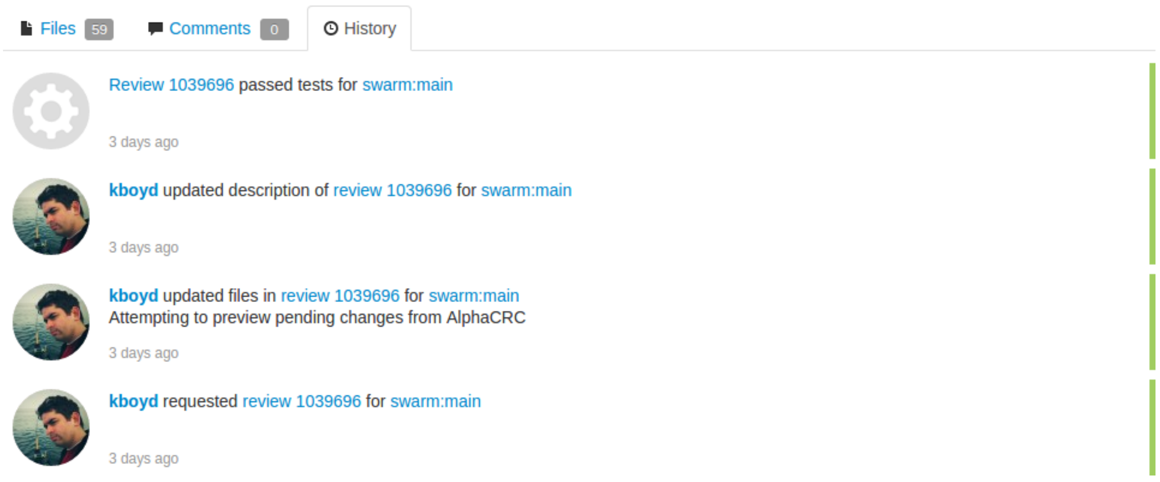
In diff mode, the file listing header displays:


#2-3: Changes between shelf 753759 and shelf 753398 into //depot/main/swarm/collateral/sphinx/administration

- Which two versions of the review's files are being compared.
- Which changelists contain the files being compared.
- The common path for all of the files in both versions of the review.

History tab

The **History** tab presents a list of the events that affect this review, including:



Review 1039696 passed tests for swarm:main

3 days ago

kboyd updated description of review 1039696 for swarm:main

3 days ago

kboyd updated files in review 1039696 for swarm:main
Attempting to preview pending changes from AlphaCRC

3 days ago

kboyd requested review 1039696 for swarm:main

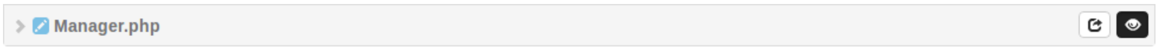
3 days ago

- When the review was started
- When a new reviewer joins the review
- When the review's state changes
- When the review's files are updated
- When a reviewer votes on the review
- When someone comments on the review, or one of its files
- When tests pass or fail, provided continuous integration is configured

Mark as read

Beside each file in a review is a **Mark as Read** button, which help you keep track of which files you have reviewed. The read flag is remembered independently for each user. If the content of a file is changed in an update to the review, the read flag automatically clears. This is particularly useful when a code review consists of many files.

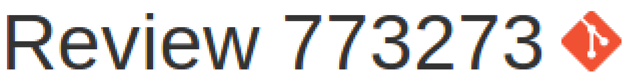
When clicked, the button's colors invert and the associated file is visually muted, to make it easy to distinguish read files from unread files:



If a file has been marked as read, click the button a second time to reset the status to unread.

Identifying reviews created with Git Fusion

Git Fusion-initiated reviews include the Git logo beside the main review identifier. This indicator is important because Helix Core users cannot update Git Fusion-initiated reviews.



Activities

This section describes the major activities that affect code reviews, including starting a review, updating a review, and fetching a review's files.

Start a review

To start a code review, choose one of the following approaches:

Important
If your Helix server is configured as a commit-edge deployment, and your normal connection is to an edge server, Swarm refuses to start reviews for shelved changes that have not been promoted to the commit server.

Within Swarm, this means that the **Request Review** button does not appear for unpromoted shelved changes. Outside of Swarm, attempts to start reviews for unpromoted shelved changelists appear to do nothing. Ask your Helix server administrator for assistance if you cannot start a review.

An administrator of the Helix server can automatically promote shelved changes to the commit server by setting the configurable `dm.shelve.promote` to `1`.

- When you use Swarm to view a submitted changelist, click the **Request Review** button to request a review of that changelist. This uses the post-commit model.
- When you use Swarm to view a shelved changelist, click the **Request Review** button to request a review of that shelved changelist. This uses the pre-commit model.

- When you are about to shelve or submit files:
  1. Include #review within your changelist (separated from other text with whitespace, or on a separate line).

     Once the review begins, Swarm replaces #review with #review-12345, where 12345 is the review's identifier.

     Note
     The `#review` keyword is customizable. For details, see "Review keyword" on page 279.

  2. At this time, you can add reviewers to the code review by using @mention for users, and @@mention for groups in the changelist description for each desired reviewer.

     If your `@mention` or `@@mention` includes an asterisk (`*`) before the *userid* or *groupid*, for example `@*userid`, that user or all of the group members become required reviewers. If your `@@mention` includes an exclamation mark (`!`) before the *groupid*, for example `@@!groupid`, the members of that group become required reviewers but only one member of the group is required to vote. See "Required reviewers" on page 229 for details.

  3. Complete your shelve or submit operation.

  Warning
  If you shelve a changelist and subsequently edit the description to include #review, a review is not started. You must re-shelve the files after adding #review.

- When you are using Git Fusion, you can start a review by pushing your changes to a target branch using the following command:

  ```
  $ git push origin task1:review/master/new
  ```

  task1 is the name of the current Git task branch, and ***master*** is the target branch that the proposed changes are intended for.

  Important
  The target branch must be mapped to a named Helix Core branch in the Git Fusion repo configuration.

  See "Setting up Repos" in the *Git Fusion Guide* for details on converting a lightweight branch into a fully populated Perforce branch.

  When the command completes, the output indicates the ***review id*** that has been created:

  ```
  remote: Perforce: Swarm review assigned: review/master/1234
  ```

  where *1234* is the review id that was just created.

  For more information on Git Fusion, see the *Git Fusion Guide*.

Update a review

To update a code review, use one of the following approaches:

- For a *pre-commit* review that you authored:
  1. edit the files
  2. shelve the files

  You can repeat these steps as many times as necessary.
- For a *post-commit* review, or a review where you are not the author:
  1. fetch the review's files into a new changelist
  2. edit the files
  3. update the changelist's description to include `#review-12345` (separated from other text with whitespace, or on a separate line)
  4. shelve the changelist's files

  Once these steps are complete, further updates involve editing the files, and then shelving the changelist's files.

  Warning
  If you use an invalid review identifier, it will appear that nothing happens. Swarm is currently unable to notify you of this situation.

- When you are working with Git Fusion:

Important
You can only update Git Fusion-initiated reviews using Git Fusion.

In the following example, the current Git task branch is ***task1***, the target branch is ***master***, the review id is ***1234***, the Git Fusion hostname is ***gfserver***, and the remote repo name is ***p4gf_repo***.

1. Fetch the review's head version:

```
$ git fetch --prune origin
From gfserver:p4gf_repo
* [new_branch]      review/master/1234 ->
origin/review/master/1234
x [deleted]         (none)     -> origin/review/dev/new
```

   The **`--prune`** option lets the local Git repo delete the unwanted **review/*master*/new** reference created by the initial **git push origin *task1*:review/*master*/new** command.

2. Check out the review's head version:

```
$ git checkout review/master/1234
```

3. Edit the files as required.
4. Add the edited files to the index of files, in preparation for the next commit.

   There are several ways to do this. For example, to add all modified files to the index, run:

```
$ git add -A
```

5. Commit the files in Git:

```
$ git commit -m "made some changes"
```

6. Push the Git changes to the review:

```
$ git push origin review/master/1234
```

Note
If you get review feedback that is better expressed as a Git rebase and cleaned up history, you can make your changes and push them as a new review.

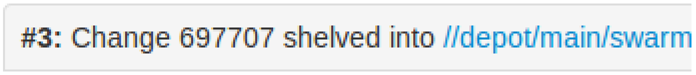
You **cannot** clean up history and then push your changes to the *same* review.

For more information on Git Fusion, see the *Git Fusion Guide*.

Fetch a review's files

First, determine the changelist containing the review's files:

1. Visit the review's page.
2. The current review version's changelist appears in the file list heading:

   #3: Change 697707 shelved into //depot/main/swarm

   In this example, the changelist is `697707`. You use the identified changelist in place of shelved changelist below.
3. Decide whether you will use p4, P4V, or Git Fusion to fetch the files, and follow the instructions in the appropriate section below.

Using P4

1. For a **shelved changelist**, use a command-line shell and type:

   ```
   $ p4 unshelve -s shelved changelist
   ```
2. For a committed changelist, use a command-line shell and type:

   ```
   $ p4 sync @committed changelist
   ```

Note
Your client's view mappings need to include the changelist's path.

Using P4V

For a shelved changelist:

1. Select **Search > Go To**.
2. Change the select box to **Pending Changelist**.
3. Type in the *shelved changelist* number and click **OK**.
4. Select the files in the **Shelved Files** area.
5. Right-click and select **Unshelve**.
6. Click **Unshelve**.

For a committed changelist:

1. Select **Search > Go To**.
2. Change the select box to **Submitted Changelist**.
3. Type in the *submitted changelist* number and click **OK**.

4. Select the files in the **Files** area.
5. Right-click and select **Get this Revision**.
6. Click **Close**.

Using Git Fusion

In the following example, the current local task branch is `task1`, the target branch is `master`, the review id is `1234`, the Git Fusion hostname is `gfserver`, and the remote repo name is `p4gf_repo`.

1. Fetch the review's head version:

   ```
   $ git fetch --prune origin
   From gfserver:p4gf_repo
   * [new_branch]      review/master/1234 ->
   origin/review/master/1234
   x [deleted]         (none)     -> origin/review/dev/new
   ```

   The `--prune` option lets the local Git repo delete the unwanted `review/master/new` reference created by the initial git `push origin task1:review/master/new` command.

2. Check out the review's head version:

   ```
   $ git checkout review/master/1234
   ```

Important
You can only update Git Fusion-initiated reviews using Git Fusion.

For more information on Git Fusion, see the *Git Fusion Guide*.

Edit reviewers

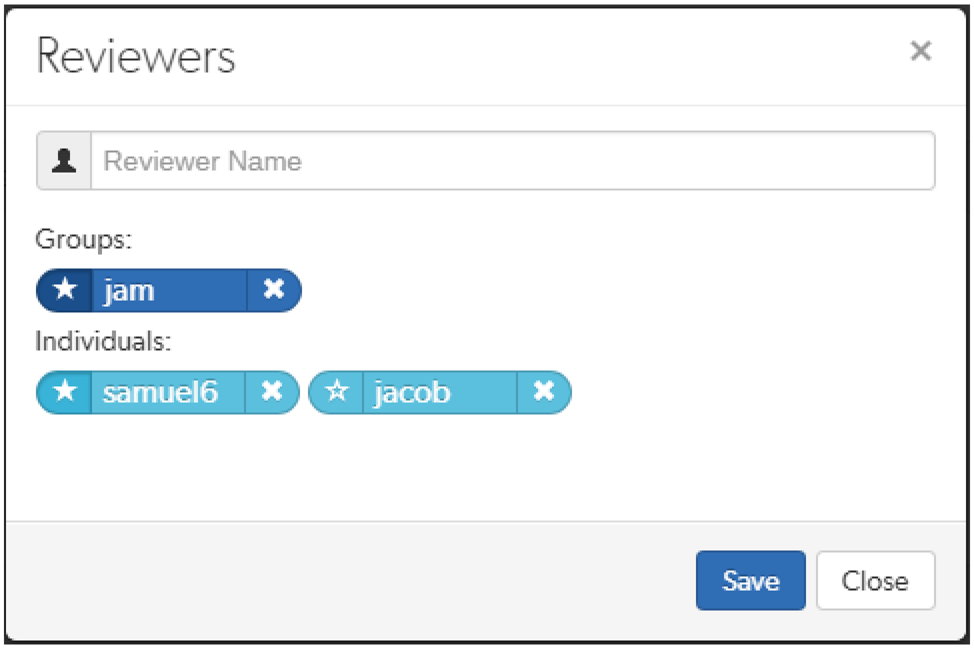
A review author, or users with *admin* or *super* privileges are always able to edit the reviewers for a review. Reviewers are always able to join or leave reviews, or to change whether their vote is required or optional.

Additionally, the following individuals may edit reviewers:

- If the review is moderated, the moderators.
- If the review is part of a project, but not moderated, all project members.
- If the review is not part of a project, any authenticated user.

To edit reviewers for a review:

1. Navigate to a review.
2. Click the edit reviewers button ✎, which appears just to the left of reviewer avatars.
   The **Reviewers** dialog is displayed.



3. Add or remove reviewers, or change the vote requirement.

   Use the reviewer search field to find users and groups by name, *userid*, *groupid*. The field auto-completes as you type, click on the user or group to add to the review.

   - **Groups:** click the star icon to the left of the *groupid* , and select whether the group is a required reviewer (one vote), a required reviewer (all votes), or an optional reviewer. A solid star means that all group member votes are required to approve a review, a solid star with a 1 inside means at least one group member must vote up and no group members vote down to approve a review, and the outlined star means that the group vote is optional.
   - **Users:** click the star icon to the left of the *userid* to toggle whether their vote is required or not. A solid star means that their vote is required to approve a review, whereas the outlined star means that their vote is optional.

   Click the **X** icon to the right of the *userid* or *groupid* to remove that reviewer from the review.
4. Click the **Save** button to save any changes made.

Responsibility

Initially, code reviews have no reviewers.

Code review authors can designate users as reviewers by including an @mention for each desired reviewer, and an @*mention for each required reviewer. They can also designate groups as reviewers by using @@mention for each desired group reviewer, an @@*mention for each required reviewer group (all members required), and an @@!mention for each required reviewer group (one vote required). Code reviewers can also designate users or groups as reviewers by using the **Edit Reviewers** dialog which allows reviewers to be added, removed, and to configure the reviewer type.

Other users can show their interest in participating in the code review by clicking their avatar in the **Reviewers** area of the code review and selecting **Join review**, or by commenting on a review or one of its files. Once a user shows such interest, they are added to the review's list of reviewers and share in the responsibility of performing the code review. Later on, reviewers can change whether their vote is required or optional, or leave a review (perhaps to prevent further notifications).

Looking at a review queue can help you determine which reviews have likely not been started, using the **No Reviewers** filter. Once a review has reviewers, it is considered to be active and appears in the review queue with the state `Has Reviewers`.

Review participation is *advisory* by default, and is used to inform your team that a code review is being conducted. The disposition of the review is reflected in the review's current state, the badges that may appear over each reviewer's avatar, and any comments reviewers might add.

Moderators

A project moderator is a user assigned to moderate reviews for a specific branch associated with a project. See how to specify moderators.

When **Only Moderators can approve or reject reviews** is set for a project branch, changing the state of any review associated with the moderated branch is restricted as follows:

- Only moderators can approve or reject the review. Moderators can also transition a review to any other state.
- The review's author, when not a moderator, can change the review's state to **Needs Review**, **Needs Revision**, **Archived**, and can attach committed changelists.

  Normally, the review's author cannot change the review's state to **Approved** or **Rejected** on moderated branches. However, authors that are also moderators have moderator privileges, and may approve or reject their own review.

  When `disable_self_approve` is enabled, authors who are moderators (or even users with admin privileges) cannot approve their own reviews.
- Project members can change the review's state to **Needs Review** or **Needs Revision**, and can attach committed changelists. Project members cannot change the review's state to **Approved**, **Rejected**, or **Archived**.
- Users that are not project members, moderators, or the review's author cannot transition the review's state.
- For the review's author and project members, if a review is not in one of their permitted states, for example if the review's state is **Rejected**, they cannot transition the review to another state.

These restrictions have no effect on who can start a review.

Required reviewers

Reviews can optionally have required reviewers. When a review has required reviewers, the review cannot be approved until all required reviewers and required reviewer groups have up-voted the review. If the review is associated with a project that has assigned moderators, even the moderators cannot approve the review without up-votes from all required reviewers (but they can reject the review).

When a group is a required reviewer, it can be set to operate in one of two ways:

- **All votes required:** all members of the group must up-vote the review to allow the review to be approved.
- **One vote required:** at least one member of the group must up-vote the review to allow the review to be approved. If any member of the group down-votes the review, the review cannot be approved.

Required reviewers are expected to take greater care while performing a review than non-required reviewers, as their votes affect whether a review can be approved or not.

To edit the reviewers for a review, and to change whether a reviewer is required or not, see "Edit reviewers" on page 226.

Note
If a review involves a branch with assigned moderators, only a moderator can approve the review, even if all required reviewers have up-voted the review.

See the description of assigning moderators.

Add yourself as a reviewer

1. Visit the review's page.
2. Log in, if you have not already done so.
3. Click your avatar in the **Reviewers** area of the review display, which should be grayed out since you are not yet a reviewer. A dropdown menu appears.
4. Select **+ Join Review**. Alternatively, you can select **^ Vote Up** or **v Vote Down** if you approve or disapprove of the review, respectively; either will cast your vote and make you a reviewer.

Your avatar is no longer grayed out, and you are now a reviewer.

Remove yourself as a reviewer

1. Visit the review's page.
2. Log in, if you have not already done so.

3. Click your avatar in the **Reviewers** area of the review display, which should not be grayed out since you are already a reviewer. A dropdown menu appears.
4. Select **X Leave Review**.

Your avatar is now grayed out, and you are no longer a reviewer.

Review workflow

There are many possible code review workflows. The section describes two typical scenarios for a code review that Swarm can handle.

Another developer reviews your code

1. You request a code review with a shelved change.
2. Another developer, Bill, sees the email notification from Swarm, clicks the review link in the email, and begins looking at the diffs in the files belonging to the review. Curious about an implementation detail, Bill clicks the line he's curious about and adds his query in a Swarm comment.
3. You receive an email notification regarding Bill's query. His query prompts you to clarify the code, say by renaming some variables and adding some better descriptive text in the surrounding code comments. You then update the review with your changes.
4. Bill sees the email notification that you have updated the review. He checks out the change, likes what he sees, and marks the review **Approved**.
5. You see the email notification that Bill has approved your review, so you commit your code.

You review another developer's code

1. Another developer, Charlie, requests a code review with a shelved change.
2. You receive an email notification from Swarm, click the review link in the email, and begin looking at the diffs in the files belong to the review. You don't like what you see, as Charlie has tried to fix a bug using a technique you have already tried previously and know to be incorrect. You add comments to the code that needs attention, flag your comments as tasks, and mark the review **Needs Revision**.
3. Charlie receives an email notification regarding your review, but disagrees with you, and adds his own comments justifying his implementation.
4. You receive an email notification regarding Charlie's comment. The technique is somewhat complicated, so rather than attempt to describe how it is incorrect, you unshelve the review's code to your own workspace, change Charlie's code, and shelve your changes. Swarm updates the review with your new code.

5. Charlie receives an email notification regarding your updates to the review. He's still unconvinced, but he unshelves your changes to try them in his local workspace. He finds that your implementation works better, but sees a couple of areas where there could be improvements. He reshelves his latest work to update the review.
6. You receive an email notification regarding Charlie's updates, check out his changes, and realize that Charlie's work is now moot because the customer has revised his plans. You add a comment to the review reporting that fact, and **Reject** the review.

States

Reviews can be in one of several states. The biggest differentiator is whether the review's files have any outstanding, uncommitted changes or not.

Whenever a review's state changes, an email notification is sent to all review participants, including the author, anyone who comments on the review or its files, anyone who has changed the review's state previously, anyone who is @mentioned, or a member of a group that is @@mentioned in the review's description or comments.

Code reviews can be in one of the following states:

- **Needs review:** The review has started and the changes need to be reviewed.
- **Needs revisions:** The changes have been reviewed and the reviewer has indicated that further revisions are required.
- **Approved:** The review has completed. The changes may need to be committed. If the changes have been committed then this review will be Approved and closed, otherwise it will be Approved and open. See the note below.
- **Rejected:** The review has completed. The changes are undesirable and should not be committed.
- **Archived:** The review has completed for now. However, it is neither rejected nor approved; it is simply put aside in case it is needed in the future.

Note
By default, when an **Approved** review is committed or updated, Swarm changes the state to **Needs Review** if the files have been modified since the review was approved. Files are considered modified if the list of involved files changes, or if the file content or file-type changes.

If one or more files in a review has the filetype +k (`ktext`), this behavior is undesirable because the files will appear to be modified when the Helix server replaces RCS keywords with their current values. See "Unapprove modified reviews" on page 303 to see how to disable this behavior.

Self-approval by review authors

By default, review authors can approve their own reviews. This behavior is based on Swarm's advisory nature.

Self-approval by authors can be prohibited on a project-by-project basis by specifying moderators for project branches (see "State change restrictions with moderation" below). However, authors who are moderators can self-approve their own reviews.

Administrators can configure Swarm to prevent all self-approval by review authors. See "Disable self-approval of reviews by authors" on page 284.

State change restrictions with moderation

Typically, any authenticated user can change the state of a review (remember that the review state is merely advisory in most cases). When the **Only Moderators can approve or reject reviews** restriction is enabled for a project branch, and one or more moderators have been assigned to that branch (see "Add a project" on page 188 for details), that branch is *moderated*. Changing the state of any review associated with a moderated branch is restricted as follows:

- Only moderators can approve or reject the review. Moderators can also transition a review to any other state.
- The review's author, when she is not a moderator, can change the review's state to **Needs Review**, **Needs Revision**, **Archived**, and can attach committed changelists.

  Normally, the review's author cannot change the review's state to **Approved** or **Rejected** on moderated branches. However, authors that are also moderators have moderator privileges, and may approve or reject their own review.

  When `disable_self_approve` is enabled, authors who are moderators (or even users with *admin* privileges) cannot approve their own reviews.
- Project members can change the review's state to **Needs Review** or **Needs Revision**, and can attach committed changelists. Project members cannot change the review's state to **Approved**, **Rejected**, or **Archived**.
- Users that are not project members, moderators, or the review's author cannot transition the review's state.
- For the review's author and project members, if a review is not in one of their permitted states, for example if the review's state is **Rejected**, they cannot transition the review to another state.

  These restrictions have no effect on who can start a review.

Required reviewers

Reviews can optionally have required reviewers. When a review has required reviewers, the review cannot be approved until all required reviewers and required reviewer groups have up-voted the review. If the review is associated with a project that has assigned moderators, even the moderators cannot approve the review without up-votes from all required reviewers (but they can reject the review).

When a group is a required reviewer, it can be set to operate in one of two ways:

- **All votes required:** all members of the group must up-vote the review to allow the review to be approved.
- **One vote required:** at least one member of the group must up-vote the review to allow the review to be approved. If any member of the group down-votes the review, the review cannot be approved.

Required reviewers are expected to take greater care while performing a review than non-required reviewers, as their votes affect whether a review can be approved or not.

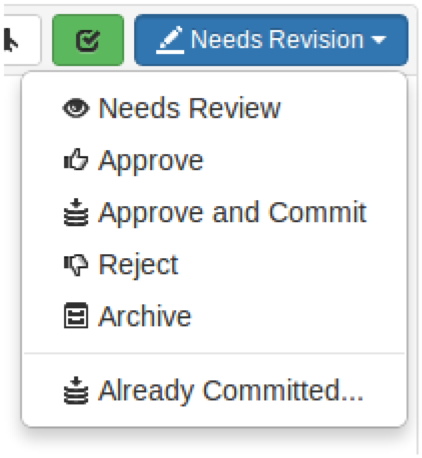
To edit the reviewers for a review, and to change whether a reviewer is required or not, see "Edit reviewers" on page 226.

Note
If a review involves a branch with assigned moderators, only a moderator can approve the review, even if all required reviewers have up-voted the review.

See the description of assigning moderators.

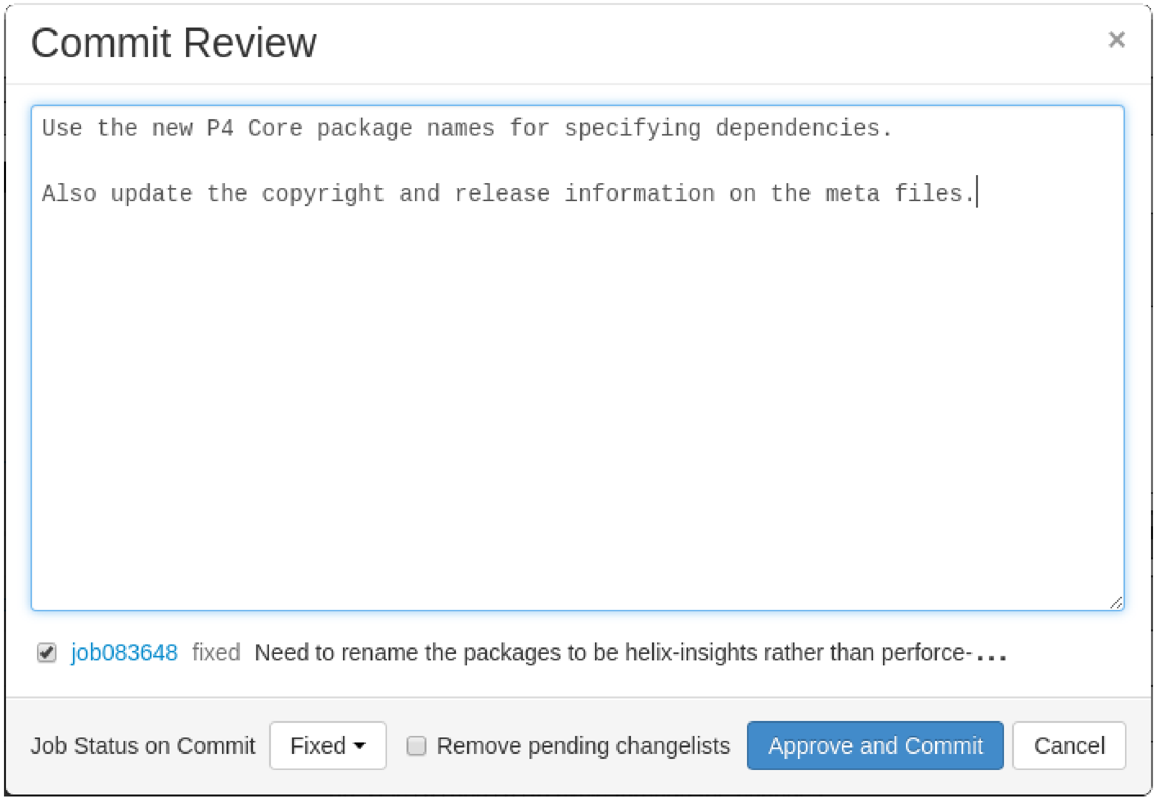
State actions

The drop-down menu provides two special actions included with the state change for uncommitted reviews:



- **Approve and Commit**

  When selected, a dialog appears allowing you to update the description, select which jobs should be associated, and specify the job status upon commit. When you click the **Approve and Commit** button, the review becomes approved and its associated files are committed. By default, Swarm's activity stream entries and email notifications note that you committed the review on behalf of the review's author. This can be configured to credit only the committer, see "Commit credit" on page 246 for details.


Commit Review

```
Use the new P4 Core package names for specifying dependencies.

Also update the copyright and release information on the meta files.
```

☑ job083648 fixed Need to rename the packages to be helix-insights rather than perforce-...

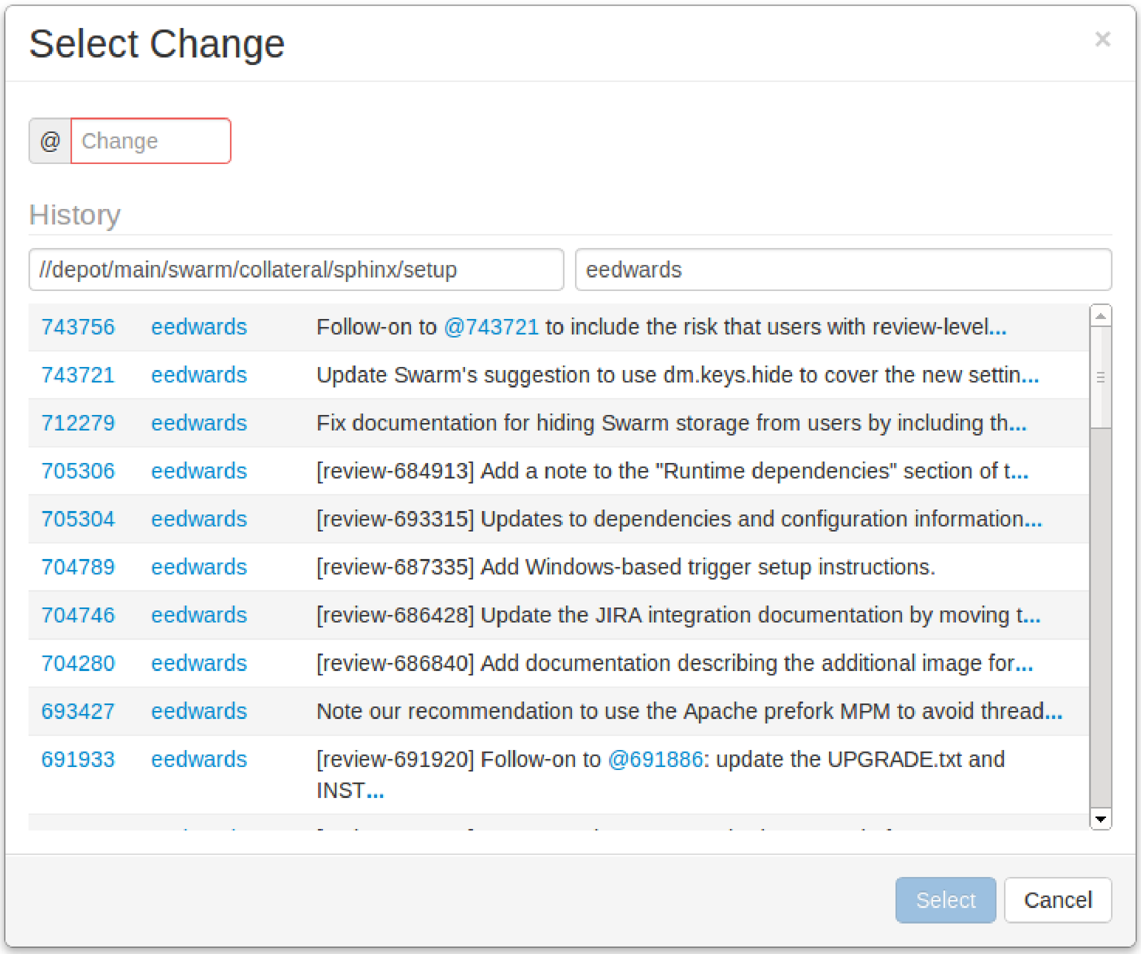
If selected, the **Remove pending changelists** button attempts to clean up automatically any changelists left behind after the review has been committed, including removing any shelved files. This option can be removed by an administrator. See "Review cleanup" on page 277 for details.

Note
The commit option can be removed by an administrator. See "Disable commit" on page 289 for details.

By default, if the committer is not the review's author, Swarm credits both users. If you prefer to credit only the committer, see "Commit credit" on page 246 for details.

- **Already Committed**

  Sometimes, a changelist that a review is based upon gets committed without the review being updated. In such a situation, selecting **Already Committed** displays a dialog presenting a list of candidate changelists:



//depot/main/swarm/collateral/sphinx/setup

eedwards

743756	eedwards	Follow-on to @743721 to include the risk that users with review-level...
743721	eedwards	Update Swarm's suggestion to use dm.keys.hide to cover the new settin...
712279	eedwards	Fix documentation for hiding Swarm storage from users by including th...
705306	eedwards	[review-684913] Add a note to the "Runtime dependencies" section of t...
705304	eedwards	[review-693315] Updates to dependencies and configuration information...
704789	eedwards	[review-687335] Add Windows-based trigger setup instructions.
704746	eedwards	[review-686428] Update the JIRA integration documentation by moving t...
704280	eedwards	[review-686840] Add documentation describing the additional image for...
693427	eedwards	Note our recommendation to use the Apache prefork MPM to avoid thread...
691933	eedwards	[review-691920] Follow-on to @691886: update the UPGRADE.txt and INST...

  Specify the changelist, if you know it, or browse the recent changes to locate the submitted change. The first field under **History** allows you to filter changes by depot path. The second field allows you to filter changes by userid. Click a changelist to select it, and then click **Select** to associate the changelist with the review and complete the review. Or, just double-click a changelist to do the same.

The review state drop-down menu for committed changes When a review has been committed, sometimes a follow-up change needs to be associated with the review. For committed reviews, the drop-down menu provides the **Add a commit** entry in place of **Already Committed**. Select **Add a Commit** to use the **Select Change** dialog as described above.

8 | Integrations

Swarm integrates with a variety of other applications and processes to provide important functionality, such as automated testing and deployment of code in reviews, issue tracking, and file previewing. This chapter describes each of the integrations available with Swarm.

Included integrations:

- JIRA
- LibreOffice
- Automated Testing for Reviews
- Automated Deployment for Reviews
- P4V

JIRA

Swarm's JIRA integration allows code reviews and committed changes to be associated with JIRA issues, making it easy to reference associated issues, and see the state of a code review or committed change within JIRA.

To associate a code review with a JIRA issue, include a JIRA issue identifier in the review's description, e.g. `SW-1234`; Swarm links to the JIRA issue and creates a link within the JIRA issue back to the code review in Swarm. Multiple JIRA issues can be included in the changelist description.

As a code review progresses, Swarm updates each associated JIRA issue with the review's current status.

Note
Swarm fetches JIRA project identifiers using a worker, once per the worker process' lifetime. See "Worker configuration" on page 307. New JIRA projects will not auto-link to JIRA until the project identifiers have been updated. By default, project identifiers are updated once every ten minutes.

By default, the JIRA module is disabled. Use the following instructions to enable the JIRA module.

Enabling the JIRA module

When enabled, the JIRA module links JIRA issues referenced in change descriptions, job descriptions, comments, and reviews to your local JIRA service. By default, the JIRA module is not enabled.

To enable the JIRA module, add the following configuration block to your *SWARM_ ROOT*`/data/config.php` file:

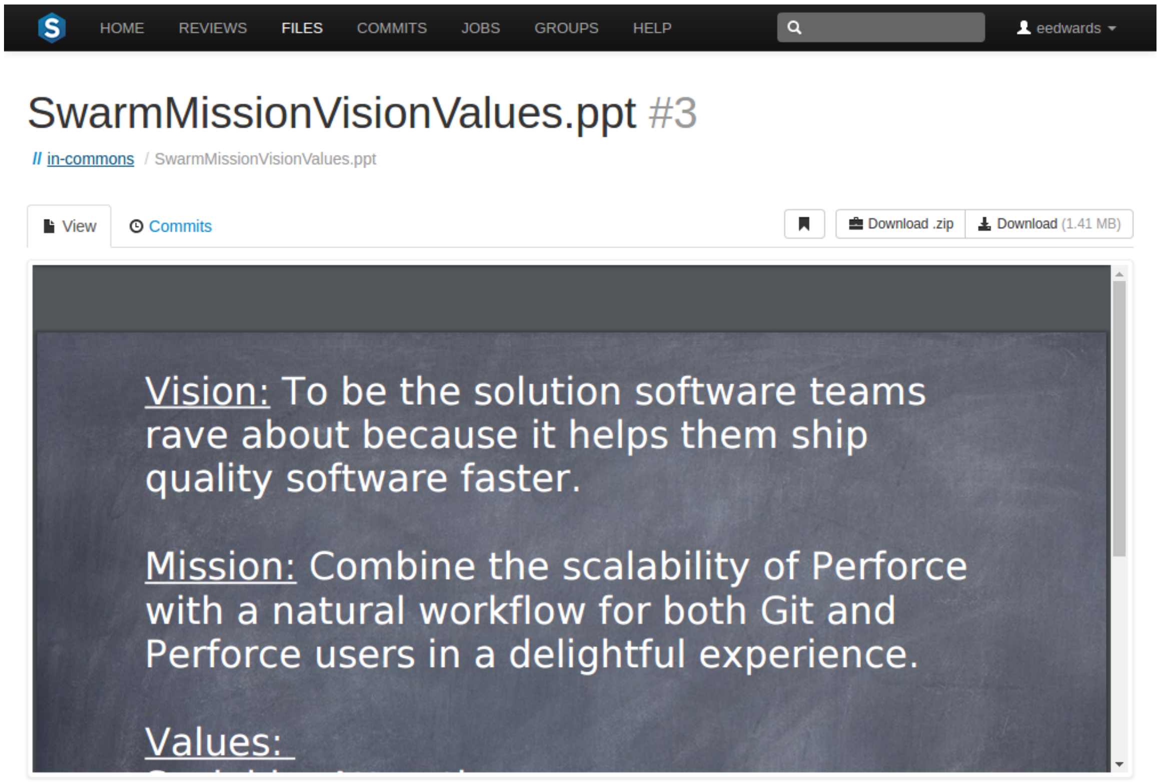
```
<?php
        // this block should be a peer of 'p4'
        'jira' => array(
                'host'      => '', // URL for your installed JIRA web interface
                'user'      => '', // the username required for JIRA API access
                'password'  => '', // the password required for JIRA API access
                'job_field' => '', // optional, if P4DTG is replicating JIRA issue IDs
                                   // to a job field, list that field here
        ),
```

Note
If your JIRA web interface uses HTTPS, you may need to configure "HTTP client options" on page 292 so that Swarm can connect successfully.

LibreOffice

LibreOffice is a free power-packed open source personal productivity suite. When LibreOffice is installed on the server hosting Swarm, Swarm automatically detects its presence and uses LibreOffice to prepare PDF previews of a variety of file types, including:

- Word documents (`.doc`, `.docx`)
- PowerPoint presentations (`.ppt`, `.pptx`)
- Excel spreadsheets (`.xls`, `.xlsx`)
- Visio diagrams (`.vsd`)
- Rich-text files (`.rtf`).



Vision: To be the solution software teams rave about because it helps them ship quality software faster.

Mission: Combine the scalability of Perforce with a natural workflow for both Git and Perforce users in a delightful experience.

Values:

Depending on your server's platform and distribution, LibreOffice may be provided as multiple packages and not all packages may be installed by default. If certain filetypes do not preview as expected in Swarm, you may need to install these optional packages to include all of the file handling capabilities of LibreOffice.

For more information on LibreOffice, see https://www.libreoffice.org.

Limitations

The LibreOffice integration has several limitations:

- Document previews in Swarm may appear different than on a desktop system if the document's fonts are not installed on the server hosting Swarm and LibreOffice. In addition, LibreOffice has some limitations rendering Microsoft Office file types, so LibreOffice-generated previews may differ from what you see using Microsoft Office.
- Large files may require a notable amount of time to preview. Very large files may exhaust the resources of the server hosting Swarm, causing the preview to fail and temporarily impacting Swarm performance.
- Swarm is currently not able to detect or show differences in LibreOffice-supported file types.
- LibreOffice cannot currently provide previews on a Mac OSX server hosting Swarm.

Installation

We recommend that you install LibreOffice from your OS distribution, via `apt-get`, `yum`, etc.

The minimal packages, and their transitive dependencies required for Swarm are:

- `libreoffice-calc`
- `libreoffice-draw`
- `libreoffice-impress`
- `libreoffice-writer`
- `libreoffice-headless` (CentOS/RHEL only)

9 | Administration

This section covers administration and configuration of Swarm.

Archives configuration

When the `zip` command-line tool is available, Swarm allows users to download a ZIP archive of a file or folder. You configure the archiving feature with the following configuration block in the *`SWARM_ROOT`*`/data/config.php` file:

```
<?php
    // this block should be a peer of 'p4'
    'archives' => array(
      'max_input_size'    => 512 * 1024 * 1024, // 512M (in bytes)
      'archive_timeout'   => 1800,              // 30 minutes
      'compression_level' => 1,                 // 0-9
      'cache_lifetime'    => 60 * 60 * 24,      // 1 day
    ),
```

The `max_input_size` key specifies the maximum file/folder content size that can be processed into a ZIP archive. The default value permits up to 512 megabytes of content to be compressed. Smaller values limit the amount of file/folder content but provide faster downloads; larger values can allow increased scanning, syncing, compressing, and downloading times.

The `archive_timeout` key specifies the amount of time, in seconds, to allow Swarm to prepare the ZIP archive for downloading. Shorter times can limit the practical size of a ZIP archive, depending on the performance of your network and the filesystem hosting Swarm; even with a generous `max_input_size` setting, if `archive_timeout` seconds have elapsed, the archive operation is terminated.

The `compression_level` key specifies the compression level to use, and must be within the range `0` to `9`. `0` means no compression, `9` means maximum compression. As this value is increased, smaller ZIP archives may result, but may require greater compression time. Swarm uses the default of `1`, which provides a reasonable tradeoff of fast compression times with light compression that can still result in an archive notably smaller than the original file/folder content.

The `cache_lifetime` key specifies the desired maximum age of cached ZIP archives. Increasing the value increases the amount of time that ZIP archives exist in the cache, which can improve the user experience for frequently downloaded files. However, ZIP archives can be quite large (depending on the size of your depot within the Helix server) and can require significant disk storage. Decreasing the value can mitigate the amount of disk space required for the cache; the tradeoff is that frequently accessed ZIP archives may need to be generated more frequently, which can have an impact on CPU and disk resources.

Avatars

Swarm uses *avatars*, images that represent users and groups responsible for events in activity streams, projects, reviews, etc.

Avatars are retrieved from an avatar provider; the default provider is gravatar.com. Swarm sends an identifier to the avatar provider (for `gravatar.com`, an MD5 hash of the user's or group's email address), and the provider returns the configured image (if one exists). If no avatar is defined with the provider or the requests fails for any reason, Swarm selects an avatar from its internal collection.

You configure the avatar lookups with the avatars configuration block in the *`SWARM_ROOT`*`/data/config.php` file. Here is an example:

```
<?php
    // this block should be a peer of 'p4'
    'avatars' => array(
        'http_url'  => 'http://www.gravatar.com/avatar/{hash}?s={size}&d=
{default}',
        'https_url' => 'https://secure.gravatar.com/avatar/{hash}?s=
{size}&d={default}',
    ),
```

Both `http_url` and `https_url` specify URLs that should be used instead of the default `gravatar.com` URLs. Swarm picks which URL to use based on the current request; for HTTPS requests, Swarm picks the `https_url` URL. If the picked URL is not defined, Swarm will use `gravatar.com`.

Several replacement values are available for inclusion in the URLs:

- `{user}`

  The current Swarm userid, Perforce groupid, or empty string

- `{email}`

  The current Swarm user's or group's email address, or empty string

- `{hash}`

  The MD5 hash of the Swarm user's or group's email address, or `00000000000000000000000000000000` if no email address is configured

- `{default}`

  The value blank for a transparent GIF (allowing users or groups without avatars to fallback to Swarm's internal avatars) or the value `mm` for a *mystery man* used in circumstances where no user or group identifier is known

- `{size}`

  the size Swarm would like in pixels for both the width and height, without units, e.g. 64

The URL you specify must include one of `{user}`, `{email}`, or `{hash}` to properly select a user-specific or group-specific avatar. The URL should include `{size}` to assist Swarm's presentation. `{default}` is not necessary, but helps provide a consistent avatar experience.

Note
By default, gravatar.com serves only G-rated avatar images. If your Swarm users and groups wish to use PG-, R-, or X-rated images, you need to configure the avatar lookup URLs with the appropriate rating flag. For example, to allow avatars with G or PG ratings, the configuration would look like:

```
<?php
    // this block should be a peer of 'p4'
    'avatars' => array(
        'http_url'  => 'http://www.gravatar.com/avatar/{hash}?r=pg&s=
{size}&d={default}',
        'https_url' => 'https://secure.gravatar.com/avatar/
{hash}?r=pg&s={size}&d={default}',
     ),
```

For more information on gravatar.com image requests, see:
https://en.gravatar.com/site/implement/images

Disable avatar lookups

If you wish to disable avatar lookups altogether and simply use Swarm's internal bee-themed avatars, set each URL to `false`. For example:

```
<?php
    // this block should be a peer of 'p4'
    'avatars' => array(
        'http_url'  => false,
        'https_url' => false,
    ),
```

Backups

Swarm stores all of the information it requires to operate within the Helix server. This includes project definitions, code reviews, comments, followers, and more. Code reviews are largely built on top of Helix Core's shelving feature, and most other records are stored in custom counters called *keys*.

Therefore, the standard recommendations for backing up your Helix server also apply when backing up your Swarm data.

For more information, see "Backup and Recovery" in *Helix Versioning Engine Administrator Guide: Fundamentals*.

In addition, your Swarm configuration and any modifications you might make to the provided modules, templates, CSS, JavaScript, etc. also need to be backed up. The *`SWARM_ROOT`*`/data` directory contains the configuration, as well as temporary working files and browser session storage.

Changelist files limit

Changelists or reviews with many files present a challenge to Swarm; it can take a notable amount of time for the file listing to reach Swarm from the Helix server, and a notable amount of time to provide the file listing HTML to a user's browser. When the file listing is many thousands of files, Swarm may run out of memory. Increasing the amount of memory for Swarm can help, but the Swarm interface may work slowly or not at all depending on the amount of memory available to the user's browser.

With the 2015.1 release, Swarm limits the number of files presented, even if a changelist or review involves more files than the limit.

You can adjust the limit to suit your needs, by specifying a value for `max_changelist_files` in the *`SWARM_ROOT`*`/data/config.php` file:

```
<?php
    'p4' => array(
         'max_changelist_files'  => 1000,
),
```

The default value of `1000` should suffice for most Swarm installations. Consider the effects of changing this value:

- Problems in the Swarm UI still exist with arbitrarily large values; there is likely no advantage to using a value over `10000`.
  Performance will be better with Helix server 2014.1 or later, as the file listing will be limited by Helix server.
- Smaller values can potentially interfere with reviews and reading changelists; the file limit may cause interesting files in a changelist or review to no longer be displayed in Swarm.

Client integration

P4V and P4VS can now integrate with Swarm. To indicate how these applications should connect with Swarm, Swarm sets the `P4.Swarm.URL` property set in Helix server. P4V and P4VS read this property, and if set, they connect to the specified URL to make Swarm API calls. If the property is unset, Swarm integration features are disabled.

When `P4.Swarm.URL` is set, P4V provides the following integration features:

- **Request a review**: requests a review for pending or committed changelists.
- **Update a review**: updates a review from the current state of a pending changelist. This works for changelists that are already associated with a review, or for unassociated changelists.
- **Open review in Swarm**: opens the review associated with the selected changelist in your system's default web browser.
- **Review Id and State columns**: adds **Review Id** and **Review State** columns to both the **Pending** and **Submitted** tabs.

By default, the first Swarm worker auto-detects the URL it is running under and sets `P4.Swarm.URL` accordingly.

For customized Swarm installations, the auto-detected URL may not use the correct hostname or port. In these scenarios, you can disable the URL auto-detection by editing the *SWARM_ROOT*`/data/config.php` file and setting the `auto_register_url` item to `false` in the p4 configuration block. For example:

```
<?php
    'p4' => array(
        'auto_register_url'  => false,
    ),
```

If you choose to disable this feature, you should manually set the `P4.Swarm.URL` property in Helix server to the URL for your Swarm installation:

```
$ p4 property -a -n P4.Swarm.URL -v https://myswarm.url:port/
```

Replace ***`https://myswarm.url:port/`*** with the URL for your Swarm installation.

Note
P4V uses an integration timeout, specified in the `P4.Swarm.Timeout` property, to limit delays in the P4V user interface. The default timeout is 10 seconds.

To change the integration timeout, run:

```
$ p4 property -a -n P4.Swarm.Timeout -v 10
```

Replace the ***`10`*** with the desired timeout in seconds. Increasing the timeout could cause notable delays in the P4V user interface, and decreasing the timeout could cause sporadic integration failures if Swarm's API responses take longer than the specified timeout.

Comment attachments

Swarm supports attaching arbitrary files to comments in code reviews and jobs.

To store files attached to comments, Swarm looks for a depot named `//.swarm`. As Swarm does not create this depot, you need to create it, or specify another depot that the Swarm *admin* user can write to.

To create a `//.swarm` depot, run the following as a user with *admin*-level privileges:

```
$ p4 depot .swarm
```

Ensure that the Swarm *admin* user can write to the `//.swarm` depot.

For more information and depot creation, see "Using multiple depots" in *Helix Versioning Engine Administrator Guide: Fundamentals*.

Specifying a depot path for comment attachments, if you prefer not to use the default `//.swarm` depot, is done with the depot_storage configuration block in the ***SWARM_ROOT*/data/config.php** file:

```
<?php
    // this block should be a peer of 'p4'
    'depot_storage' => array(
        'base_path'  => '//depot_name',
    ),
```

Replace `depot_name` with the depot where comment attachments should be stored. The Swarm *admin* needs to be able to write to this depot.

You can limit the size of comment attachments with the `attachments` configuration block in the ***SWARM_ROOT*/data/config.php** file:

```
<?php
    // this block should be a peer of 'p4'
    'attachments' => array(
        'max_file_size'  => 0, // the maximum file size to accept in bytes
),
```

Replace the *`0`* with the maximum file size in bytes that you want Swarm to accept for a comment attachment. If the file size is exceeded, users will see an error.

Note
Be aware that PHP's `upload_max_filesize` setting in *`SWARM_ROOT`*`/public/.htaccess` overrides `max_file_size` (which overrides the setting in PHP's `php.ini`). You can only use `max_file_size` to be more restrictive than the setting in *`SWARM_ROOT`*`/public/.htaccess`.

The default for `upload_max_filesize` is 8M (8 megabytes). Increase this limit if your commentors need to upload larger files.

You may also have to increase `post_max_size`. `post_max_size` should always be set larger or equal to `upload_max_filesize`, and Swarm's `max_file_size` should always be either unset, or set smaller or equal to `upload_max_filesize`, otherwise users will encounter unexpected rejection of their comment attachments.

See Handling file uploads: Common Pitfalls for more details.

Comment mentions

Since 2017.1, it is possible to use @mentions in review comments. By default, @mentions will auto-complete to all users in the system, but it is possible to fine tune this behaviour.

You configure the behaviour of @mentions in comments with the mentions configuration block in the *SWARM_ROOT*/data/config.php file. Here is an example:

```
<?php
    'mentions' => array(
        'mode' => 'global',
        'usersBlacklist' => array('super', 'swarm-admin'),
    ),
```

- **mode**

  This can be one of `global`, `projects` or `disabled`, and controls the scope of the auto-complete dropdown that is shown when the user starts typing an @mention in a comment. global will display all users in the system, `projects` will only display project members in the auto-complete list, and `disabled` means that no auto-complete drop down will be shown.

- **usersBlacklist**

  Any users listed here will never appear in the drop down list of auto-suggested users when @mentions are used.

Commit credit

When you use Swarm to commit a review, but you are not the review's author, Swarm gives credit to the review author by default. Activity stream entries and email notifications include both the committer and review author's details.

If you prefer Swarm's original behavior, which was to give credit only to the committer, you can do so by editing the *SWARM_ROOT*/data/config.php file and setting the `commit_credit_author` item to `false` in the `reviews` configuration block. For example:

```
<?php
    // this block should be a peer of 'p4'
    'reviews' => array(
        'commit_credit_author'  => false,
    ),
```

Commit-edge deployment

Swarm can connect to a Helix server configured to use the *commit-edge architecture*, which is a specific replication configuration that employs a *commit* server and one or more *edge* servers. This configuration distributes the compute, storage, and network requirements for improved performance and geographic distribution.

When Swarm is connected to a commit server, the first worker detects this situation and sets a key in the Helix server, `P4.Swarm.CommitURL`, to an auto-detected URL. This allows any other Swarm instances that may be connected to edge servers to share reviews amongst all edge servers.

For more information on Helix server's commit-edge architecture, see the "Commit-edge Architecture chapter" in the *Helix Versioning Engine Administrator Guide: Multi-Site Deployment*.

P4V Authentication

When using P4V's Swarm integration in a commit-edge deployment, users may encounter authentication errors; such errors can result from incorrect configuration of login tickets in distributed environments. Essentially, the problem is that while P4V is connected to an edge server, Swarm is connected to the commit server, and the login tickets do not match.

If P4V users see the `error Host requires authentication`, the solution we recommend is to forward login requests to the commit server. This can be achieved by executing the following two commands as a user with *operator* or *super* privileges in the Helix server:

```
$p4 configure set auth.id=authid
$p4 configure set rpl.forward.login=1
```

Replace ***`authid`*** with the authentication identifier for your Helix server.

For more information, see our Knowledge Base article Single Ticket Login in Distributed Environments, and the `p4 serverid` command in the *P4 Command Reference*.

Commit timeout

When a code review contains many files, or large files, or both, committing the review within Swarm can take some time. The default configuration, within the *`SWARM_ROOT`*`/data/config.php` file:

```
<?php
    // this block should be a peer of 'p4'
    'reviews' => array(
        'commit_timeout'  => 1800, // 30 minutes
    ),
```

The `commit_timeout` key is expressed in seconds. If a commit operation takes longer than this limit, it is terminated. It is likely that a terminated commit requires administrator intervention to complete the commit using another client.

Configuration overview

This section provides an overview of all the possible configuration blocks in the *SWARM_ ROOT*/data/config.php file. Click on any underlined item to see a detailed description.

Warning
While the syntax of this example is correct, it includes configuration values that **cannot work**. Ensure that you adjust the configuration appropriately for your Swarm installation before using this example in testing or production.

```
<?php
    return array(
        'activity' => array(
            'ignored_users' => array(
                'git-fusion-user',
                'p4dtguser',
                'system',
            ),
        ),
        'archives' => array(
            'max_input_size'    => 512 * 1024 * 1024, // 512M (in bytes)
            'archive_timeout'   => 1800,              // 30 minutes
            'compression_level' => 1,                 // 0-9
            'cache_lifetime'    => 60 * 60 * 24,      // 1 day
        ),
        'attachments' => array(
            'max_file_size'  => 0, // the maximum file size to accept in
bytes
        ),
        'avatars' => array(
            'http_url'  => 'http://www.gravatar.com/avatar/{hash}?s=
{size}&d={default}',
            'https_url' => 'https://secure.gravatar.com/avatar/{hash}?s=
{size}&d={default}',
        ),
        'depot_storage' => array(
            'base_path'  => '//depot_name',
        ),
        'diffs' => array(
            'ignore_whitespace_default'  => false,
            'max_diffs'                  => 1500,
        ),
        'environment' => array(
            'mode'         => 'production',
            'hostname'     => 'myswarm.hostname',
            'external_url' => null,
            'base_url'     => null,
        ),
        'files' => array(
            'max_size'         => 1048576,
            'download_timeout' => 1800,
        ),
        'groups' => array(
            'super_only'  => true,
        ),
        'http_client_options'   => array(
            'timeout'   => 5,
            'hosts'     => array(),
        ),
        'jira' => array(
            'host'      => '',
            'user'      => '',
            'password'  => '',
            'job_field' => '',
        ),
        'log' => array(
            'priority'  => 5,
        ),
        'mail' => array(
            // 'recipients' => array('user@my.domain'),
            'notify_self'   => false,
            'transport' => array(
            'host' => 'my.mx.host',
            ),
        ),
        'mentions' => array(
            'mode'  => 'global',
            'usersBlacklist'  => array('super', 'swarm-admin'),
        ),
        'notifications' => array(
            'honor_p4_reviews'      => false,
            'opt_in_review_path'    => '//depot/swarm',
            'disable_change_emails' => false,
        ),
        'p4' => array(
            'port'      => 'my-helix-versioning-engine:1666',
            'user'      => 'admin_userid',
            'password'  => 'admin user ticket or password',
            'slow_command_logging' => array(
                3,
                10 => array('print', 'shelve', 'submit', 'sync',
'unshelve'),
            ),
            'max_changelist_files' => 1000,
            'auto_register_url'    => true,
        ),
        'projects' => array(
            'mainlines' => array(
                'main', 'mainline', 'master', 'trunk',
            ),
            'add_admin_only'            => false,
            'add_groups_only'           => array(),
            'edit_name_admin_only'      => false,
            'edit_branches_admin_only' => false,
            'readme_mode'              => 'restricted',
            'sidebar_sort_field'       => 'name',
        ),
        'queue'  => array(
            'workers'             => 3,    // defaults to 3
            'worker_lifetime'     => 595,  // defaults to 10 minutes (less
5 seconds)
            'worker_task_timeout' => 1800, // defaults to 30 minutes
            'worker_memory_limit' => '1G', // defaults to 1 gigabyte
        ),
        'reviews' => array(
            'patterns' => array(
                // #review or #review-1234 with surrounding whitespace/eol
                'octothorpe'      => array(
                    'regex'  => '/(?P<pre>\s+|^)'
                              . '\#(?P<keyword>review)(?:-(?P<id>[0-
9]+))?'
                              . '(?P<post>\s+|$)/i',
                    'spec'   => '%pre%#%keyword%-%id%%post%',
                    'insert' => "%description%\n\n#review-%id%",
                    'strip'  => '/^\s*\#review(-[0-9]+)?(\s+|$)'
                              . '|(\s+|^)\#review(-[0-9]+)?\s*$/i',
                ),

                // [review] or [review-1234] at start
                'leading-square'   => array(
                    'regex' => '/^(?P<pre>\s*)'
                             . '\[(?P<keyword>review)(?:-(?P<id>[0-
9]+))?\]'
                             . '(?P<post>\s*)/i',
                    'spec'  => '%pre%[%keyword%-%id%]%post%',
                ),

                // [review] or [review-1234] at end
                'trailing-square' => array(
                    'regex' => '/(?P<pre>\s*)'
                             . '\[(?P<keyword>review)(?:-(?P<id>[0-
9]+))?\]'
                             . '(?P<post>\s*)?$/i',
                    'spec'  => '%pre%[%keyword%-%id%]%post%',
            ),
            'filters' =>array(
                'result_sorting' => true,
                'date_field' => 'updated', // 'created' displays and sorts by
created date, 'updated' displays and sorts by last updated
                ),
            ),
            'commit_timeout'        => 1800, // 30 minutes
            'disable_commit'        => true,
            'disable_self_approve' => false,
            'commit_credit_author' => true,
            'ignored_users'         => array(),
            'unapprove_modified'    => true,
            'allow_author_change'   => true,
            'sync_descriptions'     => true,
            'expand_group_reviewers' => false
        ),
        'search' => array(
            'maxlocktime'     => 5000, // 5 seconds, in milliseconds
            'p4_search_host'  => '',   // optional URL to Helix Search
Tool
        ),
        'security'  => array(
            'disable_system_info'      => false,
            'email_restricted_changes' => false,
            'emulate_ip_protections'   => true,
            'https_port'               => null,
            'https_strict'             => false,
            'https_strict_redirect'    => true,
            'prevent_login'            => array(),
            'require_login'            => true,
        ),
        'session'  => array(
            'cookie_lifetime'              => 0, // 0=expire when browser
closed
            'gc_maxlifetime'               => 60*60*24*30, // 30 days
            'remembered_cookie_lifetime'   => 60*60*24*30, // 30 days
        ),
        'short_links' => array(
            'hostname'     => 'myho.st',
            'external_url' => 'https://myho.st:port/sub-folder',
        ),
        'upgrade' => array(
            'status_refresh_interval' => 10,    //Refresh page every 10
seconds
            'batch_size' => 1000,       //Fetch 1000 reviews to lower memory
usage
        ),
        'xhprof' => array(
            'slow_time'      => 3,
            'ignored_routes' => array('download', 'imagick',
'libreoffice', 'worker'),
        ),
    );
```

Note
The Swarm configuration file does not include PHP's standard closing tag (?>). This is intentional as it prevents unintentional whitespace from being introduced into Swarm's output, which would interfere with Swarm's ability to control HTTP headers. Debugging problems that result from unintentional whitespace can be challenging, since the resulting behavior and error messages often appear to be misleading.

Diff configuration

Swarm's Diffs feature can be customized via the following config items.

ignore_whitespace_default

Normally, Swarm diffs do not ignore whitespace by default. The user can toggle **ignore whitespace** off and on. However, whenever a review is reloaded in your browser (or you visit another review), the default is in effect.

If you and your users prefer that ignore whitespace is enabled by default, set the `ignore_whitespace_default` config item to `true` in the *SWARM_ROOT*`/data/config.php` file. For example:

```
<?php
    // this block should be a peer of 'p4'
    'diffs' => array(
        'ignore_whitespace_default' => true,
    ),
```

max_diffs

If the number of diffs is greater than the configured maximum, then the list of diffs is truncated by default. An option will be given to show all the diffs for that file.

```
    'diffs' => array(
        'max_diffs' => 1500,
    ),
```

Email configuration

Swarm's email delivery is controlled by the mail configuration block in the *SWARM_ROOT*/**data/config.php** file. Here is an example:

```
<?php
    // this block should be a peer of 'p4'
    'mail' => array(
        // 'sender' => 'swarm@my.domain',   // defaults to
'notifications@hostname'
        'transport' => array(
            'name' => 'localhost',          // name of SMTP host
            'host' => '127.0.0.1',          // host/IP of SMTP host
            'port' => 587,                  // SMTP host listening port
            'connection_class'  => 'plain', // 'smtp', 'plain', 'login',
'crammd5'
            'connection_config' => array(   // include when auth required
to send
                'username'  => 'user',      // user on SMTP host
                'password'  => 'pass',      // password for user on SMTP
host
                'ssl'       => 'tls',       // empty, 'tls', or 'ssl'
            ),

            // override email deliveries and store all messages in this
path
            // 'path' => '/var/spool/swarm',
        ),

        // override regular recipients; send email only to these addresses
        // 'recipients' => array(
        //     'user1@my.domain',
        //     'user2@my.domain',
        // ),

        // send notifications of comments to comment authors?
        'notify_self' => false,

        // blind carbon-copy recipients
        // 'use_bcc' => true,

        // suppress reply-to header
        // 'use_replyto' => false,
    ),
```

Note
Without any mail configuration in the *`SWARM_ROOT`*`/data/config.php` file, Swarm attempts to send email according to PHP's configuration, found in the `php.ini` file. By default, the configuration in `php.ini` relies on SendMail being installed.

Important
Email delivery for events related to restricted changes is disabled by default. See "Restricted Changes" on page 289 for details on how to enable restricted change notifications.

Sender

The sender item within the `mail` block specifies the sender email address that should be used for all notification email messages. The default value is:

```
notifications@hostname
```

`hostname` is the name of the host running Swarm, or when specified with the "Environment" on page 259 configuration.

Transport

The `transport` block within the `mail` block defines which mail server Swarm should use to send email notifications. Most of the items in this block can be omitted, or included as needed. See the Zend Framework's Mail Transport Configuration Options for a description of most fields and their default values.

Swarm uses the custom path item to direct all email messages to a directory instead of attempting delivery via SMTP. For details, see "Save all messages to disk" on the next page.

Recipients

The `recipients` item within the `mail` block allows you to specify a list of recipients that should receive email notifications, overriding the normal recipients. This is useful if you need to debug mail deliveries.

```
<?php
    // this block should be a peer of 'p4'
    'mail' => array(
        'recipients' => array(
            'user1@my.domain',
            'user2@my.domain',
        ),
    ),
```

Any number of recipients can be defined. If the array is empty, email notifications are delivered to the original recipients.

notify_self

The `notify_self` item within the `mail` block specifies whether comment authors should receive an email for their comments. The default value is `false`. When set to `true`, comment authors receive an email notification for their own comments.

Use BCC

The `use_bcc` item within the `mail` block allows you to address recipients using the BCC email field. Setting the value to `true` causes Swarm to use the `Bcc:` field in notifications instead of the `To:` field, concealing the email addresses of all recipients.

```
<?php
    // this block should be a peer of 'p4'
    'mail' => array(
        'use_bcc' => true,
    ),
```

Use Reply-To

The `use_replyto` item within the `mail` block allows you to suppress populating the Reply-To email field. Setting the value to `false` causes Swarm to omit the `Reply-To:` field in notifications; by default, it is populated with the author's name and email address. When this field is `true`, users receiving an email notification can simply reply to the email and their response will be addressed to the author.

```
<?php
    // this block should be a peer of 'p4'
    'mail' => array(
        'use_replyto' => false,
),
```

Save all messages to disk

For testing purposes, you may want to send all email to disk without attempting to send it to recipients. Use the following configuration block to accomplish this:

```
<?php
    // this block should be a peer of 'p4'
    'mail' => array(
        'transport'  => array('path' => MAIL_PATH),
    ),
```

`<MAIL_PATH>` should be replaced with the absolute path where email messages should be written. This path must already exist and be writable by the web server user.

Note
Use of the path item causes Swarm to ignore **all** other configuration within the transport block. This is why path is commented out in the main example.

Email headers

Swarm sends out notification emails that contain custom headers which email programs can use for automatic filtering. Emails also contain headers to ensure they are correctly threaded in email clients which support doing so.

All Swarm emails contain the following headers, which can be used to identify which Swarm server they came from:

```
X-Swarm-Host: swarm.perforce.com
X-Swarm-Version: SWARM/2017.1/1500036 (2017/03/27)
```

The exact values may differ according to the version of Swarm you are running, and its configuration.

If a notification is applicable to one or more projects, then each project will be listed in the `X-Swarm-Project` header, which contains a list of one or more project names. Reviews may span multiple projects, so in this case a single email is sent out with each project listed.

```
X-Swarm-Project: gemini, apollo
X-Swarm-Host: swarm.perforce.com
X-Swarm-Version: SWARM/2017.1/1500036 (2017/03/27)
```

If one or more of the applicable projects is private, then two or more emails may be sent. In order for the existance of the private project to be hidden from non-members, any email sent to them will not contain references to the private project. Members of each private project will receive an email tailored to them which contains references to that private project. The email to the non-private projects will not contain references to any of the private projects in the `X-Swarm-Project` header.

For example, if a review spans three projects, called `Gemini`, `Apollo`, and `Ultra`, where `Ultra` is a private project, then members of projects `Gemini` and `Apollo` will receive an email with the following headers:

```
X-Swarm-Project: gemini, apollo
X-Swarm-Host: swarm.perforce.com
X-Swarm-Version: SWARM/2017.1/1500036 (2017/03/27)
```

Members of the `Ultra` project will receive an email with the following header:

```
X-Swarm-Project: ultra
X-Swarm-Host: swarm.perforce.com
X-Swarm-Version: SWARM/2017.1/1500036 (2017/03/27)
```

This can result in users receiving two notification emails (if they are members of `Ultra` and one of the other two projects), but privacy for the private project is preserved.

Emoji

By default, Swarm uses a font to provide Emoji images. Swarm can make use of Emoji images from the Gemoji project. Gemoji provides support for more Emojis and works on more browsers and platforms than the font Swarm normally uses.

Due to licensing issues, Gemoji cannot be distributed with Swarm. You can use the Gemoji images after following these steps:

1. Download the latest release (currently 3.0.0) of the Gemoji project from their releases page.
2. Unpack the release archive into *SWARM_ROOT*`/public/vendor`.

   After unpacking, you should see a new folder: ***SWARM_ROOT*/public/vendor/gemoji-*3.0.0***

   3.0.0 represents the version of Gemoji, which may differ if you downloaded a different or newer release.
3. Rename the new folder:

   ```
   $ cd SWARM_ROOT/public/vendor
   $ mv gemoji-3.0.0 gemoji
   ```

   Replace `3.0.0` in the above command if you have downloaded a different or newer release of Gemoji.
4. Ensure that the new images are readable.

   ```
   $ cd SWARM_ROOT/public/vendor
   $ chmod -R +r gemoji
   ```

Swarm detects and uses Gemoji images automatically.

Environment

This section describes the *environment* configuration items available for Swarm:

- **mode**: whether Swarm operates in *production* or *development* mode.
- **hostname**: specifies the canonical hostname Swarm should use, such as in links to Swarm in email notifications.
- **external_url**: specifies the canonical URL Swarm should use, such as in links to Swarm in email notifications. Swarm can often auto-detect the correct URL, but use of `external_url` might be necessary in complex web hosting environments.
- **base_url**: specifies the folder name Swarm is installed within when Swarm is not installed in the web server's document root.

Add the following configuration block to the *SWARM_ROOT*`/data/config.php` file:

```
<?php
    // this block should be a peer of 'p4'
    'environment' => array(
        'mode'         => 'development',      // defaults to 'production'
        'hostname'     => 'myswarm.hostname', // defaults to requested
hostname
        'external_url' => null,               // defaults to null
```

```
        'base_url'     => null,               // defaults to null
    ),
```

mode

By default, Swarm operates in *production* mode. When `mode` is set to `development`, Swarm displays greater error detail in the browser. Also, Swarm switches from including aggregated and minified JavaScript and CSS to requesting each JavaScript and CSS resource for all active modules. The default value is production. Any value other than `development` is assumed to mean production.

`development` mode makes it easier to discover problems and to identify their source, but also incurs additional browser overhead due to many more JavaScript and CSS requests for larger files. We recommend that you do not use development mode in production environments, unless directed to do so by Perforce technical support.

hostname

The hostname item allows you to specify Swarm's hostname. This could be useful if you have multiple virtual hosts deployed for a single Swarm instance; Swarm uses the hostname you configure when generating its web pages and email notifications.

Note
The value specified for the hostname item should be just the hostname. It should not include a scheme (e.g. `"http://"`), nor should it include a port (e.g. `":80"`).

external_url

The `external_url` item allows you to specify Swarm's canonical URL. This is useful if your Swarm instance is proxied behind another web service, such as a load balancer, caching proxy, etc., because Swarm's auto-detection of the current hostname or port could otherwise result in incorrect self-referencing URLs.

When specified, Swarm uses the `external_url` item as the prefix for any URLs it creates that link to itself in its web pages and email notifications.

Note
Any path components included in external_url are ignored. If you specify `https://myswarm.url:8080/a/b/c`, Swarm only uses `https://myswarm.url:8080/` when composing URLs.

Important
If you specify `base_url` along with `external_url` and you have deployed multiple Swarm

instances that connect to the same Helix server, ensure that all Swarm instances specify the same **`base_url`**. Varying **`base_url`** amongst cooperating Swarm instances is not supported.

base_url

The **`base_url`** item allows you to specify Swarm's folder within the web server's document root. This is useful if you cannot configure Swarm to operate within its own virtual host, such as when you have an existing web service and Swarm must exist alongside other applications or content.

By default, **`base_url`** is null, which is equivalent to specifying **`/`**. If you specify a folder, include the leading **`/`**. For example, **`/swarm`**.

Important
If you specify `external_url` along with **`base_url`** and you have deployed multiple Swarm instances that connect to the same Helix server, ensure that all Swarm instances specify the same **`base_url`**. Varying **`base_url`** amongst cooperating Swarm instances is not supported.

Excluding Users from Activity Streams

Larger Helix server installations often have one or more *service* users that perform automated tasks, such as build systems, continuous integration test servers, integrations with 3rd-party databases via P4DTG, or with Git via Perforce Git Fusion.

As Swarm reports the activity of users, and these service users can generate significant volumes of activity entries, Swarm provides a mechanism to ignore activity from specified users.

Update the *SWARM_ROOT*/**`data/config.php`** file to include the following configuration block:

```
<?php
    // this block should be a peer of 'p4'
    'activity' => array(
        'ignored_users' => array(
            'git-fusion-user',
            'p4dtguser',
            'system',
        ),
    ),
```

After ***SWARM_ROOT*/`data/config.php`** is updated, Swarm no longer records activity for any of the listed userids. Any previously recorded activity is included in activity streams.

Files configuration

Swarm can be customized using the following config items:

max_size

Swarm limits the size of files that are displayed in a review or standard file view. This defaults to 1MB, but can be configured to be larger or smaller. Files larger than this will be truncated.

If the value is set to zero, then the file size is unlimited.

```
<?php
    // this block should be a peer of 'p4'
    'files' => array(
        'max_size' => 1 * 1024 * 1024, // 1MB (in bytes)
    ),
```

download_timeout

When downloading files, there is a timeout which defaults to 30 minutes. This can be changed by setting the `download_timeout` configurable to a value in seconds. Alternatively, setting it to zero removes the timeout.

```
<?php
    // this block should be a peer of 'p4'
    'files' => array(
      'download_timeout' => 1800, // seconds (30 minutes)
    ),
```

Groups configuration

By default, Swarm allows all users to see the list of user groups under the 'Groups' tab in the top bar. Users can view the members of a group, as well as their activities.

You configure the visibility of this tab with the groups configuration block in the *SWARM_ROOT*`/data/config.php` file, as in the following example:

```
<?php
    // this block should be a peer of 'p4'
    'groups' => array(
        'super_only' => true,
    ),
```

Setting the `super_only` option to `true` hides the groups tab from non-admin users.

Ignored users for reviews

Automated test environments may inadvertently participate in code reviews if they copy user-generated change descriptions. For example, if an automated system copied a change description containing #review and subsequently shelved or committed files, a new review would be started. Similarly copying a description with `#review-123` could inadvertently update an existing review. As test environments may involve thousands or millions of tests, such interactions can potentially generate far too many Swarm notifications.

To mitigate this problem, Swarm can be configured to ignore specified users for the purposes of starting or updating a review. Edit the *SWARM_ROOT*`/data/config.php` file, and provide the list of users to ignore in the `ignored_users` item in the reviews configuration block. For example:

```
<?php
    // this block should be a peer of 'p4'
    'reviews' => array(
        'ignored_users'  => array('build_user1', 'build_user2'),
    ),
```

License

Helix Swarm has always been free to use with an unlicensed Helix server. With the 2014.4 release, Swarm is free to use with any Helix server. You no longer need to purchase a Swarm-specific license, and any existing Swarm-specific licenses are no longer evaluated.

An unlicensed Helix server provides unlimited use for up to 5 users and 20 workspaces. When the user or workspace limit is crossed, Helix server imposes a maximum of 1,000 files. Swarm honors the restrictions of Helix server.

Locale

Swarm is fully localized; with an appropriate language pack installation, Swarm can support users in multiple languages.

A language pack consists of `gettext-style` `default.po` and `default.mo` files, placed in a folder named for the locale they represent, within the language folder in the Swarm root directory. In addition, language packs contain two javascript files to provide translation strings for the in-browser UI, `locale.js` and `locale.jsgz`, which both appear in the *SWARM_ROOT*`/public/build/language` folder.

The following example illustrates the directory layout of a language pack:

```
SWARM_ROOT/
    language/
        locale/
```

```
            default.mo
            default.po
    public/
        build/
            language/
                locale.js
                locale.jsgz
```

You can configure certain localization behaviors with the `translator` configuration block in the *SWARM_ROOT*`/data/config.php` file. Here is an example:

```
<?php
    // this block should be a peer of 'p4'
    |'translator' => array(
        'detect_locale'               => true,
        'locale'                      => "",
        'translation_file_patterns'   => array(),
    ),
```

The `detect_locale` key determines whether Swarm attempts to detect the browser's locale. The default value is `true`. Set the value to `false` to disable browser locale detection.

The `locale` key is a string specifying the default locale for Swarm. Alternately, an array of 2 strings can be used to specify the default locale, and a fallback locale. For example:

```
<?php
    // this block should be a peer of 'p4'
    'translator' => array(
        'locale'                      => array("en_GB", "en_US"),
    ),
```

The `translation_file_patterns` key allows you to customize Zend's translation infrastructure, which you might do if you are developing your own language pack. For details, see Zend\I18n.

Logging

Helix Swarm is a web application, so there are several types of logging involved during the course of Swarm's normal operations.

Web server logging

All accesses to Swarm may be logged by the web server hosting Swarm. As web server log configuration is web server specific, refer to your web server's documentation. Since we recommend the use of Apache, more information regarding log configuration in Apache can be found here: https://httpd.apache.org/docs/2.2/logs.html

Helix Versioning Engine logs

Swarm communicates with the associated Helix server for every page request. Review the Swarm-generated requests on your Helix server by enabling logging.

For more information, see "Configuring logging" and "Auditing user file access" in the *Helix Versioning Engine Administrator Guide: Fundamentals*.

Swarm logs

Depending on the log configuration you provide to Swarm, Swarm can log its own operations. Swarm's logs are much more helpful if you encounter problems.

Swarm stores its log data in the ***SWARM_ROOT*/data/log** file. The volume of entries recorded in the log depends on the configuration stored in the ***SWARM_ROOT*/data/config.php** file. Here is an example:

```
<?php
    // this block should be a peer of 'p4'
    'log' => array(
        'priority'  => 3, // 7 for max, defaults to 3
),
```

The log `priority` specifies the importance of the messages to be logged. When `priority` is set to `0` (the lowest value), only the most important messages are logged. When `priority` is set to `7` (the highest value), all messages, including the least important messages, are logged. The default priority is `3`.

The different priority levels are:

Priority	Description
0	Emergency: a message about a system instability
1	Alert: a message about a required immediate action
2	Critical: a message about a critical condition
3	Error: a message about an error
4	Warning: a message about a warning condition

Priority	Description
5	Notice: a message about a normal but significant condition
6	Info: an informational message
7	Debug: a debugging message

Note
Setting `priority` to a value higher than **7** does not result in increased logging. Setting `priority` to a value lower than **0** does not result in reduced logging.

Trigger Token Errors

If the trigger tokens are missing or invalid, the web server error log contains a suitable error:

```
queue/add attempted with invalid/missing token: token used
```

A token is *invalid* when it is not formed from the characters A through Z (in upper or lowercase), 0 through 9, or a dash (-).

A token is *missing* when a file using the token as its name does not exist in the *`SWARM_ROOT`*`/data/queue/tokens` directory.

Performance logging

Swarm logs warnings whenever commands issued to the Helix server take longer than expected to complete. These warnings can be used to diagnose performance problems in the Helix server.

Note
By default, warnings are not captured in the Swarm log. To capture warnings, the log priority must be set to 4 or higher.

The default configuration is:

```
<?php
    // this block should be placed within the 'p4' block
    'slow_command_logging' => array(
        3, // commands without a specific rule have a 3-second limit
        10 => array('print', 'shelve', 'submit', 'sync', 'unshelve'),
    ),
```

In this configuration block, the numeric key specifies the time threshold in seconds, and the value (if present) is an array of Helix server commands that should use the threshold. Should a command be associated with multiple thresholds, the largest is used for that command.

In addition, Swarm automatically detects when the PHP extension **xhprof** is installed and collects profiling data for requests that take longer than expected. The profiling data could be helpful in diagnosing performance issues within Swarm itself, particularly when analyzed in combination with the slow command logging described above. When collected, profiling data is stored in the *SWARM_ROOT*/data/xhprof folder.

The default configuration is:

```
<?php
    // this block should be a peer of 'p4'
    'xhprof' => array(
        'slow_time'       => 3, // the threshold in seconds
        'ignored_routes' => array('download', 'imagick', 'libreoffice',
'worker'),
    ),
```

`slow_time` specifies the threshold, in seconds, that should be used to determine when a Swarm request is slow. `ignored_routes` is an array that specifies a list of Zend Framework `route` names that should not be profiled. For example, Swarm's `Files` module specifies that the download route should be ignored from profiling as downloads could take significantly longer than the default threshold.

Note
Depending on the performance of the server hosting Swarm, and particularly the performance of the associated Helix server, you may want to monitor the *SWARM_ROOT*/data/xhprof folder for disk usage. Each request that exceeds the time threshold causes 200-600KB of data to be written.

Mainline branch identification

When viewing a project's files, the initial view is the list of the project's branches. The branches appear in alphabetical order, but the branch identified as the main branch appears first and bolded.

Swarm uses a list of names to identify which of a project branches should be considered as the main branch. The default names are *main*, *mainline*, *master*, and *trunk*.

You can adjust the configuration to match your local branch naming convention. Here is an example of the configuration block:

```
<?php
    // this block should be a peer of 'p4'
    'projects' => array(
        'mainlines' => array(
            'main', 'mainline', 'master', 'trunk',
        ),
    ),
```

Notifications

Swarm can be configured to provide generic notifications of committed changes in Helix server, taking the role of a review daemon.

Notifications configuration is expressed with a **`notifications`** block in the ***`SWARM_ROOT`*****`/data/config.php`** file, similar to the following example:

```
<?php
    // this block should be a peer of 'p4'
    'notifications' => array(
        'honor_p4_reviews'      => false,                   // defaults to
false
        'opt_in_review_path'    => '//depot/swarmReviews', // required if
honor_p4_reviews is true; defaults to ''
        'disable_change_emails' => false,                   // optional;
defaults to false
    ),
```

If **`honor_p4_reviews`** is set to **`true`**, then **`opt_in_review_path`** must be set to a path somewhere in the depot. This path does not need to point to an actual file that exists, but it must be accessible by all users who want to make use of this functionality. For example:

```
'notifications' => array(
        'honor_p4_reviews'   => true,
        'opt_in_review_path' => '//depot/swarmReviews',
),
```

If these two values are set, then users can make use of the Perforce review functionality by subscribing to the **`opt_in_review_path`** in their user spec. Any user subscribed to that file, will receive notifications for all the other paths they are subscribed to.

We recommend that **`opt_in_review_path`** point to a file that does not exist. Ideally, it points to a file that no user is likely to want to create. It must however be in a valid depot.

For example, if a user has the following review paths set in their user spec:

```
$ p4 user -o asmith
User:   asmith

Email:  asmith@example.com

FullName:       Alice Smith

Reviews:
        //depot/swarmReviews
        //depot/main/acme/...
        //depot/main/orion/...
        //depot/dev/asmith/...
```

The `//depot/swarmReviews` means that this user is subscribed to the path set in `opt_in_review_path`, and therefore will receive notifications. The rest of the subscription lines define which paths in the depot this user is interested in. Therefore this user will receive a notification for a change made to `//depot/main/acme/foo.txt`, but not a change made to `//depot/dev/acme/foo.txt`.

For example, to see which users are subscribed to receive notifications you can run `p4 reviews <path>` against the `opt_in_review_path` value:

```
$ p4 reviews //depot/swarmReviews
asmith <asmith@example.com> (Alice Smith)
bbrown <bbrown@example.com> (Bob Brown)
erogers <erogers@example.com> (Eve Rogers)
```

To see which users are subscribed to files in a particular changelist, you can run `p4 reviews -c <changelist>`. Swarm will notify the users who subscribe to both the review of this changelist and the review path, `opt_in_review_path`.

- **honor_p4_reviews**: When set to `true`, Swarm sends notification emails for every committed change to all users where the change matches one of their `Reviews:` paths.
- **opt_in_review_path**: Optional item, required only if `honor_p4_reviews` is set. This item specifies a special depot path, which typically would not exist in the Helix server machine. When a path is specified, users must include this path (or a path that contains it) in the `Reviews:` field of their user spec to cause Swarm to send the user a notification for every committed change that matches one of their `Reviews:` paths.

  For example, if the `opt_in_review_path` is set `to //depot/swarmReviews`, users can opt-in to Swarm review notifications by adding that path, or a path such as `//depot/...`, to the `Reviews:` field in their user spec.
- **disable_change_emails**: Optional item. When set to `true`, notifications for committed changes, based on the `Reviews:` field and the users and projects you follow, are disabled. Notifications for reviews and comments will still be sent.

Note
If your Helix server machine already has a review daemon in operation, users receive two notifications for `Reviews:` paths. You may want to deprecate the review daemon in favor of Swarm's change notifications.

Note
"Groups" on page 155 have per-group notification settings. See "Add a group" on page 180 for details.

Global settings

There are many situations that can result in email notifications being sent out to users and groups. Whilst it is possible for a user and group owner to configure their own settings, it is also possible for the system owner to configure the defaults for all users and groups by modifying the settings in the `config.php`.

Note

- **If a group owner is not specified for the group:** group members and users with *super* privileges can configure group notification settings.
- **If one or more group owners are specified for the group:** only group owners and users with *super* privileges can configure group notification settings.

Each notification consists of an `Event` and a `Role`. The `Event` is what happened (for example, a new review was created, a file was submitted) and the `Role` is the role of the user or group who could receive a notification. A user or group can belong to multiple roles, in which case if any of them are set to send a notification, then the user or group will receive a notification.

For example, when a review is voted on (`review_vote`) , there are a number of roles of users, and groups that could be notified:

- **Users:** the user that voted on the review (`is_self`), the author of the review (`is_author`), a user who is a moderator of the project branch the review is in (`is_moderator`), and a user who is a reviewer of the review (`is_reviewer`).
- **Groups:** members of a moderator group for the project branch the review is in (`is_moderator`), and members of a reviewer group for the review (`is_reviewer`).

Configuration options

By default, all notifications are enabled for all roles. The system-wide defaults can be changed by adding the following options into the notifications block of the *SWARM_ROOT*`/data/config.php`. These options are in addition to those described above.

```
<?php
    // this block should be a peer of 'p4'
    'notifications' => array(
        'review_new' => array(
            'is_author' => 'Enabled',
            'is_member' => 'Enabled'
        ),
        'review_files' => array(
            'is_self'      => 'Enabled',
            'is_author'    => 'Enabled',
            'is_reviewer'  => 'Enabled',
            'is_moderator' => 'Enabled'
        ),
        'review_vote' => array(
            'is_self'      => 'Enabled',
            'is_author'    => 'Enabled',
            'is_reviewer'  => 'Enabled',
            'is_moderator' => 'Enabled'
       ),
       'review_required_vote' => array(
            'is_self'      => 'Enabled',
            'is_author'    => 'Enabled',
            'is_reviewer'  => 'Enabled',
            'is_moderator' => 'Enabled'
       ),
       'review_optional_vote' => array(
            'is_self'      => 'Enabled',
            'is_author'    => 'Enabled',
            'is_reviewer'  => 'Enabled',
            'is_moderator' => 'Enabled'
       ),
       'review_state' => array(
            'is_self'      => 'Disabled',
            'is_author'    => 'Enabled',
            'is_reviewer'  => 'Enabled',
            'is_moderator' => 'Enabled'
        ),
        'review_tests' => array(,
             'is_author'    => 'Enabled',
             'is_reviewer'  => 'Enabled',
             'is_moderator' => 'Enabled'
        ),
        'review_changelist_commit' => array(
            'is_author'    => 'Enabled',
            'is_reviewer'  => 'Enabled',
            'is_member'    => 'Enabled',
            'is_moderator' => 'Enabled'
        ),
        'review_comment_new' => array(
            'is_author'   => 'Enabled',
            'is_reviewer' => 'Enabled'
        ),
        'review_comment_update' => array(
            'is_author'   => 'Enabled',
            'is_reviewer' => 'Enabled'
        ),
        'review_comment_liked' => array(
            'is_commenter' => 'Enabled'
        ),
        'review_opened-issue' => array(,
            'is_self'      => 'Enabled',
            'is_author'    => 'Enabled',
            'is_reviewer'  => 'Enabled',
            'is_moderator' => 'Enabled'
        ),
        'review_join_leave' =>   array (
            'is_self'      => 'Enabled'
            'is_author'    => 'Enabled'
            'is_reviewer'  => 'Enabled'
            'is_moderator' => 'Enabled'
        ),
    )
```

Each setting can have one of four possible values to either enable or disable notifications of that type. If multiple settings apply to a given event, then a user will receive a notification if *any* of them are enabled.

- **Enabled**: Notifications for this event and role are enabled, and a user will receive emails by default.
- **Disabled**: Notifications for this event and role are disabled, and a user will not receive emails by default.
- **ForcedEnabled**: Same as for Enabled, but this is forced enabled for all users. An individual user is not able to disable this notification type on their settings page.
- **ForcedDisabled**: As for Disabled, but this is forced disabled for all users. An individual user will not be able to enable this notification type on their settings page.

Unless one of the forced options is used, system wide options can be overridden by individual users and group owners, who can configure which notifications they receive.

Notification Roles

The various roles are as follows:

- **is_self**: This role is the user who changed the state of the review.
- **is_author**: This role is the user who was the author of the review.
- **is_reviewer**: This role includes all users and groups who are listed as being a reviewer on the review.
- **is_member**: This role includes all users who are a member of the project in which the event happened.
- **is_moderator**: This role includes all users and groups who are listed as being a moderator of the project branch in which the event happened.
- **is_follower**: This role includes all users who are followers of the project in which the event happened.
- **is_commenter**: This role is the user who was the author of a comment.

Notification events

An event is the action that causes the notification:

- **review_new**: A new review has been created.
- **review_files**: Files have been added to a review.
- **review_vote**: A user has voted on a review.
- **review_state**: The status of a review has been changed.
- **review_tests**: Automated tests have failed for a review. The first time automated tests pass for a review after a test failure for that review.
- **review_changelist_commit**: Files on a review have been committed.
- **review_comment_new**: A comment has been added to a review.
- **review_comment_update**: A comment on a review has been updated.
- **review_comment_liked**: A user has liked a comment.
- **review_join_leave**: A user or group has joined or left a review.

OVA Management

The Helix Swarm OVA is installed with Ubuntu 16.04 LTS. Ubuntu's "Long-Term Support" releases receive security updates periodically over their 5-year support window. We recommend that use of the OVA involve package updates from time to time, which can be accomplished as follows:

1. Log in to the OVA as ***root***. The password for the root account was established during setup.
2. Update the catalog of available packages:

   ```
   # apt-get update
   ```

3. Download and install any updated packages:

   # **apt-get upgrade**

See "Package management with APT" for more information:
https://help.ubuntu.com/community/AptGet/Howto

Dependency Conflicts

Ubuntu software packages are often dependent on other packages that provide, for example, common libraries, utilities, and configuration. Occasionally, an upgraded package may have differing dependencies than its previous version, which can lead to dependency conflicts that can prevent package updates from completing successfully.

Should this situation occur with the Swarm OVA, use the following command to use "smart" conflict resolution, which attempts to upgrade the most important packages at the expense of less important packages if necessary:

```
# apt-get dist-upgrade
```

Warning
Upgrading packages could potentially make the OVA-hosted Swarm no longer functional. If you use the Swarm OVA in a production environment, perform the package updates on a copy of the OVA and test that Swarm continues to function properly.

P4TRUST

When Swarm is configured to connect to a Helix server (**p4d**) using an SSL connection, Swarm automatically executes the `p4 trust` command, which accepts the SSL fingerprint and creates the file *SWARM_ROOT*`/data/p4trust` containing a list of trusted servers and their fingerprints. If the certificate that **p4d** uses is changed for any reason, then when **p4d** is restarted, Swarm connections to **p4d** fail.

The solution is to delete ***SWARM_ROOT*/data/p4trust**. On the next request to Swarm, `p4 trust` is again automatically executed and Swarm can then connect to **p4d**.

Projects

By default, once a project has been created, any member of the project can edit or delete the project's settings. Projects can also set **Only Owners and Administrators can edit the project**, which prevents all project changes by users who are neither owners or administrators.

Instead of allowing any changes, or preventing all changes, you may want to prevent project members from making select changes, such as to the project's name (and associated identifier), or adjusting the branch definition(s). This is useful when build infrastructure or other tooling treats these details as operational configuration, but you still want members to be able to adjust other aspects of the project configuration.

To do so, edit the *SWARM_ROOT*`/data/config.php` file, and set the following two items, similar to the following example:

```
    'projects' => array(
        'edit_name_admin_only'     => true,
        'edit_branches_admin_only' => true,
    ),
```

- `edit_name_admin_only`: when set to true, only users with *admin* privileges in the Helix server can modify a project's name.
- `edit_branches_admin_only`: when set to true, only users with *admin* privileges in the Helix server can modify a project's branch definition(s).

Both items default to `false`.

Limit adding projects to administrators

By default, any authenticated user can add new projects. Swarm can restrict project creation to users with *admin*-level privileges or higher. Once restricted, Swarm prevents non-administrators from adding projects, and does not display the **+** icon to add a project to non-administrators.

Add or update the following configuration block to the *SWARM_ROOT*`/data/config.php` file, at the same level as the p4 entry:

```
<?php
    // this block should be a peer of 'p4'
    'projects' => array(
        'add_admin_only' => true,
    ),
```

Important
If `add_admin_only` is enabled and `add_groups_only` has one or more groups configured, project creation is only available to users with administrator privileges and who are members of the specified groups.

Limit adding projects to members of specific groups

Swarm can restrict project creation to members of specific groups. The groups and membership need to be defined in the Helix server.

Add or update the following configuration block to the *SWARM_ROOT*/**data/config.php** file, at the same level as the p4 entry:

```
<?php
    // this block should be a peer of 'p4'
    'projects' => array(
        'add_groups_only' => array('wizards', 'slayers', 'phbs'),
    ),
```

Important
If `add_admin_only` is also enabled, project creation is only available to users with administrator privileges **and** who are members of the specified groups.

Project readme

Projects can have a README.md file associated with them that is automatically displayed on the project overview page. This file is read from the root of the project's mainline if it is available, and shown above the activity feed. See "Mainline branch identification" on page 267 for details on configuring the project mainline.

Add or update the following configuration block to the *SWARM_ROOT*/**data/config.php** file, at the same level as the p4 entry:

```
<?php
    // this block should be a peer of 'p4'
    'projects' => array(
        'readme_mode'    => 'restricted',
    ),
```

- `disabled`: the use of README.md file will be disabled and no text content will be shown in the project overview. page.
- `restricted`: the content of the README.md file is displayed, but Markdown support is limited to prevent inclusion of raw HTML and Javascript content. This is the default.
- `unrestricted`: the content of the README.md file is displayed, and Markdown support is unrestricted, allowing full HTML and Javascript to be used. This is insecure as any person with access to the README.md file can add script to the page which would execute as the currently logged in user.

Changing the project sidebar order

The sort order of the project sidebar on the home page is alphabetical. In order to change it to an order based on project popularity, add the following configuration block to the *SWARM_ROOT*/`data/config.php` file, at the same level as the p4 entry:

```
<?php
    // this block should be a peer of 'p4'
    'projects' => array(
        'sidebar_sort_field'    => 'rank',
    ),
```

The options for the `sidebar_sort_field` are `name`, `rank`, and `id`:

- `name`: sort alphabetically according to the project display name. This is the default sort order.
- `id`: sort alphabetically according to the internal project id.
- `rank`: sort according to the popularity of the project. This is based on the number of followers and members that the project has.

Review cleanup

When a review is created in Swarm, it creates its own version of the changelist, leaving the user's own changelist untouched so as not to interfere with the user's ongoing work. Each time new work is submitted to the review, Swarm creates a new changelist so that there is a versioned history of the review.

When the review is finally committed it is Swarm's own changelist that is committed. Swarm's changelists are generally hidden from the users, but the user's changelist will remain open. By default it is necessary for the user to tidy up and remove this changelist themselves after the review has been completed.

There is an option to do this automatically when the review is committed. Configuration for this is expressed with a reviews block in the *SWARM_ROOT*/`data/config.php` file, similar to the following example:

```
<?php
    'reviews' => array(
        'cleanup'               => array(
            'mode'        => 'user', // auto - follow default, user -
present checkbox(with default)
            'default'     => false,  // clean up pending changelists on
commit
            'reopenFiles' => false   // re-open any opened files into the
default changelist
```

```
        ),
    ),
```

By default, this option is enabled but defaults to no clean up (so a user can select the option if they want to when they commit a review).

If the Helix Core user that Swarm is configured to use is a super user, then the user can clean up all user changelists associated with a review. If this is not the case, then the user who commits a review can only clean up changelists that are in their name.

By default, Swarm only cleans up changelists that are owned by the commiting user. In the case where a user is commiting a review that has been contributed to by other users, their changelists will not be cleaned up.

If you want to configure Swarm to clean up all changelists contributing to a review, regardless of whether they are owned by the commiting user, you can do this by granting the Helix Core user that Swarm is configured as 'super' permissions/privileges (rather than just the 'admin' permissions/privileges that swarm requires for its other operations).

Note
There is an API option that allows full cleanup to be executed with super permissions using an external script. This removes the need for the Swarm Helix Core user to have super privileges. See the API notes for details on this.

- **mode**

  If the mode is `user` then a checkbox is displayed when a review is committed, and the user has the option as to whether to clean up changelists or not.

  If the mode is `auto`, then no checkbox is displayed, and all committed reviews will either always be tidied up, or never will be, depending on the value of `default`.

- **default**

  If mode is set to `user`, then this determines whether the checkbox shown on commit is ticked by default or not.

  If mode is set to `auto`, then it determines the action to be taken automatically. If set to `true` then changelists will always be cleaned up, otherwise they will never be cleaned up.

- **reopenFiles**

  If a changelist has files checked out (not shelved), then it cannot be deleted. Setting `reopenFiles` to `true` will mean that when a changelist is cleaned up, any opened files will first be moved to the default changelist, allowing the changelist to be removed.

  If set to `false`, then a changelist with checked out files will not be cleaned up.

  If users normally revert their files after shelving them, then this option may not be needed. If set to `true` then it may result in files appearing in the user's default changelist unexpectedly.

The review cleanup feature is designed to help users keep their workspaces clean, and prevent the proliferation of unwanted changelists after reviews have been approved and committed.

However, it cannot guarantee to be perfect, and because it is taking actions automatically on the part of the user, it may do things that the user doesn't expect. The following caveats should be kept in mind when using this feature.

- The changelists created by Swarm itself will not be touched by this process. There will always be a record of the review history that is kept.
- If the user who commits a review is normally different to the user that did the work, then unless Swarm runs as a super user then many changelists will not be cleaned up.
- If the committer created some of the changelists, then those changelists will be removed. However, changelists created by other users will not be removed.
- If the user has shelved files into changelists without reverting them, then they will still remain in the user's local workspace, and will need to be cleaned up manually.

There is an API endpoint available which can be used by a *super user* to tidy up changelists which can't be removed automatically. See API Endpoints for details.

Review keyword

By default, including the keyword `#review` within a changelist description (separated from other text with whitespace, or on a separate line) informs Swarm that a review should begin when the changelist is shelved or committed. Once a review has begun, Swarm adjusts the keyword with the review's identifier, such as `#review-1234`. This adjustment informs Swarm which review should be updated whenever the original changelist is re-shelved or committed.

Note
Swarm can also accept `[review]` at the start or end of the changelist description, but this form of review keyword is now deprecated and is likely to be removed in a future version of Swarm.

The keyword can be configured with a regular expression so that most any keyword syntax can be used. If you choose to customize the review keyword, take care to choose syntax and terminology that is unlikely to occur in a changelist description, to avoid unexpected Swarm activity.

To configure the review keyword, add the following block to the *SWARM_ROOT*`/data/config.php` file:

```
<?php
    // this block should be a peer of 'p4'
    'reviews' => array(
        'patterns' => array(
            // #review or #review-1234 with surrounding whitespace/eol
            'octothorpe'      => array(
            'regex'   => '/(?P<pre>\s+|^)'
                    .  '\#(?P<keyword>review)(?:-(?P<id>[0-9]+))?'
                    .  '(?P<post>\s+|$)/i',
            'spec'   => '%pre%#%keyword%-%id%%post%',
            'insert' => "%description%\n\n#review-%id%",
            'strip'  => '/^\s*\#review(-[0-9]+)?(\s+|$)'
                    .  '|(\s+|^)\#review(-[0-9]+)?\s*$/i',
            ),

            // [review] or [review-1234] at start
            'leading-square'  => array(
                'regex' => '/^(?P<pre>\s*)'
                        .  '\[(?P<keyword>review)(?:-(?P<id>[0-9]+))?\]'
                        .  '(?P<post>\s*)/i',
                'spec'  => '%pre%[%keyword%-%id%]%post%',
            ),

            // [review] or [review-1234] at end
            'trailing-square' => array(
                'regex' => '/(?P<pre>\s*)'
                        .  '\[(?P<keyword>review)(?:-(?P<id>[0-9]+))?\]'
                        .  '(?P<post>\s*)?$/i',
                'spec'  => '%pre%[%keyword%-%id%]%post%',
            ),
        ),
    ),
```

Multiple patterns can be specified; the first successful match is used and none of the other patterns are evaluated.

The keyword types are grouped under their identifiers. In each group, the `regex` item specifies the regular expression to be used to identify the review keyword in the changelist description. The spec item is used when the review keyword needs to be updated.

Note the use of named capture groups in the `regex`, for example `(?<pre>\s*)`. The values captured during regex matching are used to replace any identically named placeholder values in the spec item that are surrounded by percent % characters. In the example configuration above, the pre and post capture groups and placeholders maintain any whitespace surrounding the review keyword.

For `octothorpe` (or "hashtag") review keywords, these can appear anywhere in the changelist description. The `strip` item is used to ensure that the keyword is removed from the review description if it appears at the start or end of the changelist description. The `insert` item is currently not used; it is included here to prevent future upgrade issues. The intended use case is when a review is started and the changelist does not already contain a review keyword, the insert item would be used to add the review keyword to the changelist description.

For more information on named capture groups in PHP, see: http://www.regular-expressions.info/named.html

Reviews filter

Swarm review sorting on the "Review queues" on page 204 page is customized by using the following configuration block in the `SWARM_ROOT/data/config.php` file:

```
<?php
    // this block should be a peer of 'p4'
    'reviews' => array(
        'filters' => array(
           'result_sorting' => true,
           'date_field' => 'updated', // 'created' displays and sorts by created
date,'updated' displays and sorts by last updated
        ),
    ),
```

result_sorting

Controls whether the **Result order** button is displayed on the "Review queues" on page 204 page.

- Set `result_sorting` to `true` to display the **Result order** button on the "Review queues" on page 204 page, this is the default setting.
- Set `result_sorting` to `false` to remove the **Result order** button from the "Review queues" on page 204 page, reviews are sorted by when they were created.

date_field

Note
The `date_field` setting is only used if `result_sorting` is set to `true`.

`date_field` sets the initial setting for the **Result order** button on the "Review queues" on page 204 page.

- `created`: reviews are sorted by when they were created. This is the default setting.
- `updated`: reviews are sorted by when they were last updated.

When a user starts a new session the **Result order** button is set to the initial sort order set in `date_field`. The review sort order can be changed by the user from the **Result order** button. This setting is remembered even if the user navigates away from the "Review queues" on page 204 page. When the user starts a new session, the **Result order** button is reset back to the setting in `date_field`.

Reviews

This section provides information on how to enforce reviews, disable self-approval of reviews by authors, allow author changes, synchronize the description of a review, and expand reviewer group so the group members are displayed individually on the review page.

Review enforcement

Using the enforce trigger script type option, Swarm can optionally require that a change to be submitted is tied to an approved code review, or the submit is rejected. You would most often use this option to ensure that files within specific depot paths have been reviewed.

Additionally, using the `strict` trigger script type option, Swarm can optionally require that the content of a change to be submitted matches the content of its associated approved code review, or the submit is rejected. Using the `strict` type implies use of the `enforce` type. You would most often use this option to prevent users from making changes prior to submitting an already approved review.

These capabilities are provided via the trigger script included with Swarm, but are not enabled by default nor covered in the standard installation steps.

To enable these capabilities, edit the Perforce trigger table by running the `p4 triggers` command as a user with *super*-level privileges and add the following lines:

```
swarm.enforce.1 change-submit  //DEPOT_PATH1/...
"%//.swarm/triggers/swarm-trigger.pl% -c %//.swarm/triggers/swarm-
trigger.conf% -t enforce -v %change% -p %serverport%"
swarm.enforce.2 change-submit  //DEPOT_PATH2/...
"%//.swarm/triggers/swarm-trigger.pl% -c %//.swarm/triggers/swarm-
trigger.conf% -t enforce -v %change% -p %serverport%"
swarm.strict.1  change-content //DEPOT_PATH1/...
"%//.swarm/triggers/swarm-trigger.pl% -c %//.swarm/triggers/swarm-
trigger.conf% -t strict -v %change% -p %serverport%"
swarm.strict.2  change-content //DEPOT_PATH2/...
"%//.swarm/triggers/swarm-trigger.pl% -c %//.swarm/triggers/swarm-
trigger.conf% -t strict -v %change% -p %serverport%"
```

Note
These trigger table entries assume that the trigger script, swarm-trigger.pl, has been committed to the

Helix server within the `//.swarm` depot. If you have instead copied the trigger script to your Helix server's filesystem (and to the same path on all edge servers in a commit-edge deployment), replace `//.swarm/triggers/swarm-trigger.pl` with the path to the trigger script.

Customize each line by replacing `DEPOT_PATH1` or `DEPOT_PATH2` with the appropriate depot path where you wish to enforce review approvals or to apply a strict comparison of review contents.

The above lines include two examples of each of the two new trigger behaviors. Remove unnecessary lines, or add additional lines for specific depot paths as required.

It is also possible to configure exemptions to the *enforce* and `strict` verifications, for the number of files in a review or the filetypes in a review. For more information on the trigger's options, see "Trigger options" on page 299.

Group exclusion

You may want to exclude specific users from the enforcement provided by these new trigger lines:

1. Create a group in the Helix server whose members should be excluded from enforce or strict review restrictions.
2. Add users who should be excluded to the group.

   Note
   The owner of a group is not counted as a member of the group, unless the owner's userid is listed as a user in the group.

3. Edit the trigger table and add `-g` ***`group_name`*** to each `enforce` or `strict` trigger line as desired.

If the group name is `review_exclusions`, the trigger lines would be similar to:

```
swarm.enforce.1 change-submit  //DEPOT_PATH1/...
"%//.swarm/triggers/swarm-trigger.pl% -c %//.swarm/triggers/swarm-
trigger.conf% -t enforce -v %change% -p %serverport% -g review_exclusions"
swarm.enforce.2 change-submit  //DEPOT_PATH2/...
"%//.swarm/triggers/swarm-trigger.pl% -c %//.swarm/triggers/swarm-
trigger.conf% -t enforce -v %change% -p %serverport% -g review_exclusions"
swarm.strict.1  change-content //DEPOT_PATH1/...
"%//.swarm/triggers/swarm-trigger.pl% -c %//.swarm/triggers/swarm-
trigger.conf% -t strict -v %change% -p %serverport% -g review_exclusions"
swarm.strict.2  change-content //DEPOT_PATH2/...
"%//.swarm/triggers/swarm-trigger.pl% -c %//.swarm/triggers/swarm-
trigger.conf% -t strict -v %change% -p %serverport% -g review_exclusions"
```

Disable self-approval of reviews by authors

The Swarm 2015.2 release provides the ability to disable review approval by authors, even if they are moderators or administrators. This is useful for development workflows where review by others is of paramount importance.

To disable review approval by authors, update the *SWARM_ROOT*/data/config.php file to include the following configuration item within the reviews block:

```
            'reviews' => array(
                'disable_self_approve' => true,
            ),
```

The default value is `false`.

Allow author change

It is possible to allow a user to make themselves the author of a review. This is useful in the case where the original author is no longer available, and someone else needs to take over ownership.

```
            'reviews' => array(
                'allow_author_change' => true,
            ),
```

The default value is `false`. If set to be `true`, then anyone can claim ownership of a review.

Synchronize review description

By enabling the synchronization of review descriptions, it becomes possible to update the description of a review by updating the description of a changelist associated with the review. Whenever an associated changelist is saved, the text of the review will be updated to match.

Note that attaching another changelist to a review, or updating the files in a changelist will not trigger this update, but updating the description of additional attached changelists will.

```
            'reviews' => array(
                'sync_descriptions' => false,
            ),
```

Expand All Limit

The review page has an **Expand All button** that opens all the files within that review. If the number of files is large, clicking the button might affect performance.

The `expand_all_file_limit` disables the button if the number of files in the review exceeds the given value. If the value is set to zero, the button is always enabled and can therefore open all the files.

```
            'reviews' => array(
                'Expand_all_file_limit' => 10,
            ),
```

The default value if the option is not specified is 10.

Expand group reviewers

By default, reviewer group members are not displayed in the *Individuals* area of the reviews page when they interact with a review (vote, comment, update, commit, archive, etc.). This avoids overloading the *Individuals* area with individual avatars if you have large reviewer groups.

Note
An exception to this behavior is when a member of a reviewer group is also an individual required reviewer, in this case their avatar will be displayed in the *Individuals* area.

When **`expand_group_reviewers`** is set to **`true`**, reviewer group members are added to the *Individuals* area of the review page when they interact with the review (vote, comment, update, commit, archive, etc.). If you have large reviewer groups, this might affect performance.

```
            'reviews' => array(
                'expand_group_reviewers' => false,
            ),
```

The default value for **`expand_group_reviewers`** is **`false`**.

Search

Swarm's search feature combines user, group, project, and file path searching, with full content indexing provided by the optional Helix Core Search Tool (previously known as P4Search).

You can download the Helix Core Search Tool. See its release notes for more information.

You configure Swarm search with the following configuration block in the ***SWARM_ROOT*/data/config.php** file:

```
<?php
    // this block should be a peer of 'p4'
    'search' => array(
        'maxlocktime'     => 5000, // 5 seconds, in milliseconds
        'p4_search_host'  => '',   // optional URL to Helix Search Tool
    ),
```

The `maxlocktime` key specifies the maximum amount of time, in milliseconds, that any table within the Helix server should be locked while performing `fstat` command searching. Increasing this value might allow better search results at the expense of potentially blocking other queries on the Helix server. Decreasing this value impacts the Helix server less, but may be insufficient for returning the desired search results.

The `p4_search_host` keys specifies the URL to your installed Helix Core Search Tool. When configured, Swarm issues API calls to the Helix Core Search Tool to take advantage of its full content indexing.

Security

There are many strategies for securing a Swarm installation. This section provides guidance on security features Swarm controls, and recommendations for several areas for the system hosting Swarm.

Require login

Important
Prior to Swarm's 2016.1 release, `require_login` defaulted to `false`. For 2016.1 and later releases, the default is `true`.

By default, Swarm prevents anonymous users from viewing any Helix server resources; users must login to see commits, reviews, etc.

Swarm can be configured to allow anonymous users to access any readable resources (creating or editing resources by anonymous users is not permitted). Add the following configuration block to the *SWARM_ROOT*`/data/config.php` file, at the same level as the p4 entry:

```
<?php
    // this block should be a peer of 'p4'
    'security' => array(
        'require_login' => false, // defaults to true
    ),
```

There is one exception: the `/queue/worker` endpoint is available to any user.

Note
service and operator users are not permitted to login. For more information on these user types, see *Helix Versioning Engine Administrator Guide: Fundamentals*.

Prevent login

When your Helix server has users that should not be able to log in to Swarm, for example *service* users involved with Helix Core replicas, the `prevent_login` configuration item can be used to prevent successful authentication.

Add or update the following configuration block to the *SWARM_ROOT*`/data/config.php` file, at the same level as the p4 entry:

```
<?php
    // this block should be a peer of 'p4'
    'security' => array(
        'prevent_login' => array(
            'service_user1',
            'service_user2',
        ),
    ),
```

`prevent_login` defaults to `array()`, which means no users in your Helix server are prevented from logging into Swarm.

For more information, see "Service users" in *Helix Versioning Engine Administrator Guide: Multi-Site Deployment*.

Sessions

Swarm manages logged-in sessions using cookies in the browser, and PHP session storage on the server. Swarm uses reasonable defaults for the cookie and session lifetimes (measured in seconds); when the lifetime is exceeded users need to login again. To specify session lifetimes, add the following configuration block to the *SWARM_ROOT*`/data/config.php` file, at the same level as the `p4` entry:

```
<?php
    // this block should be a peer of 'p4'
    'session' => array(
        'cookie_lifetime'            => 0, // 0=expire when browser
closed
        'gc_maxlifetime'             => 60*60*24*30, // 30 days
        'remembered_cookie_lifetime'  => 60*60*24*30, // 30 days
    ),
```

- `cookie_lifetime`

  Optional. Limits the lifetime of session cookies, when the **Remember Me** checkbox on the login dialog is unchecked. The default is `0`, which causes the session cookie to expire when the user's browser is closed.

- `gc_maxlifetime`

  Optional. If a session is inactive for the specified number of seconds, it is deleted and the user is logged out. The default is `60*60*24*30` seconds (30 days). Note that by default, the user's Perforce ticket expires after 12 hours, which also causes them to be logged out.

- `remembered_cookie_lifetime`

  Optional. Limits the lifetime of session cookies when the **Remember Me** checkbox on the login dialog is checked. The default is `60*60*24*30` seconds (30 days).

X-Frame-Options header

By default, Swarm emits a `X-Frame-Options` HTTP header set to `SAMEORIGIN`. This prevents embedding of the Swarm interface into other web pages, which avoids *click-jacking* attacks.

If your deployment of Swarm needs to be integrated into another web interface, you can adjust the `X-Frame-Options` header by adjusting the `x_frame_options` item within the security configuration block, found in the *SWARM_ROOT*`/data/config.php` file. For example:

```
<?php
    // this block should be a peer of 'p4'
    'security' => array(
        'x_frame_options' => value,
        ),
    ),
```

Where value can be one of:

- `'SAMEORIGIN'` - Swarm can only be displayed in a frame hosted on the same domain.
- `'DENY'` - Swarm cannot be displayed in a frame.
- `'ALLOW-FROM` *URI*`'` - Swarm can only be displayed in a frame hosted on the specified URI.
- `false` - The `X-Frame-Options` header is not emitted, so Swarm can be embedded without restriction.

For more information on the `X-Frame-Options` header, see this Mozilla Developer Network article.

For more information on click-jacking attacks, see this Wikipedia article.

Disable commit

Swarm provides the ability to commit reviews within the Swarm interface. You may want to disable this capability to prevent reviews from being committed by someone other than the review's author. When disabled, the **Approve and Commit** (and **Commit** if the review is already approved) option is removed from the list of states available to a code review.

To disable commits, set `disable_commit` to `true` within the reviews item in the *SWARM_ROOT*`/data/config.php` file. For example:

```
<?php
    // this block should be a peer of 'p4'
    'reviews' => array(
        'disable_commit' => true,
        ),
    ),
```

Restricted Changes

The Helix server provides two changelist types: `public` (the default), and `restricted`. Swarm honors restricted changelists by preventing access to the changelist, and any associated comments or activity related to the changelist.

If a user has *list*-level privileges to at least one file in the changelist, Swarm allows the user to see the changelist and any of the files they have permission to see.

To prevent unintended disclosures, email notifications for restricted changes are disabled by default. To enable email notifications for restricted changes, set `email_restricted_changes` to `true` within the security item in the *SWARM_ROOT*`/data/config.php` file. For example:

```
<?php
    // this block should be a peer of 'p4'
    'security' => array(
        'email_restricted_changes' => true,
        ),
    ),
```

Note
When `email_restricted_changes` is set to `true`, email notifications for restricted changes are sent to all interested parties with no permissions screening. These notifications might disclose sensitive information.

Swarm can only report on changes that the configured *admin*-level user has access to. When using restricted changes, we advise that you grant the Swarm*admin*-level user access to the restricted files and set `require_login = true` to avoid leaking information to unauthenticated users.

Limit adding projects to administrators

Important
For Swarm 2016.1, the configuration item `add_project_admin_only` was moved from the `security` block to the `projects block`, and the item was renamed to `add_admin_only`. The functionality of this configuration item remains unchanged.

If you do not update your *`SWARM_ROOT`*`/data/config.php` configuration file, the old configuration for restricting project creation to administrators continues to work.

If you add the new configuration item `add_admin_only` to the `projects` block, it takes precedence over any remaining `add_project_admin_only` setting in the `security` block.

Limit adding projects to members of specific groups

Important
For Swarm 2016.1, the configuration item `add_project_groups` was moved from the `security` block to the `projects block`, and the item was renamed to `add_groups_only`. The functionality of this configuration item remains unchanged.

If you do not update your *`SWARM_ROOT`*`/data/config.php` configuration file, the old configuration for restricting project creation to specific groups continues to work.

If you add the new configuration item `add_groups_only` to the `projects` block, it takes precedence over any remaining `add_project_groups` setting in the `security` block.

IP address-based protections emulation

A Helix server can be configured via *protections* to restrict access to a depot in a variety of ways, including by IP address. As Swarm is a web application acting as a client to the Helix server, often with *admin*-level privileges, Swarm needs to emulate IP address-based restrictions. It does so by checking the user's IP address and applying any necessary restrictions during operations such as browsing files, viewing file content, viewing and adding comments on files.

Swarm also emulates proxy-based protections, in addition to regular IP-based protections emulation. However, Swarm does not detect whether it is connecting to a Helix Proxy or not; it merely attempts to emulate protections table entries that use proxy syntax.

IP address-based protections emulation is enabled by default. Swarm performs somewhat faster without this emulation; if you do not require them for your Swarm installation these can be disabled by setting the configuration:

```
<?php
    // this block should be a peer of 'p4'
    'security' => array(
        'emulate_ip_protections' => false,
    ),
```

Known limitations

- Notification e-mails for reviews or commits include the list of affected files. Swarm cannot reliably know the IP address used to retrieve that e-mail, and makes no attempt to filter the files and their depot paths nor any details included in the description. However, when a user follows a link from the notification e-mail to a restricted resource, that access is denied.
- Swarm filters comments from activity streams, but any comments created prior to upgrading to the 2013.3 release cannot be filtered and may leak sensitive information.
- Swarm displays a comment count in code review queues, code reviews, jobs, and activity streams, but the count does not account for any comments that may be hidden from the user due to association with files the user is restricted from viewing.
- Should Swarm users connect to Swarm via a proxy or VPN, the protections will generally use the IP address of the proxy/VPN.
- When the user's IP address and Swarm's IP address both have restrictions applied, the user experiences the most constraining of the two IP address-based restrictions; Swarm cannot bypass restrictions applied to itself.
- Swarm performs a variety of operations with *admin*-level privileges, on behalf of a user. Even if the Helix server has IP-based, or userid-based protections, installed to prevent access to some or most of its versioned data, Swarm typically does have access to this data. Therefore, *Swarm cannot guarantee that no information leakage will occur*, particularly when custom modules are in use, or Swarm source has been customized.

For more information, see "Authorizing Access" in *Helix Versioning Engine Administrator Guide: Fundamentals*.

Disable system info

Swarm provides a **System Information** page, available to users with *admin* or *super* privileges, which displays information about Helix server that Swarm is configured to use, as well as PHP information and the Swarm log file.

While this information can be invaluable when communicating with Perforce support engineers, you may wish to prevent disclosure of any system information. The **System Information** page can be disabled for all users by adding the following configuration block to the *SWARM_ROOT*/data/config.php file, at the same level as the p4 entry:

```
<?php
    // this block should be a peer of 'p4'
    'security' => array(
        'disable_system_info' => true, // defaults to false
    ),
```

Once disabled, the **System Information** link disappears from the About Swarm dialog, and 403 errors are generated for any attempts to browse to the **System Information** page.

HTTP client options

Swarm permits configuration of options that are passed through to the underlying Zend Framework 2's HTTP client. These options can be used to specify SSL certificate locations, request timeouts, and more, and can be specified globally or per host.

Here is an example configuration:

```
<?php
    // this block should be a peer of 'p4'
    'http_client_options' => array(
        'timeout'       => 5,

        // path to the SSL certificate directory
        'sslcapath'     => '',

        // the path to a PEM-encoded SSL certificate
        'sslcert'       => '',

        // the passphrase for the SSL certificate file
        'sslpassphrase' => '',

        // optional, per-host overrides;
        // host as key, array of options as value
        'hosts'     => array(
            'jira.example.com'  => array(
                'sslcapath'     => '/path/to/certs',
                'sslcert'       => 'jira.pem',
                'sslpassphrase' => 'keep my JIRA secure',
                'timeout'       => 15,
            ),
        ),
    ),
```

See the Zend Framework 2's Socket Adapter documentation for more information.

Warning
While it is possible to use a self-signed SSL certificate, adding the configuration to do so disables certificate validity checks, making connections to the configured host less secure. **We strongly recommend against using this configuration option.**

However, if you need to configure continuous integration, deployment, or JIRA connections and those connections must use a self-signed SSL certificate, set the `sslallowselfsigned` item to `true` for the specific host that needs it, as in the following example:

```
<?php
    // this block should be a peer of 'p4'
    'http_client_options' => array(
        'hosts'     => array(
            'jira.example.com'  => array(
                'sslallowselfsigned'  => true,
            ),
        ),
    ),
```

Strict HTTPS

To improve the security when users work with Swarm, particularly if they need to do so outside of your network, Swarm provides a mechanism that tries to force web browsers to use HTTPS. When enabled, Swarm's behavior changes in the following ways:

- HTTP requests to Swarm include a meta-refresh to the HTTPS version. If a load balancer handles encryption before requests reach Swarm, the meta-refresh should be disabled. See below.
- A strict transport security header is included for all requests, which pins the browser to using HTTPS for your Swarm installation for 30 days.
- All qualified URLs that Swarm produces use HTTPS for the scheme.
- Cookies are flagged as HTTPS-only.

Here is an example of how to enable strict HTTPS:

```
<?php
    // this block should be a peer of 'p4'
    'security' => array(
        'https_strict'          => true,
        'https_strict_redirect' => true, // optional; set false to avoid
meta-refresh
        'https_port'            => null, // optional; specify if HTTPS is
                                         // configured on a non-standard
port
    ),
```

When the `https_strict_redirect` item is set to `false`, Swarm does not add a meta-refresh for HTTP clients. This prevents an endless redirect when a load balancer in front of Swarm applies HTTPS to the client-to-load balancer connection, but not the load balancer-to-Swarm connection.

Apache security

There are several Apache configuration changes that can improve security for Swarm:

- **Disable identification**

  By default, each Apache response to a web request includes a list of tokens identifying Apache and its version, along with any installed modules and their versions. Also, Apache can add a signature line to each response it generates that includes similar information. By itself, this identification information is not a security risk, but it helps would-be attackers select attacks that could be successful.

  To disable Apache identification, add the following two lines to your Apache configuration:

  ```
  ServerSignature Off
  ServerTokens ProductOnly
  ```

- **Disable TRACE requests**

  `TRACE` requests cause Apache to respond with all of the information it has received, which is useful in a debugging environment. `TRACE` can be tricked into divulging cookie information, which could compromise the credentials being used to login to Swarm.

  To disable `TRACE` requests, add the following line to your Apache configuration:

  ```
  TraceEnable off
  ```

- **Update SSL configuration**

  Swarm works correctly with an SSL-enabled Apache. Several attacks on common SSL configurations have been published recently. We recommend that you update your Apache configuration with the following lines:

  ```
  <IfModule mod_ssl.c>
      SSLHonorCipherOrder On
      SSLCipherSuite ECDHE-RSA-AES128-SHA256:AES128-GCM-
  SHA256:RC4:HIGH:!MD5:!aNULL:!EDH
      SSLCompression Off
  </IfModule>
  ```

PHP security

There are several PHP configuration changes that can improve security for Swarm:

- **Disable identification**

  By default, PHP provides information to Apache that identifies that it is participating in a web request, including its version.

  To disable PHP identification, edit your system's php.ini file and change the line setting `expose_php` to:

  ```
  expose_php = Off
  ```

- **Remove scripts containing phpinfo()**

  During module development or other debugging, you may need to call `phpinfo()`, which displays PHP's active configuration, compilation details, included modules and their configuration. Typically, you would add a script to Swarm's public directory containing:

  ```
  <?php phpinfo() ?>
  ```

  Any such scripts should be removed from a production instance of Swarm.

Short links

Short links work with your Swarm installation's current hostname, but you have the option of registering/configuring an even shorter hostname to make shareable file/directory links as short as possible.

1. Register a short domain name, or if you control your own DNS server, a short domain name for your network.
2. Point the short domain name at your Swarm host.

3. Edit the *SWARM_ROOT*`/data/config.php` file and add the following configuration block:

```
<?php
    // this block should be a peer of 'p4'
    'short_links' => array(
        'hostname'  => 'myho.st',
    ),
```

Replace `myho.st` with the short domain name you registered/configured.

If your Swarm is configured to use HTTPS, a custom port, a sub-folder, or any combination of these custom installation options, the short links configuration block should look like:

```
<?php
    // this block should be a peer of 'p4'
    'short_links' => array(
        'external_url' => 'https://myho.st:port/sub-folder',
    ),
```

Replace *`myho.st`* with the short domain name you have registered/configured.

If you have not configured Swarm to use HTTPS, replace `https://` with `http://`.

If you have configured Swarm to run on a custom port, replace *`:port`* with the correct custom port. Otherwise, remove *`:port`*.

If you have configured Swarm to run in a sub-folder, replace */`sub-folder`* with the correct sub-folder name. Otherwise, remove */`sub-folder`*.

Important
The `external_url` configuration item is only honored if you have also configured the `"external_url" on page 260` item within the `"Environment" on page 259` configuration item as well. Otherwise, Swarm could generate short links that cannot correctly link to their corresponding full URLs.

When `external_url` is configured, the `hostname` configuration item is ignored.

swarm_root

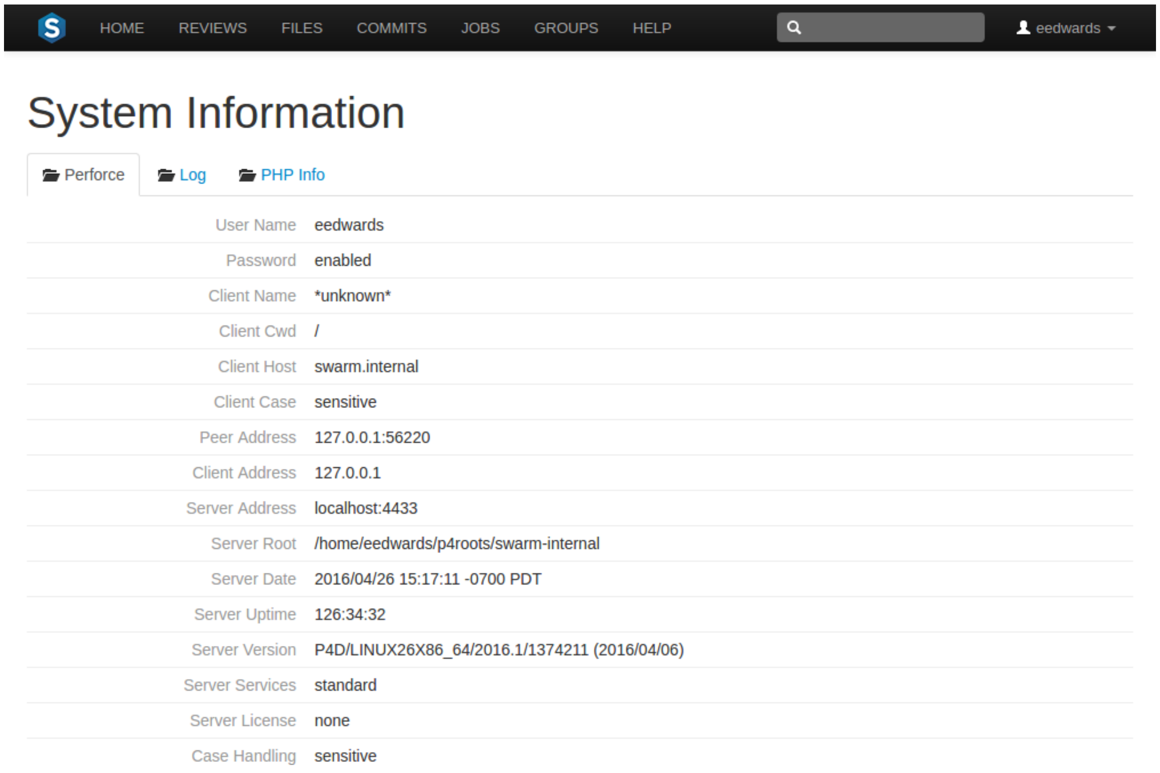
`swarm_root` refers to the directory where Swarm lives in the filesystem. Depending on how you installed Swarm, the default location could be:

- For a package or OVA installation: `/opt/perforce/swarm`
- For a tarball installation: wherever you unpacked Swarm. If you are unsure, you could check your web server's configuration to discover where Swarm exists.

System Information

The System Information page is available to users with *admin* or *super* privileges on the About Swarm dialog. Click the **System Information** link on the dialog to display the System Information page.

The System Information page provides details that can be useful to Perforce support engineers when you ask them for assistance.



System Information

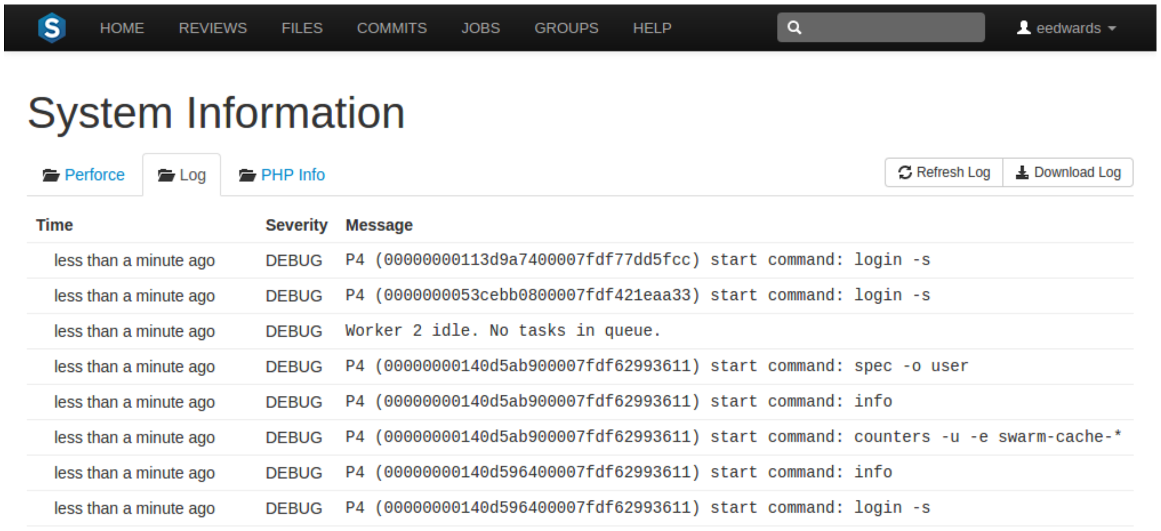
Perforce | Log | PHP Info

User Name	eedwards
Password	enabled
Client Name	*unknown*
Client Cwd	/
Client Host	swarm.internal
Client Case	sensitive
Peer Address	127.0.0.1:56220
Client Address	127.0.0.1
Server Address	localhost:4433
Server Root	/home/eedwards/p4roots/swarm-internal
Server Date	2016/04/26 15:17:11 -0700 PDT
Server Uptime	126:34:32
Server Version	P4D/LINUX26X86_64/2016.1/1374211 (2016/04/06)
Server Services	standard
Server License	none
Case Handling	sensitive

The initial display is the **Perforce** tab, which provides information similar to the `p4 info` command.

Log

Click the **Log** tab to display the most recent entries, up to 1 megabyte, in Swarm's log file, which resides in *`SWARM_ROOT`*`/data/log`. Review the logging levels for Swarm logs to ensure that the entries you want to see are included.

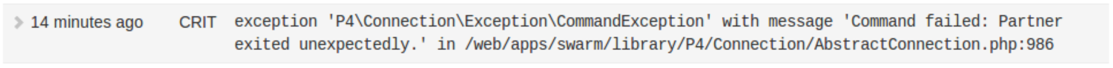


```
Time                 Severity  Message
less than a minute ago  DEBUG  P4 (00000000113d9a7400007fdf77dd5fcc) start command: login -s
less than a minute ago  DEBUG  P4 (0000000053cebb0800007fdf421eaa33) start command: login -s
less than a minute ago  DEBUG  Worker 2 idle. No tasks in queue.
less than a minute ago  DEBUG  P4 (00000000140d5ab900007fdf62993611) start command: spec -o user
less than a minute ago  DEBUG  P4 (00000000140d5ab900007fdf62993611) start command: info
less than a minute ago  DEBUG  P4 (00000000140d5ab900007fdf62993611) start command: counters -u -e swarm-cache-*
less than a minute ago  DEBUG  P4 (00000000140d596400007fdf62993611) start command: info
less than a minute ago  DEBUG  P4 (00000000140d596400007fdf62993611) start command: login -s
```

Click the **Refresh Log** button to load the latest log entries, up to 1 megabyte.

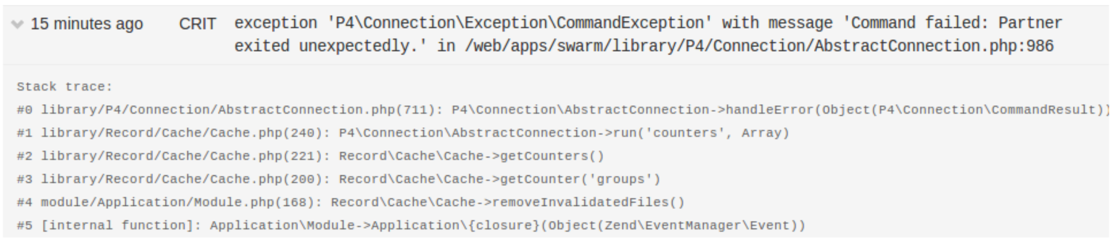
Click the **Download Log** button to download the log and all of its log entries.

If your log entries include a critical error, an arrow appears:



```
14 minutes ago   CRIT   exception 'P4\Connection\Exception\CommandException' with message 'Command failed: Partner
                        exited unexpectedly.' in /web/apps/swarm/library/P4/Connection/AbstractConnection.php:986
```

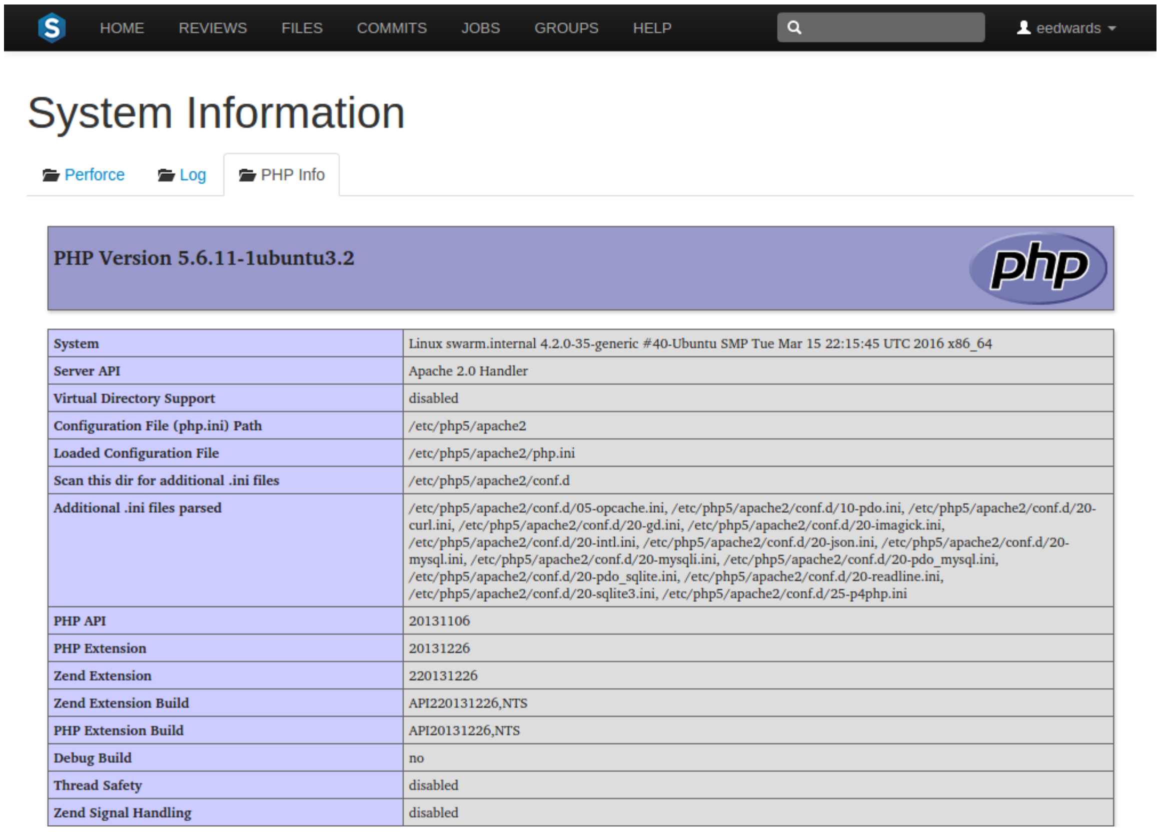
Click the error to display the stack trace that accompanies the error:



```
15 minutes ago   CRIT   exception 'P4\Connection\Exception\CommandException' with message 'Command failed: Partner
                        exited unexpectedly.' in /web/apps/swarm/library/P4/Connection/AbstractConnection.php:986

Stack trace:
#0 library/P4/Connection/AbstractConnection.php(711): P4\Connection\AbstractConnection->handleError(Object(P4\Connection\CommandResult))
#1 library/Record/Cache/Cache.php(240): P4\Connection\AbstractConnection->run('counters', Array)
#2 library/Record/Cache/Cache.php(221): Record\Cache\Cache->getCounters()
#3 library/Record/Cache/Cache.php(200): Record\Cache\Cache->getCounter('groups')
#4 module/Application/Module.php(168): Record\Cache\Cache->removeInvalidatedFiles()
#5 [internal function]: Application\Module->Application\{closure}(Object(Zend\EventManager\Event))
```

PHP Info

Click the **PHP Info** tab to display PHP's own information display generated by executing `phpinfo()`, PHP's internal diagnostic display.



System Information

PHP Version 5.6.11-1ubuntu3.2

System	Linux swarm.internal 4.2.0-35-generic #40-Ubuntu SMP Tue Mar 15 22:15:45 UTC 2016 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-curl.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-imagick.ini, /etc/php5/apache2/conf.d/20-intl.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-pdo_sqlite.ini, /etc/php5/apache2/conf.d/20-readline.ini, /etc/php5/apache2/conf.d/20-sqlite3.ini, /etc/php5/apache2/conf.d/25-p4php.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled

Trigger options

The Swarm trigger script, `swarm-trigger.pl`, provides the following command line options and configuration items.

Command-line options

Synopsis

```
swarm-trigger.pl -t type -v ID [-p port] [-r] [-g group] [-c config_file]
```

```
swarm-trigger.pl -o
```

```
swarm-trigger.pl -h
```

Informational options

The following options perform no processing, and simply provide information useful to a Swarm administrator. These are not intended to be used within trigger entries in the Helix server.

- **-h**

  Displays a list of the available options, some guidance on its usage, and a copy of the trigger table entries that should be configured in the Helix server to execute the script in its current path.

- **-o**

  Displays a copy of the trigger table entries that should be configured in the Helix server to execute the script in its current path.

Operational options

The following command-line options are used in the trigger table entries in the Helix server to specify how the Swarm trigger script should be executed.

- `-t` *`type`*

  Specifies the type of processing that the trigger script undertakes. *`type`* can be one of:

  - `changesave`: this type should be used with Helix Core`form-save` events for change forms, and informs Swarm when a changelist is created or modified.
  - `commit`: this type should be used with Helix Core`change-commit` events, and informs Swarm when a changelist is committed.
  - `enforce`: this type is used to verify that commits to specific depot paths are associated with approved reviews. If a commit includes a file within the specified depot path, and it is not associated with a review (or a review that is not approved), the commit is rejected.

    Using the enforce type prevents users from committing changes to specific depot paths without those changes being reviewed and approved,
  - `group`: this type should be used with Helix Core`form-commit` events for group forms, and informs Swarm when a group is created or modified.
  - `groupdel`: this type should be used with Helix Core`form-delete` events for group forms, and informs Swarm when a group is deleted.
  - `job`: this type should be used with Helix Core`form-commit` events for job forms, and informs Swarm when a job is created or modified.
  - `shelve`: this type should be used with Helix Core`shelve-commit` events, and informs Swarm when a changelist is shelved, which can create or update a review.

    - `strict`: this type is used to verify that the file content in a commit matches the file content of its associated approved review. If one or more files in a commit do not match the content of the file in its associated review, the commit is rejected.

      Using the `strict` type prevents users from making changes to the file content after a review has been approved and then submitting the unapproved changes.

      > **Note**
      > `strict` implies `enforce`.

    - `user`: this type should be used with Helix Core`form-commit` events for user forms, which informs Swarm when a user is added or modified.
    - `userdel`: this type should be used with Helix Core`form-delete` events for user forms, which informs Swarm when a user is deleted.

  > **Important**
  > You cannot mix Swarm trigger types with unrelated Helix Core events; the behavior is undetermined, and the information required for each type of processing may not be available to the trigger.

- `-v ID`

  Specifies ***`ID`***, which is the identifier for the current trigger type.

  When the ***`type`*** is `job`, `user`, `userdel`, `group`, `groupdel`, or `changesave`, ***`ID`*** should be `%formname%` to specify the specific form identifier Swarm should process.

  When the ***`type`*** is `shelve`, `commit`, `enforce`, or `strict`, ***`ID`*** should be `%change%` to specify the specific changelist Swarm should process.

- `-p` ***`port`*** **(optional)**

  Specifies the Helix server port (`P4PORT`). This value is optional, and is only used for types `enforce` or `strict` as the trigger has to run its own commands against the Helix server during its processing.

- `-r` **(optional)**

  Specifies that, when types `enforce` or `strict` are being processed, the verifications should only be performed on commits that are currently in review.

- `-g` ***`group`*** **(optional)**

  Specifies a ***`group`*** to exclude from `enforce` or `strict` verifications. Members of the group (including sub-groups) are not subject to `enforce` or `strict` verifications.

- `-c` ***`config_file`*** **(optional)**

  Specifies an optional ***`config_file`*** which is used to specify configuration items.

Configuration items

The following configuration items are used in the `swarm-trigger.conf` (or another file, if the `-c` option is used in the trigger entries).

- **SWARM_HOST (required)**

  Specifies the host URL of your Swarm instance, with the leading `http://` or `https://`. For example: `https://myswarm.url`

- **SWARM_TOKEN (required)**

  A token used when talking to Swarm. To obtain the token, log into Swarm as a user with *super* privileges and select "About Swarm" on page 174 from the user menu in the main navigation bar.

  You can also manually create additional tokens. Tokens are empty files stored within *`SWARM_ROOT`*`/data/queue/tokens` (the filename is the token and is reported on the **About Swarm** dialog), that should be readable by the web server.

  You might manually create additional tokens to allow other processes to talk to Swarm, such as JIRA build tasks, and to selectively invalidate access to Swarm without interfering with regular Swarm operations.

- **ADMIN_USER (optional)**

  When `enforce` or `strict` verifications are to be performed, you may need specify a username of a user in the Helix server that has admin privileges. If you do not specify a username, the trigger script uses the Helix Core user set in the environment.

- **ADMIN_TICKET_FILE (optional)**

  When `enforce` or `strict` verifications are to be performed, you may need specify the path to the `.p4tickets` file, if your Helix server tickets file is not the default `$HOME/.p4tickets`.

  Important
  Ensure that the ticket belongs to a user with *admin* privileges in the Helix server, and is a member of a group with an `unlimited` or very long ticket timeout. If this user's authentication times out, enforce and `strict` verifications stop working.

- **P4_PORT (optional)**

  When `enforce` or `strict` verifications are to be performed, you may need to set the port value (`P4PORT`) of the Helix server, particularly if the Helix server is on a non-standard port, or if the Helix server is not using the default hostname.

- **P4 (optional)**

  Specifies the full path to `p4`, the Helix Core command-line client. This is only required when `p4` is not found in the `PATH` of the Helix server environment, and when `enforce` or *strict* verifications are to be performed.

- **EXEMPT_FILE_COUNT (optional)**

  When set to a positive integer, commits with a file count greater or equal to this value are exempt from `enforce` or `strict` verifications.

- **EXEMPT_EXTENSIONS (optional)**

  A comma-separated list of file extensions. Commits with files having *only* these extensions are exempt from `enforce` or `strict` verifications.

Unapprove modified reviews

By default, when an approved review is committed or updated, Swarm changes the state to **Needs Review** if the files have been modified since the review was approved.

If one or more files in a review has the filetype +k (`ktext`), this behavior is undesirable because the files will appear to be modified as the Helix server replaces RCS keywords with their current values.

This behavior can be disabled. Edit the *SWARM_ROOT*`/data/config.php` file, and add or update the `unapprove_modified` item to `false`, within the reviews configuration block. For example:

```
<?php
    // this block should be a peer of 'p4'
    'reviews' => array(
        'unapprove_modified'  => false,
    ),
```

For for information on file types, see "File Types" in *P4 Command Reference*.

Uninstall Swarm

This section covers the steps required to uninstall Swarm.

Background

The bulk of Swarm's metadata (activity, comments, review records, followers) is stored in p4 keys under `swarm-*`. If you are using a 2012.1+ server, Swarm also defines user groups for each project that you define. The names of these groups correspond 1-to-1 with projects, for example `swarm-project-fantastico`. Swarm manages a pool of client workspaces that it uses to shelve and commit files. These clients are named `swarm-`*`{uuid}`*, for example `swarm-5ad4a9c0-06e7-20eb-897f-cbd4cc934295`.

Uninstall steps

1. Uninstall the Swarm triggers.
2. Remove your web server's virtual host configuration for Swarm.

3. Restart your web server.
4. Delete groups/clients/keys that are prefixed with `swarm-*`.

   Note
   The clients could contain shelved files for reviews. Determine how you want to handle those files prior to deleting the clients.

5. Additional indexed information is stored in the database file `db.ixtext`. Unfortunately, indexed jobs and other generic indexed information would be lost if this table was simply removed, and modifying the database file can be a dangerous operation in a number of Helix server deployment scenarios.

   Important
   Contact Perforce support for assistance if you feel the need to remove Swarm's indexed information: support@perforce.com.

6. Rebuild the job index. The best approach is to run:

   ```
   $ p4 jobs -R
   ```

   which rebuilds the `db.ixtext` table. There are two caveats that likely require discussion with support@perforce.com:

   - If you make use of the unsupported `p4 index` command, you **cannot** use this approach, as it would remove all of your indexes.
   - If you have indexing turned on for the domain table, you must also run:

     ```
     $ p4d -xf index.domain.owner
     ```

7. If the `P4.Swarm.URL` or `P4.Swarm.CommitURL` properties were set (for details, see "Client integration" on page 243 and "Commit-edge deployment" on page 247 respectively), they should be unset to prevent P4V (and potentially other clients) from attempting Swarm operations:

```
$ p4 property -d -n P4.Swarm.URL
$ p4 property -d -n P4.Swarm.CommitURL
```

If the `P4.Swarm.URL` or `P4.Swarm.CommitURL` properties were set using sequence numbers, you need to add the `-s`*N* flag to the commands, where *N* is a sequence number.

To discover all of the definitions of the `P4.Swarm.*` properties and their sequence numbers, run the command:

```
$ p4 property -Al | grep P4.Swarm
P4.Swarm.CommitURL = https://myswarm.url/ (any) #none
P4.Swarm.CommitURL = https://myswarm1.url/ (any) #1
P4.Swarm.URL = https://myswarm.url/ (any) #none
P4.Swarm.URL = https://myswarm3.url/ (any) #3
P4.Swarm.URL = https://myswarm2.url/ (any) #2
P4.Swarm.URL = https://myswarm1.url/ (any) #1
```

To delete all of these property definitions, you would run the following commands:

```
$ p4 property -d -n P4.Swarm.URL
$ p4 property -d -n P4.Swarm.URL -s1
$ p4 property -d -n P4.Swarm.URL -s2
$ p4 property -d -n P4.Swarm.URL -s3
$ p4 property -d -n P4.Swarm.CommitURL
$ p4 property -d -n P4.Swarm.CommitURL -s1
```

Upgrade index

Swarm " Upgrade index" on page 119 can be configured to suit the performance of your Swarm system via the following config items.

status_refresh_interval

The `status_refresh_interval` sets the refresh rate of the index upgrade status page. The default is 10 seconds.

```
<?php
    // this block should be a peer of 'p4'
    'upgrade' => array(
        'status_refresh_interval' => 10, //Refresh page every 10 seconds
    ),
```

batch_size

The `batch_size` sets the number of reviews held in memory and processed for each batch. This value can be increased or decreased depending on your Swarm machines system memory. The default is 1000 reviews.

For example:

If the Swarm machine does not have a lot of memory, for instance 128MB, a `batch_size` of 1000 might be too high and may make the upgrade run very slowly. In this instance reduce the `batch_size` to a more manageable size.

```
<?php
    // this block should be a peer of 'p4'
    'upgrade' => array(
        'batch_size' => 1000,    //Fetch 1000 reviews to lower memory usage
    ),
```

Important
The Swarm index must be upgraded to ensure that the Swarm review history is displayed in the correct order.

Note
Only required the first time you upgrade your Swarm system to 2017.3 or later. Subsequent Swarm upgrades do not require the index to be upgraded.

Note
Not required if your Swarm system has less than 100 reviews. In this case Swarm will automatically upgrade the index on the fly.

Note
Not required if this is a new Swarm installation.

Workers

Helix Swarm uses background processes, called *workers*, to respond to events in the Helix server. The default number of workers is 3, and each worker processes events for up to 10 minutes. When a worker terminates, a new one is spawned.

Note
Each worker maintains a connection to the Helix server for the duration of its lifetime. This may impact your Helix server management practices.

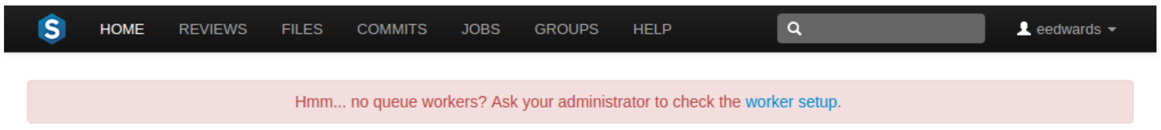
Worker status

To determine the current status of workers, visit the URL:
`https://`*`myswarm.url`*`/queue/status`

The response is formatted in JSON, and looks like this:

```
{"tasks":0,"futureTasks":1,"workers":3,"maxWorkers":3,"workerLifetime":"59
5s"}
```

During normal use of Swarm, the following error message appears for logged-in users when Swarm detects that no workers are running:



Hmm... no queue workers? Ask your administrator to check the worker setup.

Worker configuration

To adjust the configuration for workers, add a configuration block to the *SWARM_ROOT*`/data/config.php` file:

```
<?php
    // this block should be a peer of 'p4'
    'queue'  => array(
        'workers'             => 3,    // defaults to 3
        'worker_lifetime'     => 595,  // defaults to 10 minutes (less 5
seconds)
        'worker_task_timeout' => 1800, // defaults to 30 minutes
        'worker_memory_limit' => '1G', // defaults to 1 gigabyte
    ),
```

where:

- `workers` specifies the number of worker processes that should be available. The default is 3. The cron job ensures that new worker processes are started when necessary. If the limit is reached or exceeded, new worker processes are not started.
- `worker_lifetime` specifies the amount of time in seconds that a worker process should run for. The default is 595 seconds (10 minutes less 5 seconds). If a worker process exceeds this limit while processing a task, it will complete the active task and then terminate. `worker_lifetime` does not cause tasks to terminate mid-processing.
- `worker_task_timeout` specifies the maximum amount of time in seconds that a worker process can spend processing a single task. The default is 1800 seconds (30 minutes). This is useful for terminating workers that might get stalled in a variety of situations.
- `worker_memory_limit` specifies the maximum amount of memory that a worker process is allowed to use while processing a task. The default is 1G (1 gigabyte).

Manually start workers

To kick off a new worker process, visit the URL: `https://myswarm.url/queue/worker`

When the number of workers running matches the configured limit, the requested worker process is not started.

Note
This technique does start a worker, but it lasts only for its configured lifetime. Typically, you would always want at least one worker running. See "Set up a recurring task to spawn workers" on page 99 for details.

Manually restart workers

To restart an idle worker process, remove its lock file:

```
rm data/queue/workers/worker_id
```

A worker process that is busy processing a task will continue operation until its task is complete. Immediately afterwards, if the worker notices that its lock file is missing it exits.

If you have a recurring task to start workers, the recurring task starts a fresh worker, if necessary. See "Set up a recurring task to spawn workers" on page 99 for details.

10 | Extending Swarm

Helix Swarm is built with open source technologies, using modern development methods, resulting in a platform that is capable, modular, and can be extended in a variety of ways. While this section is currently incomplete, it does identify the technologies that make up Swarm and provides notes regarding some of the available extension points.

Resources

Swarm is built with a number of different technologies. As such it can be a bit of a challenge to ramp up on if you haven't done web development before. This section lists some of the best resources we've found for learning how to develop for Swarm.

jQuery

A free three hour course on jQuery basics. Highly recommended if you haven't used jQuery before. It also covers some JavaScript and CSS basics:

http://try.jquery.com

JavaScript Resources

A free, in-browser Javascript course:

https://www.codecademy.com/tracks/javascript-combined

PHP Resources

- A free PHP tutorial:

  http://www.w3schools.com/php/default.asp

- A handy PHP cheat sheet:

  https://i.emezeta.com/weblog/emezeta-php-card-v0.2.png

Zend Framework 2 Resources

- An intense tutorial, but it does cover most of the basics:

  http://framework.zend.com/manual/2.0/en/user-guide/overview.html

- Really nice high level architecture overview by a Zend engineer:

  https://speakerdeck.com/ezimuel/mvc-plus-events-plus-modules-the-new-architecture-of-zf2

Development mode

Swarm has a *development* mode that, when enabled, changes Swarm's behavior in the following ways:

- CSS and JavaScript code is not aggregated. Each CSS or JavaScript file is fetched individually, which makes it much easier to identify where styles or functions exist and how they apply to Swarm. Due to the additional HTTP requests, development mode should not be used in production environments.
- Errors and exceptions that may occur are displayed in Swarm's UI. This is particularly useful to developers of Swarm modules. As the error information might disclose system paths or configuration, development mode should not be used in production environments.

To enable development mode:

Adjust the *SWARM_ROOT*`/data/config.php` file to include:

```
<?php
    return array(
        'p4' => ...
        'environment'   => array(
            'mode'   => 'development'
        )
    );
```

To disable development mode:

Adjust the ***SWARM_ROOT***`/data/config.php` file to exclude the `'mode' => 'development'` line.

Modules

A Swarm module is a folder that exists within the modules folder within your Swarm installation, where the folder name matches the module's name, and that folder must contain (at a minimum) a `Module.php` file. The file `Module.php` describes the dependencies, namespace, events subscriptions, and otherwise how the module integrates with Swarm.

This chapter provides only cursory coverage of how modules integrate with Swarm. Refer to the included `Jira` module for a good example of a simple module implementation within Swarm:

```
swarm_install/module/Jira
```

Influence activity events, emails, etc.

When something occurs in Helix Core (change submitted, files shelved, job added/edited), or state changes within Swarm (comment added, review state changed, etc.), the event is pushed onto a task queue. A background worker process subsequently pulls events off of the queue and publishes an event alerting modules about activity they may be interested in processing. This architecture allows the Swarm user interface to be fairly quick while accommodating tasks that might require notable processing time, or need to wait for related information to become available.

Subscribers to the worker event flesh the item out (fetch the change/job details, for example) and indicate if it should result in an activity entry, email notification, etc. By subscribing to various event topics, your module can pay attention to specific types of events. While your module is processing an event, it can modify the text of activity events, change the contents of emails, drop things entirely from activity, etc.

When your module subscribes to an event, set the priority to influence how early or late in the process it runs. You will likely want your module to run after most other modules have done their work to flesh out the event, but before Swarm's activity module processes it. The activity module sets a good example of subscribing to these events:

```
swarm_install/module/Activity/Module.php
```

Note that its priority is set to -100. Select a value before that for your own module (for example, 0 would be neutral and -90 would indicate that you are interested in being last).

The activity module listens to all events. However, be more selective. For example, if you are only interested in changes, subscribe to `task.change` instead of `*`. Current task types are:

- `task.change`
- `task.shelve`
- `task.review`
- `task.comment`
- `task.job`

Templates

Override existing view templates using your custom module. Have a look at an example module that demonstrates how to customize the email templates Swarm uses for comment notifications.

For more information about views, see the Zend/View Quick Start.

View helpers

Set options on existing helpers

It is possible to influence the behavior of existing view helpers by setting options on them; for an example see: *swarm_install*`/module/Application/Module.php`

Register new helpers

It is also possible to register new view helpers by placing them within your module's hierarchy, for example, `MyModule/src/MyModule/View/Helper/Foo.php`. Use the following Swarm view helper for inspiration: ***swarm_install***`/module/Activity/src/Activity/View/Helper/Activity.php`

Then register your view helper with the view manager via your `ModuleConfig`: ***swarm_root***`/module/Activity/config/module.config.php`

Example linkify module

The following example module demonstrates how to turn text that appears in changelist, job, or code review descriptions, comments, and activity entries into links.

1. Create a folder called `Rickroll` in the module folder.

2. Create the file `Module.php` within `module/Rickroll` and edit it to contain:

```
<?php
/**
 * Helix Swarm
 *
 * @copyright   2014 Perforce Software. All rights reserved.
 * @license     Please see LICENSE.txt in top-level folder of this
distribution.
 * @version     <release>/<patch>
 */

 namespace Rickroll;

 use Application\Filter\Linkify;

 class Module
 {
   public function onBootstrap()
   {
     Linkify::addCallback(
       function ($value, $escaper) {
         if (strcasecmp($value, 'rickroll')) {
           // not a hit; tell caller we did not handle this one
           return false;
         }

         return '<a target="_new"
href="https://www.youtube.com/watch?v=dQw4w9WgXcQ">'
                . $escaper->escapeHtml($value) . "</a>";
       },
       'rickroll',
       strlen('rickroll')
     );
   }

   public function getConfig()
```

```
    {
        return array();
    }
}
```

This file achieves several things. It:

- makes the `Rickroll` folder a recognized module.
- declares the module's namespace, which matches the module's folder name `Rickroll`.
- provides an `onBootstrap()` method that allows the module's configuration to be established immediately after the module is loaded
- adds a callback to the Linkify module that declares what text is to be searched for (`rickroll`) and, when found, how to compose the link for that text.

Example email module

The following example module demonstrates how to customize the email template Swarm uses when sending notifications for comments.

1. Create a folder called Example in the module folder.

2. Create the file `Module.php` within `module/Example` and edit it to contain:

```
<?php

namespace Example;
use Zend\Mvc\MvcEvent;

/**
 * Automatically uses any custom email templates found under this
 * module's view/mail folder (e.g. Example/view/mail/commit-
html.phtml).
 *
 * Valid templates include:
 *
 *   commit-html.phtml (HTML version of commit notification)
 *   commit-text.phtml (text version of commit notification)
 *  comment-html.phtml (HTML version of comment notification)
 *  comment-text.phtml (text version of comment notification)
 *   review-html.phtml (HTML version of review notification)
 *   review-text.phtml (text version of review notification)
 *
 * Note: you need to provide custom templates for both HTML and text;
 * if you do not provide both, it is possible that the search for
 * customized templates only finds the non-customized versions,
making
 * it appear that this module is not working.
 */
class Module
{
    public function onBootstrap(MvcEvent $event)
    {
        $application = $event->getApplication();
        $services    = $application->getServiceManager();
        $events      = $services->get('queue')->getEventManager();

         $events->attach(
            '*',
            function ($event) {
                mail = $event->getParam('mail');
                if (!$mail || !isset($mail['htmlTemplate'], $mail
['textTemplate'])) {
                    return;
                }

                $html = __DIR__ . '/view/mail/' . basename($mail
['htmlTemplate']);
                $text = __DIR__ . '/view/mail/' . basename($mail
['textTemplate']);

                if (file_exists($html)) {
                    $mail['htmlTemplate'] = $html;
                }
                if (file_exists($text)) {
                    $mail['textTemplate'] = $text;
                }

                $event->setParam('mail', $mail);
            },
            -199
        );
    }
}
```

This file achieves several things. It:

- makes the `Example` folder a recognized module.
- declares the module's namespace, which matches the module's folder name `Example`.
- provides an `onBootstrap()` method that allows the module's configuration to be established immediately after the module is loaded
- attaches to events, looking for `mail` events. When such an event is encountered, it provides local paths for HTML and text-only view scripts.

- declares an event priority of `-199`. Because email delivery events are processed with a priority of `-200`, this module's templates should override any that may have been set elsewhere, and this occurs just prior to email delivery.

3. Create a folder called `view` in the `module/Example` folder.
4. Create a folder called `mail` in the `module/Example/view` folder.

5. Create the file `comment-html.phtml` within `module/Example/view/mail` and edit it to contain:

```
<?php
    $user       = $activity->get('user');
    $userLink   = $user
                ? $this->qualifiedUrl('user', array('user' => $user))
                : null;
    $targetLink = $activity->getUrl($this->plugin('qualifiedUrl'));
?>
<html>
  <body style="font-family: sans-serif; background-color: #eee;
padding: 1em;">
    <div style="background-color: #fff; border: 1px solid #ccc;
padding: 1em;">
      <div style="font-size: 115%;">
        <?php if ($user): ?>
          <a style="text-decoration: none;" href="<?php echo
$userLink ?>">
            <?php echo $this->escapeHtml($user) ?>
          </a>
        <?php endif; ?>
        <?php echo $this->escapeHtml($activity->get('action')) ?>
        <a style="text-decoration: none;" href="<?php echo
$targetLink ?>">
          <?php echo $this->escapeHtml($activity->get('target'))?>
        </a>
      </div>
      <br/>
      <?php
          // if the comment has file context, show it.
          $comment = $event->getParam('comment');
          $context = $comment
                  ? $comment->getFileContext()
                  : array('content' => null, 'line' => null);
          if (is_array($context['content']) && $context['line']) {
             $line = $context['line'] - count($context['content']) +
1;
             echo '<div style="font-family: monospace; white-space:
nowrap;'
                . ' padding: .5em 1em; overflow-x: auto; color:
#444;'
                . ' border: 1px solid #ddd; background-color:
#f7f7f7;">';
             foreach ((array) $context['content'] as $i => $content)
{
                echo '<div><span style="color: #999;">'
                    . str_pad($line + $i,
                              strlen($context['line']),
                              "0",
                              STR_PAD_LEFT
                      )
                    . '.</span>&nbsp;'
                    . $this->preformat($content)
                           ->setLinkify(false)
                           ->setEmojify(false)
                           ->setWordWrap(900)
                    . "</div>\n";
            }
            echo '</div><br/>';
         }
      ?>
      <div style="padding-bottom: .5em;">
      <?php
          echo $this->preformat($activity->get('description'))
                  ->setBaseUrl($this->qualifiedUrl())
                  ->setEmojify(false)
                  ->setWordWrap(900)
      ?>
    </div>
  </div>
```

```
</body>
</html>
```

This is a view script that provides the content for the HTML portion of the comment notification email. Note that it is considered best practice to use inline CSS for styling emails.

6. Create the file c`omment-text.phtml` within `module/Example/view/mail` and edit it to contain:

```
<?php
    echo trim($activity->get('user')
        . ' commented on '
        . $activity->get('target'));

    // if the comment has file context, show it.
    $comment = $event->getParam('comment');
    $context = $comment
             ? $comment->getFileContext()
             : array('content' => null);
    if (is_array($context['content'])) {
        echo "\n\n> " . $this->wordWrap(
            implode("\n> ", $context['content']), 900
        );
    }

    echo "\n\n" . trim($this->wordWrap($activity->get('description'),
900));
    echo "\n\n" . $activity->getUrl($this->plugin('qualifiedUrl'));
?>
```

This is a view script that provides the content for the text-only portion of the comment notification email.

If you need to customize any other types of Swarm notification email messages, locate the view scripts (both HTML and text) and copy them into `module/Example/view/mail`, maintaining the existing filenames, then modify the new files as desired.

Note
If you do not copy both the HTML and text templates, it is possible for the search for customized templates to only find non-customized versions, making it appear that your module is not working.

CSS & JavaScript

Custom CSS and JavaScript files will be loaded automatically if you place them under either:

SWARM_ROOT`/public/custom/*.`*(css|js)*

SWARM_ROOT`/public/custom/sub-folder/*.`*(css|js)*

Note
Swarm only supports customizations placed directly within the ***SWARM_ROOT***`/public/custom` folder or one sub-folder down. Customizations are added after all standard CSS and JavaScript. If more than one custom file is present they are added in alphabetical order.

Prior to creating any customizations, ensure that the ***SWARM_ROOT***`/public/custom` folder exists; Swarm does not ship with or create this folder.

Sample Javascript extensions

The following are example Javascript customizations that you might wish to apply to your Swarm installation. Each example can be implemented separately. Ideally, you would apply the Javascript customizations in a single file to reduce the number of web requests required.

Warning
Coding errors in your custom JavaScript files could cause the Swarm UI to stop working. If this occurs, use your browser's development tools to identify which file contains the problem, and move that file out of the *SWARM_ROOT*`/public/custom` folder. When the problem has been resolved, the file can be returned.

Make the *Created* column on the Reviews page clickable

```
$(document).on( 'click', '.reviews-table td.created', function() {
    var change = $(this).closest('tr').data('id');
    window.location = '/reviews/' + change;
});
```

Save these lines in a file, perhaps `reviews-created-clickable.js`, within the ***SWARM_ROOT***`/public/custom` folder. Swarm automatically includes the JavaScript file, making the entries in the *Created* column clickable immediately for all for subsequent review page views.

Adjust the number of "more context" lines for diffs

```
swarm.diff.moreContextLines = 25;
```

Replace the *25* with the number of lines of context that should be retrieved each time the *more context* bar is clicked when viewing a diff.

Save this line in a file, perhaps `more-context-lines.js`, within the ***SWARM_ROOT*/public/custom** folder. Swarm automatically includes the JavaScript file, adjusting the number of *more context* lines immediately for subsequent page views.

Note
Try to avoid making this value arbitrarily large, as the file being diffed could be very large; users wouldn't expect to see the entire file when clicking the *more context* bar. If you need to see the whole file, click the **Show full context** icon.

Customize the review state options text

```
var original = swarm.review.buildStateMenu;
swarm.review.buildStateMenu = function(){
    original();
    var needsReview = $('.icon-review-needsReview').parent();
    needsReview.html(needsReview.html().replace(
        'Needs Review', 'You need to review'
    ));
}
```

Replace the *You need to review* with the text you'd prefer to see instead of `Needs Review`.

Save this line in a file, perhaps `customize-review-states`.js, within the ***SWARM_ROOT*/public/custom**. Swarm automatically includes the JavaScript file, adjusting the text of the review states immediately for subsequent page views.

Sample CSS customizations

The following are example CSS customizations that you might wish to apply to your Swarm installation. Each example can be implemented separately. Ideally, you would apply the CSS customizations in a single file to reduce the number of web requests required.

Adjust the default tab size

```
body {
   tab-size: 4;
}
```

Replace the ***4*** with the tab size you prefer.

Save these lines in a file, perhaps `tab-size.css`, within the ***SWARM_ROOT*/public/custom** folder. Swarm automatically includes the CSS file, adjusting the tab size immediately for subsequent page views.

Apply a custom background to the login screen

```
body.route-login {
    background: url("/custom/login_background.jpg") no-repeat center
fixed;
}
```

Replace the ***/custom/login_background.jpg*** URL fragment with image you want to use. If you do not specify a full URL, the image you specify must exist within the ***SWARM_ROOT*/public** folder, preferably within the ***SWARM_ROOT*/public/custom** folder.

Save these lines in a file, perhaps **login-background.css**, within the ***SWARM_ ROOT*/public/custom** folder. Swarm automatically includes the CSS file, immediately replacing the login screen's background for subsequent page views.

Replace Swarm's logo in the main navigation bar

```
.navbar-site .brand {
    background: url("/custom/navbar_logo.jpg") no-repeat center;
}
```

Replace the ***/custom/navbar_logo.jpg*** URL fragment with image you want to use. If you do not specify a full URL, the image you specify must exist within **the *SWARM_ROOT*/public** folder, preferably within the ***SWARM_ROOT*/public/custom** folder.

Save these lines in a file, perhaps **navbar-logo.css**, within the ***SWARM_ ROOT*/public/custom** folder. Swarm automatically includes the CSS file, immediately replacing the login screen's background for subsequent page views.

Note
Swarm's navbar design supports logos up to 24 pixels tall. Even if your logo fits within that height, you may need to also adjust the width, height, margins, or padding to suit your logo.

Adjust the appearance of avatars

```
img.avatar {
    border: 2px dashed red;
    border-radius: 10%;
}
```

You can make a number of adjustments to the way Swarm presents avatars, such as adding a border and adjusting the border radius, as the example above demonstrates. You should avoid attempting to set specific sizes because Swarm uses different sizes depending on where the avatar is displayed.

Save these lines in a file, perhaps `avatars.css`, within the ***SWARM_ROOT*/`public/custom`** folder. Swarm automatically includes the CSS file, immediately replacing the login screen's background for subsequent page views.

Swarm API

This chapter describes the REST-like API provided by Swarm, which can be used to automate common Swarm interactions or integrate with external systems.

Authentication

Swarm's API requires an authenticated connection for all data-modifying endpoints. Authenticated connections are achieved using HTTP Basic Access Authentication.

Note

If the `require_login` configuration flag is set to `true`, all API endpoints require authentication.

For example:

```
$ curl -u "apiuser:password" https://myswarm.url/api/v7/projects
```

Swarm accepts a ticket from the Helix Versioning Engine (previous versions of Swarm required this ticket to be host-unlocked, this is no longer true since Swarm 2017.1). It may also be possible to use a password in place of the ticket.

To acquire a ticket, run the following command:

```
$ p4 -p myp4host:1666 -u apiuser login -p
```

Important

For a Helix Versioning Engine that has been configured for security level 3, passwords are not accepted.

For more information on security levels, see:
Helix Versioning Engine Administrator Guide: Fundamentals

Note

If you use a ticket to authenticate against the Swarm API and the ticket expires, you need to acquire a new ticket to continue using the API.

If you make a request that requires authentication and you have not authenticated, the response is:

```
{
  "error": "Unauthorized"
}
```

Requests

Swarm's API includes endpoints that provide, create, and update information within Swarm.

If you make a request against an endpoint that is not supported, the response is:

```
{
  "error": "Method Not Allowed"
}
```

GET information

Use HTTP <literal>GET</literal> requests to ask for information from the API.

For example, to get the list of reviews:

```
$ curl https://myswarm.url/api/v7/reviews
```

Certain API calls support a `fields` parameter that allows you to specify which fields to include in the response, enabling more compact data sets. The following endpoints support fields:

- `/api/v7/projects`
- `/api/v7/reviews`
- `/api/v7/reviews/{id}`

Fields can be expressed as a comma-separated list, or using array-notation. For example:

```
$ curl
'https://
myswarm.url/api/v7/reviews?fields=id,description,participants'
```

Or:

```
$ curl 'https://myswarm.url/api/v7/reviews?fields[]=id,fields
[]=description,fields[]=participants'
```

POST new information

Use HTTP <literal>POST</literal> requests to create information via the API.

For example, to create a review using form-encoded values:

```
$ curl -u "apiuser:password" -d"change=12345"
https://myswarm.url/api/v7/reviews
```

The response should be similar to:

```
{
  "isValid": true,
  "id": 12206
}
```

To create a review using JSON:

```
$ curl -u "apiuser:password" -H "Content-type: application/json" \
  -d'{"change": 12345}' https://myswarm.url/api/v7/reviews
```

Update

Use HTTP <literal>PATCH</literal> requests to update information via the API.

If your HTTP client does not support `PATCH` requests, you can emulate this behavior by submitting an HTTP `POST` with a `"?_method=PATCH"` parameter.

Pagination

Most Swarm endpoints that provide data include the ability to paginate their results.

Each time data is requested, up to `max` results are included in the response, as is a value called `lastSeen`. `lastSeen` identifies the `id` of the last entry included in the results. If there are no further results, `lastSeen` is `null`.

To get the next set of results, include `after` set to the value of `lastSeen` in the API request. Entries up to and including the `id` specified by `after` are excluded from the response, and the next `max` entries are included.

See the Activity endpoint for example usage that demonstrates pagination.

Responses

Swarm's API responses are JSON formatted.

API versions

The current Swarm API version is `v7`. Here is a list of historical API versions:

API version	Swarm Release	Date	Description
v7	2017.3	October 2017	Include support for groups as participants of a review and groups as moderators of a project branch.

API version	Swarm Release	Date	Description
v6	2017.1	May 2017	Include support for activity dashboard, archiving of inactive reviews, cleaning completed reviews and for voting reviews up and down.
v5	2016.3	December 2016	Include support for limiting comments to a specific review version.
v4	2016.2	October 2016	Include support for private projects, as well as file-level and line-level inline comments.
v3	2016.1 SP1	September 2016	Include new endpoint for comments.
v2	2016.1	May 2016	Include new endpoints for projects, groups, etc.
v1.2	2015.3	October 2015	Add author filter to the list reviews endpoint.
v1.1	2014.4	January 2015	Addition of required reviewers, and `apiVersions`.
v1	2014.3	July 2014	Initial release.

API Endpoints

This section includes coverage for each of the major endpoints provided by the API.

Activity : Swarm Activity List

GET /api/v7/activity

Summary: List Activity Entries

Description

Retrieve the Activity List.

Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
`change`	Optionally filter activity entries by associated Changelist ID. This only includes records for which there is an activity entry in Swarm.	integer	form	No	
`stream`	Optional activity stream to query for entries. This can include user-initiated actions (`user-alice`), activity relating to a user's followed projects/users (`personal-alice`), review streams (`review-1234`), and project streams (`project-exampleproject`).	string	form	No	
`type`	Type of activity, e.g., `change`, `comment`, `job`, or `review`.	string	form	No	
`after`	An activity ID to seek to. Activity entries up to and including the specified ID are excluded from the results and do not count towards `max`. Useful for pagination. Commonly set to the `lastSeen` property from a previous query.	integer	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
`max`	Maximum number of activity entries to return. This does not guarantee that `max` entries are returned. It does guarantee that the number of entries returned won't exceed `max`. Server-side filtering may exclude some activity entries for permissions reasons.	integer	query	No	100
`fields`	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No	

Successful Response:

```
HTTP/1.1 200 OK

{
  "activity": [
    {
      "id": 123,
      "action": "committed",
      "behalfOf": null,
      "behalfOfExists": false,
      "change": 1,
      "date": "2016-01-15T12:12:12-08:00",
      "depotFile": null,
      "description": "test\n",
      "details": [],
      "followers": [],
      "link": ["change", {"change": 1}],
      "preposition": "into",
      "projectList": {"restricted": ["main"]},
      "projects": {"restricted": ["main"]},
      "streams": ["review-2", "user-foo", "personal-foo", "project-
restricted"],
      "target": "change 1",
      "time": 1404776681,
      "topic": "changes/1",
      "type": "change",
      "url": "/changes/1",
      "user": "bruno",
      "userExists": true
    }
  ],
  "lastSeen": 1
}
```

Examples of usage

Fetching review history

To get the latest activity entries on a review:

```
curl -u "username:password"
"https://myswarm.url/api/v7/activity?stream=review-1234\
&fields=id,date,description,type\
&max=2"
```

You can tweak `max` and `fields` to fetch the data that works best for you.

Swarm responds with an array of activity entities, and a `lastSeen` value that can be used for pagination:

```
{
  "activity": [
    {
      "id": 10,
      "date": "2016-04-15T16:10:32-07:00",
      "description": "This is a test comment.",
      "type": "comment"
    },
    {
      "id": 9,
      "date": "2016-03-31T13:48:15-07:00",
      "description": "Updating RELNOTE review",
      "type": "review"
    }
  ],
  "lastSeen": 9
}
```

Activity pagination

To get the second page of activity entries for a review (based on the previous example):

```
curl -u "username:password"
"https://myswarm.url/api/v7/activity?stream=review-1234\
&fields=id,date,description,type\
&max=2\
&lastSeen=9"
```

Swarm again responds with a list of activity entities and a `lastSeen` value:

```
{
  "activity": [
    {
      "id": 8,
      "date": "2016-03-30T12:12:12-07:00",
      "description": "This is the first test comment.",
      "type": "comment"
    },
    {
      "id": 7,
      "date": "2016-03-29T12:13:14-07:00",
      "description": "Updating RELNOTE review",
      "type": "review"
    }
  ],
  "lastSeen": 7
}
```

POST /api/v7/activity

Summary: Create Activity Entry

Description

Creates an entry in the Activity List. Note: admin-level privileges are required for this action.

Parameters

Parameter	Description	Type	Parameter Type	Required
`type`	Type of activity, used for filtering activity streams (values can include `change`, `comment`, `job`, `review`).	string	form	Yes
`user`	User who performed the action.	string	form	Yes
`action`	Action that was performed - past-tense, e.g., `created` or `commented on`.	string	form	Yes
`target`	Target that the action was performed on, e.g., `issue 1234`.	string	form	Yes
`topic`	Optional topic for the activity entry. Topics are essentially comment thread IDs. Examples: `reviews/1234` or `jobs/job001234`.	string	form	No
`description`	Optional description of object or activity to provide context.	string	form	No
`change`	Optional changelist ID this activity is related to. Used to filter activity related to restricted changes.	integer	form	No

Parameter	Description	Type	Parameter Type	Required
`streams[]`	Optional array of streams to display on. This can include user-initiated actions (`user-alice`), activity relating to a user's followed projects/users (`personal-alice`), review streams (`review-1234`), and project streams (`project-exampleproject`).	array (of strings)	form	No
`link`	Optional URL for `target`.	string	form	No

Successful Response:

```
HTTP/1.1 200 OK

{
  "activity": {
    "id": 123,
    "action": "ate",
    "behalfOf": null,
    "change": null,
    "depotFile": null,
    "details": [],
    "description": "",
    "followers": [],
    "link": "",
    "preposition": "for",
    "projects": [],
    "streams": [],
    "target": "the manual",
    "time": 1404776681,
    "topic": "",
    "type": "comment",
    "user": "A dingo"
  }
}
```

Examples of usage

Creating an activity entry

To create a plain activity entry:

```
curl -u "username:password" -d "type=job" -d "user=jira" -d
"action=punted" -d "target=review 123" \
     "https://myswarm.url/api/v7/activity"
```

JSON Response:

```
{
  "activity": {
    "id": 1375,
    "action": "punted",
    "behalfOf": null,
    "change": null,
    "depotFile": null,
    "description": "",
    "details": [],
    "followers": [],
    "link": "",
    "preposition": "for",
    "projects": [],
    "streams": [],
    "target": "review 123",
    "time": 1461607739,
    "topic": "",
    "type": "job",
    "user": "jira"
  }
}
```

Linking an activity entry to a review

Linking activity entries to reviews is useful. This involves providing `link`, `stream`, and `topic` fields in the activity data. The `link` field is used to make the `review 123` string in the activity entry clickable. The `stream` field is needed so that the activity entry can be attached to the review in the Swarm interface. The `topic` field is used to link the activity entry to the comment thread for that topic, in the event that a user wants to comment on the activity.

To create a fully linked activity entry:

```
curl -u "username:password" -d "type=job" -d "user=jira" -d
"action=punted" -d "target=review 123" \
     -d "streams[]=review-123" \
     -d "link=reviews/123" \
     -d "topic=reviews/123" \
     "https://myswarm.url/api/v7/activity"
```

Swarm responds with an activity entity:

```
{
  "activity": {
    "id": 1375,
    "action": "punted",
    "behalfOf": null,
    "change": null,
    "depotFile": null,
    "description": "",
    "details": [],
    "followers": [],
    "link": "reviews/123",
    "preposition": "for",
    "projects": [],
    "streams": ["review-123"],
    "target": "review 123",
    "time": 1461607739,
    "topic": "reviews/123",
    "type": "job",
    "user": "jira"
  }
}
```

Comments : Swarm Comments

GET /api/v7/comments/

Summary: Get List of Comments

Description

List comments.

Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
`after`	A comment ID to seek to. Comments up to and including the specified ID are excluded from the results and do not count towards `max`. Useful for pagination. Commonly set to the `lastSeen` property from a previous query.	integer	query	No	
`max`	Maximum number of comments to return. This does not guarantee that `max` comments are returned. It does guarantee that the number of comments returned won't exceed `max`.	integer	query	No	100

Parameter	Description	Type	Parameter Type	Required	Default Value
`topic`	Only comments for given topic are returned. Examples: `reviews/1234`, `changes/1234` or `jobs/job001234`.	string	query	No	
`context[version]`	If a `reviews/1234` topic is provided, limit returned comments to a specific version of the provided review.	integer	query	No	
`ignoreArchived`	Prevents archived comments from being returned. (v5+)	boolean	query	No	
`tasksOnly`	Returns only comments that have been flagged as tasks. (v5+)	boolean	query	No	
`taskStates`	Limit the returned comments to ones that match the provided task state (one or more of `open`, `closed`, `verified`, or `comment`). (v5+)	array (of strings)	query	No	
`fields`	An optional comma-separated list (or array) of fields to show for each comment. Omitting this parameter or passing an empty value shows all fields.	string	query	No	

Examples of successful responses

Successful Response:

```
HTTP/1.1 200 OK

{
  "topic": "",
  "comments": {
    "51": {
      "id": 51,
      "attachments": [],
      "body": "Short loin ground round sin reprehensible, venison west
participle triple.",
      "context": [],
      "edited": null,
      "flags": [],
      "likes": [],
      "taskState": "comment",
      "time": 1461164347,
      "topic": "reviews/885",
      "updated": 1461164347,
      "user": "bruno"
    }
  },
  "lastSeen": 51
}
```

Note
`lastSeen` can often be used as an offset for pagination, by using the value in the `after` parameter of subsequent requests.

When no results are found, the `comments` array is empty:

```
HTTP/1.1 200 OK

{
  "topic": "jobs/job000011",
  "comments": [],
  "lastSeen": null
}
```

Examples of usage

Listing comments

To list comments:

```
curl -u "username:password" "https://my-swarm-host/api/v7/comments\
?topic=reviews/911&max=2&fields=id,body,time,user"
```

Swarm responds with a list of the first two comments for review 911 and a `lastSeen` value for pagination:

```
{
  "topic": "reviews/911",
  "comments": {
    "35": {
      "id": 35,
      "body": "Excitation thunder cats intelligent man braid organic
bitters.",
      "time": 1461164347,
      "user": "bruno"
    },
    "39": {
      "id": 39,
      "body": "Chamber tote bag butcher, shirk truffle mode shabby chic
single-origin coffee.",
      "time": 1461164347,
      "user": "swarm_user"
    }
  },
  "lastSeen": 39
}
```

Paginating a comment listing

To obtain the next page of a comments list (based on the previous example):

```
curl -u "username:password" "https://my-swarm-host/api/v7/comments\
?topic=reviews/911&max=2&fields=id,body,time,user&after=39"
```

Swarm responds with the second page of results, if any comments are present after the last seen comment:

```
{
  "topic": "reviews/911",
  "comments": {
    "260": {
      "id": 260,
      "body": "Reprehensible do lore flank ham hock.",
      "time": 1461164349,
      "user": "bruno"
    },
    "324": {
      "id": 324,
      "body": "Sinter lo-fi temporary, nihilist tote bag mustache swag
consequence interest flexible.",
      "time": 1461164349,
      "user": "bruno"
    }
  },
  "lastSeen": 324
}
```

POST /api/v7/comments/

Summary: Add A Comment

Description

Add a comment to a topic (such as a review or a job)

Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
`topic`	Topic to comment on. Examples: `reviews/1234`, `changes/1234` or `jobs/job001234`.	string	form	Yes	
`body`	Content of the comment.	string	form	Yes	
`taskState`	Optional task state of the comment. Valid values when adding a comment are `comment` and `open`. This creates a plain comment or opens a task, respectively.	string	form	No	comment
`flags[]`	Optional flags on the comment. Typically set to `closed` to archive a comment.	array (of strings)	form	No	
`context[file]`	File to comment on. Valid only for `changes` and `reviews` topics. Example: `//depot/main/README.txt`.	string	form	No	
`context[leftLine]`	Left-side diff line to attach the inline comment to. Valid only for `changes` and `reviews` topics. If this is specified, `context[file]` must also be specified.	integer	form	No	
`context[rightLine]`	Right-side diff line to attach the inline comment to. Valid only for `changes` and `reviews` topics. If this is specified, `context[file]` must also be specified.	integer	form	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
`context [content]`	Optionally provide content of the specified line and its four preceding lines. This is used to specify a short excerpt of context in case the lines being commented on change during the review. When not provided, Swarm makes an effort to build the content on its own - as this involves file operations, it could become slow.	array (of strings)	form	No	
`context [version]`	With a `reviews` topic, this field specifies which version to attach the comment to.	integer	form	No	

Successful Response contains Comment entity:

```
HTTP/1.1 200 OK

{
  "comment": {
    "id": 42,
    "attachments": [],
    "body": "Best. Comment. EVER!",
    "context": [],
    "edited": null,
    "flags": [],
    "likes": [],
    "taskState": "comment",
    "time": 123456789,
    "topic": "reviews/2",
    "updated": 123456790,
    "user": "bruno"
  }
}
```

Examples of usage

Create a comment on a review

To create a comment on a review:

```
curl -u "username:password" \
     -d "topic=reviews/2" \
     -d "body=This is my comment. It is an excellent comment. It contains
a beginning, a middle, and an end." \
     "https://my-swarm-host/api/v7/comments"
```

JSON Response:

```
{
  "comment": {
    "id": 42,
    "attachments": [],
    "body": "This is my comment. It is an excellent comment. It contains a
beginning, a middle, and an end.",
    "context": [],
    "edited": null,
    "flags": [],
    "likes": [],
    "taskState": "comment",
    "time": 123456789,
    "topic": "reviews/2",
    "updated": 123456790,
    "user": "username"
  }
}
```

Open a task on a review

To create a comment on a review, and flag it as an open task:

```
curl -u "username:password" \
     -d "topic=reviews/2" \
     -d "taskState=open" \
     -d "body=If you could go ahead and attach a cover page to your TPS
report, that would be great." \
     "https://my-swarm-host/api/v7/comments"
```

JSON Response:

```
{
  "comment": {
    "id": 43,
    "attachments": [],
    "body": "If you could go ahead and attach a cover page to your TPS
report, that would be great.",
    "context": [],
    "edited": null,
    "flags": [],
    "likes": [],
    "taskState": "open",
    "time": 123456789,
    "topic": "reviews/2",
    "updated": 123456790,
    "user": "username"
  }
}
```

PATCH /api/v7/comments/{id}

Summary: Edit A Comment

Description

Edit a comment

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	ID of the comment to be edited	integer	path	Yes
`topic`	Topic to comment on. Examples: `reviews/1234`, `changes/1234` or `jobs/job001234`.	string	form	No

Parameter	Description	Type	Parameter Type	Required
`body`	Content of the comment.	string	form	Yes
`taskState`	Optional task state of the comment. Note that certain transitions (such as moving from `open` to `verified`) are not possible without an intermediate step (`addressed`, in this case). Examples: `comment` (not a task), `open`, `addressed`, `verified`.	string	form	No
`flags[]`	Optional flags on the comment. Typically set to `closed` to archive a comment.	array (of strings)	form	No

Successful Response contains Comment entity:

```
HTTP/1.1 200 OK

{
  "comment": {
    "id": 1,
    "attachments": [],
    "body": "Best. Comment. EVER!",
    "context": [],
    "edited": 123466790,
    "flags": [],
    "likes": [],
    "taskState": "comment",
    "time": 123456789,
    "topic": "reviews/42",
    "updated": 123456790,
    "user": "bruno"
  }
}
```

Examples of usage

Edit and archive a comment on a review

To edit and archive a comment on a review:

```
curl -u "username:password" \
     -X PATCH \
     -d "flags[]=closed" \
     -d "body=This comment wasn't as excellent as I may have lead you to
believe. A thousand apologies." \
     "https://my-swarm-host/api/v7/comments/42"
```

JSON Response:

```
{
  "comment": {
    "id": 42,
    "attachments": [],
    "body": "This comment wasn't as excellent as I may have lead you to
believe. A thousand apologies.",
    "context": [],
    "edited": 123466790,
    "flags": ["closed"],
    "likes": [],
    "taskState": "comment",
    "time": 123456789,
    "topic": "reviews/2",
    "updated": 123456790,
    "user": "username"
  }
}
```

Flag a task as addressed on a review

To flag an open task as addressed on a review:

```
curl -u "username:password" \
     -X PATCH \
     -d "taskState=addressed" \
     "https://my-swarm-host/api/v7/comments/43"
```

JSON Response:

```
{
  "comment": {
    "id": 43,
    "attachments": [],
    "body": "If you could go ahead and attach a cover page to your TPS
report, that would be great.",
    "context": [],
    "edited": 123466790,
    "flags": ["closed"],
    "likes": [],
    "taskState": "comment",
    "time": 123456789,
    "topic": "reviews/2",
    "updated": 123456790,
    "user": "username"
  }
}
```

Groups : Swarm Groups

GET /api/v7/groups/

Summary: Get List of Groups

Description

Returns the complete list of groups in Swarm.

Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
`after`	A group ID to seek to. Groups prior to and including the specified ID are excluded from the results and do not count towards `max`. Useful for pagination. Commonly set to the `lastSeen` property from a previous query.	string	query	No	
`max`	Maximum number of groups to return. This does not guarantee that `max` groups are returned. It does guarantee that the number of groups returned won't exceed `max`.	integer	query	No	100
`fields`	An optional comma-separated list (or array) of fields to show for each group. Omitting this parameter or passing an empty value shows all fields.	string	query	No	
`keywords`	Keywords to limit groups on. Only groups where the group ID, group name (if set), or description contain the specified keywords are returned.	string	query	No	

Successful Response:

```
HTTP/1.1 200 OK

{
  "groups": [
    {
      "Group": "test-group",
      "MaxLockTime": null,
      "MaxResults": null,
      "MaxScanRows": null,
      "Owners": [],
      "PasswordTimeout": null,
      "Subgroups": [],
      "Timeout": 43200,
      "Users": ["bruno"],
      "config": {
        "description": "Our testing group",
        "emailAddress": "test-group3@host.domain",
        "emailFlags": [],
        "name": "Test Group",
        "useMailingList": true
      }
    }
  ]
}
```

Examples of usage

Listing groups

To list groups:

```
curl -u "username:password" \
     "https://myswarm.url/api/v7/groups?keywords=test-
group&fields=Group,Owners,Users,config&max=2"
```

Swarm responds with a list of groups:

```
{
  "groups": [
    {
      "Group": "test-group",
      "Owners": [],
      "Users": ["bruno"],
      "config": {
        "description": "Our testing group",
        "emailAddress": "test-group@host.domain",
        "emailFlags": {
          "reviews": "1",
          "commits": "0"
        },
        "name": "Test Group",
        "useMailingList: true
      }
    },
    {
      "Group": "test-group2",
      "Owners": [],
      "Users": ["bruno"],
      "config": {
        "description": "Our second testing group",
        "emailAddress": "test-group2@host.domain",
        "emailFlags": [],
        "name": "Test Group 2",
        "useMailingList: true
      }
    }
  ],
  "lastSeen": "test-group2"
}
```

Paginating the groups list

Based on the previous example, we can pass a lastSeen value of `test-group2` to see if there are any subsequent groups in Swarm.

```
curl -u "username:password" \
     "https://myswarm.url/api/v7/groups?keywords=test-
group&fields=Group,config&max=2&lastSeen=test-group2"
```

Swarm responds with a list of groups (minus the Owners and Users fields, as we haven't requested them):

```
{
  "groups": [
    {
      "Group": "test-group3",
      "config": {
        "description": "Our 3rd testing group",
        "emailAddress": "test-group3@host.domain",
        "emailFlags": [],
        "name": "Test Group 3",
        "useMailingList": true
      }
    }
  ],
  "lastSeen": "test-group3"
}
```

GET /api/v7/groups/{id}

Summary: Get Group Information

Description

Retrieve information about a group.

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Group ID	string	path	Yes
`fields`	An optional comma-separated list (or array) of fields to show for each group. Omitting this parameter or passing an empty value shows all fields.	string	query	No

Successful Response:

```
HTTP/1.1 200 OK

{
  "group": {
    "Group": "test-group",
    "MaxLockTime": null,
    "MaxResults": null,
    "MaxScanRows": null,
    "Owners": [],
    "PasswordTimeout": null,
    "Subgroups": [],
    "Timeout": 43200,
    "Users": ["bruno"],
    "config": {
      "description": "Our testing group",
      "emailFlags": [],
      "name": "Test Group"
    }
  }
}
```

Examples of usage

Fetching a group

To fetch an individual group:

```
curl -u "username:password" "https://myswarm.url/api/v7/groups/my-group"
```

Swarm responds with the group entity:

```
{
  "group": {
    "Group": "test-group",
    "LdapConfig": null,
    "LdapSearchQuery": null,
    "LdapUserAttribute": null,
    "MaxLockTime": null,
    "MaxResults": null,
    "MaxScanRows": null,
    "Owners": [],
    "Users": ["bruno"],
    "config": {
      "description": "Our testing group",
      "emailAddress": "test-group@host.domain",
      "emailFlags": {
        "reviews": "1",
        "commits": "0"
      },
      "name": "Test Group",
      "useMailingList": true
    }
  }
}
```

Limiting returned fields

To limit the returned fields when fetching an individual group:

```
curl -u "username:password" "https://myswarm.url/api/v7/groups/my-
group?fields=Group,Owners,Users,config"
```

Swarm responds with the group entity:

```
{
  "group": {
    "Group": "test-group",
    "Owners": [],
    "Users": ["bruno"],
    "config": {
      "description": "Our testing group",
      "emailFlags": [],
      "name": "Test Group"
    }
  }
}
```

POST /api/v7/groups/

Summary: Create a new Group

Description

Creates a new group in Swarm.

Parameters

Parameter	Description	Type	Parameter Type	Required
`Group`	Group identifier string.	string	form	Yes
`Users`	An optional array of group users. At least one of Users, Owners, or Subgroups is required.	array	form	No
`Owners`	An optional array of group owners. At least one of Users, Owners, or Subgroups is required.	array	form	No
`Subgroups`	An optional array of subgroups. At least one of Users, Owners, or Subgroups is required.	array	form	No
`config[name]`	An optional full name for the group.	string	form	No
`config [description]`	An optional group description.	string	form	No
`config [emailAddress]`	The email address for this group.	string	form	No
`config [emailFlags] [reviews]`	Email members when a new review is requested.	boolean	form	No
`config [emailFlags] [reviews]`	Email members when a new review is requested.	boolean	form	No
`config [useMailingList]`	Whether to use the configured email address or expand individual members addresses.	boolean	form	No

Successful Response:

```
HTTP/1.1 200 OK
```

```
{
  "group": {
    "Group": "test-group",
    "MaxLockTime": null,
    "MaxResults": null,
    "MaxScanRows": null,
    "Owners": [],
    "PasswordTimeout": null,
    "Subgroups": [],
    "Timeout": null,
    "Users": ["alice"],
    "config": {
      "description": "Test test test",
      "emailAddress": "test-group@host.domain",
      "emailFlags": [],
      "name": "TestGroup",
      "useMailingList": true
    }
  }
}
```

Creating a group

Important

- Only users with *super* privileges in the Helix Versioning Engine (`p4d`), or users with *admin* privileges in `p4d` versions 2012.1 or newer, can create groups.
- This API version is only capable of setting specific fields: `Group`, `Users`, `Owners`, `Subgroups`, `config`. Any other fields specified in the creation request are ignored.

To create a new group:

```
curl -u "username:password" \
     -d "Group=my-group" \
     -d "Owners[]=alice" \
     -d "Owners[]=bob" \
     -d "Users[]=bruno" \
     -d "Users[]=user2" \
     -d "config[description]=This group is special to me." \
     -d "config[name]=My Group" \
     -d "config[emailFlags][reviews]=1" \
     -d "config[emailFlags][commits]=0" \
     -d "config[emailAddress]=my-group@host.domain" \
     -d "config[useMailingList]=false" \
     "https://myswarm.url/api/v7/groups"
```

Assuming that the authenticated user has permission, Swarm responds with the new group entity:

```
{
  "group": {
    "Group": "my-group",
    "MaxLockTime": null,
    "MaxResults": null,
    "MaxScanRows": null,
    "Owners": ["username"],
    "PasswordTimeout": null,
    "Subgroups": [],
    "Timeout": null,
    "Users": [],
    "config": {
      "description": "This group is special to me.",
      "emailFlags": {
        "reviews": "1",
        "commits": "0"
      },
      "name": "My Group",
      "useMailingList": true
    }
  }
}
```

PATCH /api/v7/groups/{id}

Summary: Edit a Group

Description

Change the settings of a group in Swarm. Only super users and group owners can perform this action.

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Group ID	string	path	Yes
`Users`	An optional array of group users.	array	form	No
`Owners`	An optional array of group owners.	array	form	No
`Subgroups`	An optional array of group subgroups.	array	form	No
`config[name]`	An optional full name for the group.	string	form	No
`config[description]`	An optional group description.	string	form	No
`config[emailAddress]`	The email address for this group.	string	form	No
`config[emailFlags][commits]`	Email members when a change is committed.	boolean	form	No
`config[emailFlags][reviews]`	Email members when a new review is requested.	boolean	form	No
`config[useMailingList]`	Whether to use the configured email address or expand individual members addresses.	boolean	form	No

Successful Response:

```
HTTP/1.1 200 OK

{
  "group": {
    "Group": "test-group",
    "Users": [],
    "Owners": [],
    "Subgroups": [],
    "config": {
      "description": "New Group Description",
      "name": "TestGroup",
      "useMailingList": true
    }
  }
}
```

Editing a group

Important

- Only users with *super* privileges in the Helix Versioning Engine, or group owners, can edit groups.
- This API version is only capable of modifying specific fields: `Users`, `Owners`, `Subgroups`, `config`. Any other fields specified in the edit request are ignored.

Here is how to update the `name`, `description`, and `emailFlags` configuration of the group `my-group`:

```
curl -u "username:password" -X PATCH \
     -d "config[description]=This group is special to me." \
     -d "config[name]=My Group" \
     -d "config[emailFlags][commits]=1" \
     "https://myswarm.url/api/v7/groups/my-group"
```

Assuming that the authenticated user has permission, Swarm responds with the modified group entity:

```
{
  "group": {
    "Group": "my-group",
    "Users": [],
    "Owners": [],
    "Subgroups": [],
    "config": {
      "description": "This group is special to me.",
      "emailAddress": "test-group@host.domain",
      "emailFlags": {
        "reviews": "1",
        "commits": "1"
      },
      "name": "My Group",
      "useMailingList": true
    }
  }
}
```

DELETE /api/v7/groups/{id}

Summary: Delete a Group

Description

Delete a group. Only super users and group owners can perform this action.

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Group ID.	string	path	Yes

Successful Response:

```
HTTP/1.1 200 OK

{
  "id": "test-group"
}
```

Deleting a group

Important
Only *super* users and group owners can delete groups.

```
curl -u "username:password" -X DELETE
```

```
"https://myswarm.url/api/v7/groups/my-group"
```

Assuming that the authenticated user has permission, Swarm responds with the `id` of the deleted group:

```
{
  "id": "my-group"
}
```

Index : Basic API controller providing a simple version action

GET /api/v7/version

Summary: Show Version Information

Description

This can be used to determine the currently-installed Swarm version, and also to check that Swarm's API is responding as expected.

Successful Response:

```
HTTP/1.1 200 OK

{
    "year": "2017",
    "version": "SWARM/2017.3-MAIN/8499605 (2017/10/25)"
}
```

Note

Note: `year` refers to the year of the Swarm release, not necessarily the current year.

Projects : Swarm Projects

GET /api/v7/projects/

Summary: Get List of Projects

Description

Returns a list of projects in Swarm that are visible to the current user. Administrators will see all projects, including private ones.

Parameters

Parameter	Description	Type	Parameter Type	Required
`fields`	An optional comma-separated list (or array) of fields to show for each project. Omitting this parameter or passing an empty value shows all fields.	string	query	No

Successful Response:

```
HTTP/1.1 200 OK

{
  "projects": [
    {
      "id": "testproject",
      "branches": [
        {
          "id": "main",
          "name": "main",
          "paths": ["//depot/main/TestProject/..."],
          "moderators": [],
          "moderators-groups": []
        }
      ],
      "deleted": false,
      "description": "Test test test",
      "followers": [],
      "jobview": "subsystem=testproject",
      "members": ["alice"],
      "name": "TestProject",
      "owners": [],
      "private": false,
      "subgroups": []
    }
  ]
}
```

Listing projects

To list all projects:

```
curl -u "username:password" "https://my-swarm-
host/api/v7/projects?fields=id,description,members,name"
```

Pagination is not currently supported by this endpoint. Swarm responds with a list of all projects:

```
{
  "projects": [
    {
      "id": "testproject",
      "description": "Test test test",
      "members": ["alice"],
      "name": "TestProject"
    },
    {
      "id": "testproject2",
      "description": "Test test test",
      "members": ["alice"],
      "name": "TestProject"
    }
  ]
}
```

Project administrators wishing to see the `tests` and `deploy` fields must fetch projects individually.

GET /api/v7/projects/{id}

Summary: Get Project Information

Description

Retrieve information about a project.

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Project ID	string	path	Yes
`fields`	An optional comma-separated list (or array) of fields to show for each project. Omitting this parameter or passing an empty value shows all fields.	string	query	No

Successful Response:

```
HTTP/1.1 200 OK

{
  "project": {
    "id": "testproject",
    "branches": [
      {
        "id": "main",
        "name": "main",
        "paths": ["//depot/main/TestProject/..."],
        "moderators": [],
        "moderators-groups": []
      }
    ],
    "deleted": false,
    "description": "Test test test",
    "jobview": "subsystem=testproject",
    "members": ["alice"],
    "name": "TestProject",
    "owners": [],
    "private": false,
    "subgroups": []
  }
}
```

Fetching a project

To fetch an individual project:

```
curl -u "username:password" \
     "https://my-swarm-
host/api/v7/projects/testproject2?fields=id,description,members,name"
```

Swarm responds with a project entity:

```
{
  "project": {
    "id": "testproject2",
    "description": "Test test test",
    "members": ["alice"],
    "name": "TestProject 2"
  }
}
```

Project administrators have access to additional fields (`tests` and `deploy`) when fetching individual projects using this endpoint.

POST /api/v7/projects/

Summary: Create a new Project

Description

Creates a new project in Swarm.

Parameters

Parameter	Description	Type	Parameter Type	Required
`name`	Project Name (is also used to generate the Project ID)	string	form	Yes
`members`	An array of project members.	array	form	Yes
`subgroups`	An optional array of project subgroups.	array	form	No
`owners`	An optional array of project owners.	array	form	No
`description`	An optional project description.	string	form	No

Parameter	Description	Type	Parameter Type	Required
`private`	Private projects are visible only to Members, Moderators, Owners, and Administrators. (Default: false)	boolean	form	No
`deploy`	Configuration for automated deployment. Example: {"enabled": true, "url": "http://localhost/?change={change}"}	array	form	No
`tests`	Configuration for testing/continuous integration.	array	form	No
`branches`	Optional branch definitions for this project.	array	form	No
`jobview`	An optional jobview for associating certain jobs with this project.	string	form	No
`emailFlags[change_email_project_users]`	Email members, moderators and followers when a change is committed.	boolean	form	No
`emailFlags[review_email_project_members]`	Email members and moderators when a new review is requested.	boolean	form	No

Successful Response:

```
HTTP/1.1 200 OK

{
  "project": {
    "id": "testproject",
    "branches": [
      {
        "id": "main",
        "name": "main",
        "paths": ["//depot/main/TestProject/..."],
        "moderators": [],
        "moderators-groups": []
      }
    ],
    "deleted": false,
    "deploy": {"url": "", "enabled": false},
    "description": "Test test test",
    "followers": [],
    "jobview": "subsystem=testproject",
    "members": ["alice"],
    "name": "TestProject",
    "owners": [],
    "private": false,
    "subgroups": [],
    "tests": {"url": "", "enabled": false}
  }
}
```

Examples of usage

Creating a new project

To create a project:

```
curl -u "username:password" \
     -d "name=TestProject 3" \
     -d "description=The third iteration of our test project." \
     -d "members[]=alice" \
     -d "members[]=bob" \
     "https://my-swarm-host/api/v7/projects/"
```

Swarm responds with the new project entity:

```
{
  "project": {
    "id": "testproject3",
    "branches": [],
    "deleted": false,
    "deploy": {"url": "", "enabled": false},
    "description": "The third iteration of our test project.",
    "followers": [],
    "jobview": "subsystem=testproject",
    "members": ["alice", "bob"],
    "name": "TestProject 3",
    "owners": [],
    "private": false,
    "subgroups": [],
    "tests": {"url": "", "enabled": false}
  }
}
```

Creating a private project with branches

Specifying a branch requires using array notation and providing at least two fields (`name` and `paths`) for each branch you wish to create. Creating more than one branch involves incrementing the `branches[0]` specifier for each branch - an example of this accompanies the PATCH endpoint documentation.

Projects are public by default. Marking a project as Private requires using `{private: true}` in JSON, and using `-d "private=1"` in regular form-encoded requests.

Important

Form-encoded requests only accept `0` for false in boolean values — using the word `false` will be evaluated as a non-zero (and therefore non-false) value.

```
curl -u "username:password" \
     -d "name=TestProject 4" \
     -d "private=1" \
     -d "members[]=bob" \
     -d "branches[0][name]=Branch One" \
     -d "branches[0][paths][]=//depot/main/TestProject/..." \
     "https://my-swarm-host/api/v7/projects"
```

Swarm responds with the new project entity:

```
{
  "project": {
    "id": "testproject-4",
    "branches": [
      {
        "paths": [
          "//depot/main/TestProject/..."
        ],
        "name": "Branch One",
        "id": "branch-one",
        "moderators": [],
        "moderators-groups": []
      }
    ],
    "deleted": false,
    "deploy": {"url": "", "enabled": false},
    "description": null,
    "emailFlags": [],
    "jobview": null,
    "members": ["bob"],
    "name": "TestProject 4",
    "owners": [],
    "private": true,
    "subgroups": [],
    "tests": {"url": "", "enabled": false}
  }
}
```

PATCH /api/v7/projects/{id}

Summary: Edit a Project

Description

Change the settings of a project in Swarm. If a project has owners set, only the owners can perform this action.

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Project ID	string	path	Yes
`name`	Project Name (changing the project name does not change the project ID)	string	form	No
`members`	An array of project members.	array	form	No
`subgroups`	An optional array of project subgroups.	array	form	No
`owners`	An optional array of project owners.	array	form	No
`description`	Your project description.	string	form	No
`private`	Private projects are visible only to Members, Moderators, Owners, and Administrators. (Default: false)	boolean	form	No
`deploy`	Configuration for automated deployment. Example: {"enabled": true, "url": "http://localhost/?change={change}"}	array	form	No
`tests`	Configuration for testing/continuous integration.	array	form	No
`branches`	Optional branch definitions for this project.	array	form	No
`jobview`	A jobview for associating certain jobs with this project.	string	form	No
`emailFlags[change_email_project_users]`	Email members, moderators and followers when a change is committed.	boolean	form	No
`emailFlags[review_email_project_members]`	Email members and moderators when a new review is requested.	boolean	form	No

Successful Response:

```
HTTP/1.1 200 OK

{
  "project": {
    "id": "testproject",
    "branches": [
      {
        "id": "main",
        "name": "main",
        "paths": ["//depot/main/TestProject/..."],
        "moderators": [],
        "moderators-groups": []
      }
    ],
    "deleted": false,
    "deploy": {"url": "", "enabled": false},
    "description": "New Project Description",
    "followers": [],
    "jobview": "subsystem=testproject",
    "members": ["alice"],
    "name": "TestProject",
    "owners": [],
    "private": false,
    "subgroups": [],
    "tests": {"url": "", "enabled": false}
  }
}
```

Examples of usage

Editing a project

To edit a project:

Note

It is safe to edit a project without specifying branches, but the instructions for adding branches contain important information for modifying branch configuration.

```
curl -u "username:password" \
     -X PATCH
     -d "description=Witness the power of a fully operational Swarm
project." \
     "https://my-swarm-host/api/v7/projects/testproject3"
```

Swarm responds with the updated project entity:

```
{
  "project": {
    "id": "testproject3",
    "branches": [],
    "deleted": false,
    "deploy": {"url": "", "enabled": false},
    "description": "Witness the power of a fully operational Swarm
project.",
    "followers": [],
    "jobview": "subsystem=testproject",
    "members": ["alice"],
    "name": "TestProject 3",
    "owners": [],
    "private": false,
    "subgroups": [],
    "tests": {"url": "", "enabled": false}
  }
}
```

Editing a project to add a moderated branch and make the project public

Specifying a branch requires using array notation and providing at least two fields (`name` and `paths`) for each branch you wish to create. Creating more than one branch involves incrementing the `branches[0]` specifier for each branch.

Important

If you have existing branches, you must specify all of them in the query to avoid data loss. This operation sets the value of the entire `branches` property to match the provided input.

Marking a private project as Public requires using `{private: false}` in JSON, or using `-d "private=0"` in regular form-encoded requests.

Important

Form-encoded requests only accept `0` for false in boolean values — using the word `false` will be evaluated as a non-zero (and therefore non-false) value.

```
curl -u "username:password" \
     -X PATCH \
     -d "private=0" \
     -d "branches[0][name]=Branch One" \
     -d "branches[0][paths][]=//depot/main/TestProject/..." \
     -d "branches[1][name]=Branch Two" \
     -d "branches[1][paths][]=//depot/main/SecondBranch/..." \
     -d "branches[1][moderators][]=bob" \
     -d "branches[1][moderators-groups][]=group1" \
     "https://my-swarm-host/api/v7/projects/testproject-4"
```

Swarm responds with the new project entity:

```
{
  "project": {
    "id": "testproject-4",
    "branches": [
      {
        "paths": [
          "//depot/main/TestProject/..."
        ],
        "name": "Branch One",
        "id": "branch-one",
        "moderators": [],
        "moderators-groups": []
      },
      {
        "paths": [
          "//depot/main/SecondBranch/..."
        ],
        "name": "Branch Two",
        "id": "branch-two",
        "moderators": ["bob"],
        "moderators-groups": ["group1"]
      },
    ],
    "deleted": false,
    "deploy": {"url": "", "enabled": false},
    "description": null,
    "emailFlags": [],
    "jobview": null,
    "members": ["bob"],
    "name": "TestProject 4",
    "owners": [],
    "private": false,
    "subgroups": [],
    "tests": {"url": "", "enabled": false}
  }
}
```

DELETE /api/v7/projects/{id}

Summary: Delete a Project

Description

Mark a Swarm project as deleted. The project ID and name cannot be reused. If a project has owners set, only the owners can perform this action.

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Project ID	string	path	Yes

Successful Response:

```
HTTP/1.1 200 OK

{
  "id": "testproject"
}
```

Deleting a project

Super users, administrators, and owners can delete projects. Members can delete projects that have no owners set.

```
curl -u "username:password" -X DELETE "https://my-swarm-
host/api/v7/projects/testproject3"
```

Assuming that the authenticated user has permission, Swarm responds with the id of the deleted project:

```
{
  "id": "testproject3"
}
```

Reviews : Swarm Reviews

GET /api/v7/dashboards/action

Summary: Get reviews for action dashboard

Description

Gets reviews for the action dashboard for the authenticated user

Successful Commit contains Review and Commit Entities:

```
HTTP/1.1 200 OK

{
  "lastSeen": 120,
  "reviews": [
    {
      "id": 7,
      "author": "swarm_admin",
      "changes": [6],
      "comments": [0,0],
      "commits": [6],
      "commitStatus": [],
      "created": 1485793976,
      "deployDetails": [],
      "deployStatus": null,
      "description": "test\n",
      "groups": ["swarm-project-test"],
      "participants": {"swarm_admin":[]},
      "pending": false,
      "projects": {"test":["test"]},
      "roles": ["moderator|reviewer|required_reviewer|author"],
      "state": "needsReview",
      "stateLabel": "Needs Review",
      "testDetails": [],
      "testStatus": null,
      "type": "default",
      "updated": 1485958875,
      "updateDate": "2017-02-01T06:21:15-08:00"
    }
  ],
  "totalCount": null
}
```

Getting reviews for the action dashboard

To list reviews:

```
curl -u "username:password" "http://my-swarm-
host/api/v7/dashboards/action"
```

Swarm responds with a list of the latest reviews, a `totalCount` field, and a `lastSeen` value for pagination:

```
{
  "lastSeen": 120,
  "reviews": [
    {
      "id": 7,
      "author": "swarm_admin",
      "changes": [6],
      "comments": [0,0],
      "commits": [6],
      "commitStatus": [],
      "created": 1485793976,
      "deployDetails": [],
      "deployStatus": null,
      "description": "test\n",
      "groups": ["swarm-project-test"],
      "participants": {"swarm_admin":[]},
      "pending": false,
      "projects": {"test":["test"]},
      "roles": ["moderator|reviewer|required_reviewer|author"],
      "state": "needsReview",
      "stateLabel": "Needs Review",
      "testDetails": [],
      "testStatus": null,
      "type": "default",
      "updated": 1485958875,
      "updateDate": "2017-02-01T06:21:15-08:00"
    }
  ],
  "totalCount": null
}
```

GET /api/v7/reviews/

Summary: Get List of Reviews

Description

List and optionally filter reviews.

Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
`after`	A review ID to seek to. Reviews up to and including the specified `id` are excluded from the results and do not count towards `max`. Useful for pagination. Commonly set to the `lastSeen` property from a previous query.	integer	query	No	
`max`	Maximum number of reviews to return. This does not guarantee that `max` reviews are returned. It does guarantee that the number of reviews returned won't exceed `max`. Server-side filtering may exclude some reviews for permissions reasons.	integer	query	No	1000

Parameter	Description	Type	Parameter Type	Required	Default Value
`fields`	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No	
`author[]`	One or more authors to limit reviews by. Reviews with any of the specified authors are returned. (v1.2+)	array (of strings)	query	No	
`change[]`	One or more change IDs to limit reviews by. Reviews associated with any of the specified changes are returned.	array (of integers)	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
`hasReviewers`	Boolean option to limit to reviews to those with or without reviewers. Use `true` or `false` for JSON-encoded data, `1` for true and or `0` for false for form-encoded data. The presence of the parameter without a value is evaluated as true.	boolean	query	No	
`ids[]`	One or more review IDs to fetch. Only the specified reviews are returned. This filter cannot be combined with the `max` parameter.	array (of integers)	query	No	
`keywords`	Keywords to limit reviews by. Only reviews where the description, participants list or project list contain the specified keywords are returned.	string	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
`participants[]`	One or more participants to limit reviews by. Reviews with any of the specified participants are returned.	array (of strings)	query	No	
`project[]`	One or more projects to limit reviews by. Reviews affecting any of the specified projects are returned.	array (of strings)	query	No	
`state[]`	One or more states to limit reviews by. Reviews in any of the specified states are returned.	array (of strings)	query	No	
`passesTests`	Boolean option to limit reviews by tests passing or failing. Use `true` or `false` for JSON-encoded data, `1` for true and `0` for false for form-encoded data. The presence of the parameter without a value is evaluated as true.	string	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
`notUpdatedSince`	Option to fetch unchanged reviews. Requires the date to be in the format YYYY-mm-dd, for example 2017-01-01. Reviews to be returned are determined by looking at the last updated date of the review.	string	query	No	
`hasVoted`	Should have the value 'up' or 'down' to filter reviews that have been voted up or down by the current authenticated user.	string	query	No	
`myComments`	True or false to support filtering reviews that include comments by the current authenticated user.	boolean	query	No	

Examples of successful responses

Successful Response:

```
HTTP/1.1 200 OK

{
  "lastSeen": 12209,
  "reviews": [
    {
      "id": 12206,
      "author": "swarm",
      "changes": [12205],
      "comments": 0,
      "commits": [],
      "commitStatus": [],
      "created": 1402507043,
      "deployDetails": [],
      "deployStatus": null,
      "description": "Review Description\n",
      "participants": {
        "swarm": []
      },
      "pending": true,
      "projects": [],
      "state": "needsReview",
      "stateLabel": "Needs Review",
      "testDetails": [],
      "testStatus": null,
      "type": "default",
      "updated": 1402518492
    }
  ],
  "totalCount": 1
}
```

Note

Swarm returns `null` for `totalCount` if no search filters were provided. `lastSeen` can often be used as an offset for pagination, by using the value in the `after` parameter of subsequent requests.

When no results are found, the `reviews` array is empty:

```
HTTP/1.1 200 OK
```

```
{
  "lastSeen": null,
  "reviews": [],
  "totalCount": 0
}
```

Examples of usage

Listing reviews

To list reviews:

```
curl -u "username:password" "https://my-swarm-
host/api/v7/reviews?max=2&fields=id,description,author,state"
```

Swarm responds with a list of the latest reviews, a `totalCount` field, and a `lastSeen` value for pagination:

```
{
  "lastSeen": 120,
  "reviews": [
    {
      "id": 123,
      "author": "bruno",
      "description": "Adding .jar that should have been included in
r110\n",
      "state": "needsReview"
    },
    {
      "id": 120,
      "author": "bruno",
      "description": "Fixing a typo.\n",
      "state": "needsReview"
    }
  ],
  "totalCount": null
}
```

The `totalCount` field is populated when keywords are supplied. It indicates how many total matches there are. If keywords are not supplied the `totalCount` field remains `null`, indicating that the list of all reviews is being queried.

Paginating a review listing

To obtain the next page of a reviews list (based on the previous example):

```
curl -u "username:password" "https://my-swarm-host/api/v7/reviews\
?max=2&fields=id,description,author,state&after=120"
```

Swarm responds with the second page of results, if any reviews are present after the last seen review:

```
{
  "lastSeen": 100,
  "reviews": [
    {
      "id": 110,
      "author": "bruno",
      "description": "Updating Java files\n",
      "state": "needsReview"
    },
    {
      "id": 100,
      "author": "bruno",
      "description": "Marketing materials for our new cutting-edge
product\n",
      "state": "needsReview"
    }
  ],
  "totalCount": null
}
```

Finding reviews for a change or a list of changes

Given a list of change IDs (5, 6, 7), here is how to check if any of them have reviews attached:

```
curl -u "username:password" "https://my-swarm-host/api/v7/reviews\
?max=2&fields=id,changes,description,author,state&change\[\]=5&change\
[\]=6&change\[\]=7"
```

Swarm responds with a list of reviews that include these changes:

```
{
  "lastSeen": 100,
  "reviews": [
    {
      "id": 110,
      "author": "bruno",
      "changes": [5],
      "description": "Updating Java files\n",
      "state": "needsReview"
    },
    {
      "id": 100,
      "author": "bruno",
      "changes": [6,7],
      "description": "Marketing materials for our new cutting-edge
product\n",
      "state": "needsReview"
    }
  ],
  "totalCount": 2
}
```

If no corresponding reviews are found, Swarm responds with an empty reviews list:

```
{
  "lastSeen": null,
  "reviews": [],
  "totalCount": 0
}
```

Finding inactive reviews (by checking the last updated date)

```
curl -u "username:password" "https://my-swarm-host/api/v7/reviews\
?max=2&fields=id,changes,description,author,state&notUpdatedSince=2017-01-
01"
```

Swarm responds with a list of reviews that have not been updated since the notUpdatedSince date:

```
{
  "lastSeen": 100,
  "reviews": [
    {
      "id": 110,
      "author": "bruno",
      "changes": [5],
      "description": "Updating Java files\n",
      "state": "needsReview"
    },
    {
      "id": 100,
      "author": "bruno",
      "changes": [6,7],
      "description": "Marketing materials for our new cutting-edge
product\n",
      "state": "needsReview"
    }
  ],
  "totalCount": 2
}
```

Finding reviews I have voted up

```
curl -u "username:password" "https://my-swarm-host/api/v7/reviews\
?max=2&fields=id,changes,description,author,state&hasVoted=up"
```

Swarm responds with a list of reviews that include these changes:

```
{
  "lastSeen": 100,
  "reviews": [
    {
      "id": 110,
      "author": "bruno",
      "changes": [5],
      "description": "Updating Java files\n",
      "state": "needsReview"
    },
    {
      "id": 100,
      "author": "bruno",
      "changes": [6,7],
      "description": "Marketing materials for our new cutting-edge
product\n",
      "state": "needsReview"
    }
  ],
  "totalCount": 2
}
```

If no corresponding reviews are found, Swarm responds with an empty reviews list:

```
{
  "lastSeen": null,
  "reviews": [],
  "totalCount": 0
}
```

Finding reviews I have commented on (current authenticated user)

```
curl -u "username:password" "https://my-swarm-host/api/v7/reviews\
?max=2&fields=id,changes,description,author,state&myComments=true"
```

Swarm responds with a list of reviews that include these changes:

```
{
  "lastSeen": 100,
  "reviews": [
    {
      "id": 110,
      "author": "bruno",
      "changes": [5],
      "description": "Updating Java files\n",
      "state": "needsReview"
    },
    {
      "id": 100,
      "author": "bruno",
      "changes": [6,7],
      "description": "Marketing materials for our new cutting-edge
product\n",
      "state": "needsReview"
    }
  ],
  "totalCount": 2
}
```

If no corresponding reviews are found, Swarm responds with an empty reviews list:

```
{
  "lastSeen": null,
  "reviews": [],
  "totalCount": 0
}
```

GET /api/v7/reviews/{id}

Summary: Get Review Information

Description

Retrieve information about a review.

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Review ID	integer	path	Yes
`fields`	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No

Successful Response:

```
HTTP/1.1 200 OK

{
  "review": {
    "id": 12204,
    "author": "bruno",
    "changes": [10667],
    "commits": [10667],
    "commitStatus": [],
    "created": 1399325913,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in
r10145\n",
    "participants": {
      "alex_qc": [],
      "bruno": {
        "vote": 1,
        "required": true
      },
      "vera": []
    },
    "reviewerGroups": {
      "group1" : [],
      "group2" : {
        "required" : true
      },
      "group3" : {
        "required" : true,
        "quorum": "1"
      }
    },
    "pending": false,
    "projects": {
      "swarm": ["main"]
    },
    "state": "archived",
    "stateLabel": "Archived",
    "testDetails": {
      "url": "http://jenkins.example.com/job/project_ci/123/"
    },
    "testStatus": null,
    "type": "default",
    "updated": 1399325913
  }
}
```

Example 404 Response:

```
HTTP/1.1 404 Not Found

{
  "error": "Not Found"
}
```

Fetching a review

To fetch a review:

```
curl -u "username:password" "https://my-swarm-host/api/v7/reviews/123"
```

Swarm responds with a review entity:

```
{
  "review": {
    "id": 123,
    "author": "bruno",
    "changes": [122,124],
    "commits": [124],
    "commitStatus": [],
    "created": 1399325913,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in r110\n",
    "groups": [],
    "participants": {
      "alex_qc": [],
      "bruno": {
        "vote": 1,
        "required": true
      },
      "vera": [],
    },
    "reviewerGroups": {
      "group1" : [],
      "group2" : {
        "required" : true
      },
      "group3" : {
        "required" : true,
        "quorum": "1"
      },
    },
    "pending": false,
    "projects": {
      "swarm": ["main"]
    },
    "state": "archived",
    "stateLabel": "Archived",
    "testDetails": {
      "url": "http://jenkins.example.com/job/project_ci/123/"
    },
    "testStatus": null,
    "type": "default",
    "updated": 1399325913,
    "versions": []
  }
}
```

POST /api/v7/reviews/

Summary: Create a Review

Description

Pass in a changelist ID to create a review. Optionally, you can also provide a description and a list of reviewers.

Parameters

Parameter	Description	Type	Parameter Type	Required
`change`	Change ID to create a review from	integer	form	Yes
`description`	Description for the new review (defaults to change description)	string	form	No
`reviewers`	A list of reviewers for the new review	array (of strings)	form	No
`requiredReviewers`	A list of required reviewers for the new review (v1.1+)	array (of strings)	form	No
`reviewerGroups`	A list of required reviewers for the new review (v7+)	array	form	No

Successful Response contains Review Entity:

```
HTTP/1.1 200 OK

{
  "review": {
    "id": 12205,
    "author": "bruno",
    "changes": [10667],
    "commits": [10667],
    "commitStatus": [],
    "created": 1399325913,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in
r10145\n",
    "participants": {
      "bruno": []
    },
    "reviewerGroups": {
      "group1" : [],
      "group2" : {
        "required" : true
      },
      "group3" : {
        "required" : true,
        "quorum": "1"
      }
    },
    "pending": false,
    "projects": [],
    "state": "archived",
    "stateLabel": "Archived",
    "testDetails": [],
    "testStatus": null,
    "type": "default",
    "updated": 1399325913
  }
}
```

Starting a review

To start a review for a committed change or a non-empty shelved changelist:

```
curl -u "username:password" -d"change=122" "https://my-swarm-
host/api/v7/reviews/"
```

Swarm responds with the new review entity:

```
{
  "review": {
    "id": 123,
    "author": "bruno",
    "changes": [122],
    "commits": [],
    "commitStatus": [],
    "created": 1399325913,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in r110\n",
    "groups": [],
    "participants": {
      "bruno": []
    },
    "reviewerGroups": {
      "group1" : [],
      "group2" : {
        "required" : true
      },
      "group3" : {
        "required" : true,
        "quorum": "1"
      }
    },
    "pending": true,
    "projects": [],
    "state": "needsReview",
    "stateLabel": "Needs Review",
    "testDetails": [],
    "testStatus": null,
    "type": "default",
    "updated": 1399325913,
    "versions": []
  }
}
```

POST /api/v7/reviews/archive/

Summary: Archiving the inactive reviews (v6+)

Description

Archiving reviews not updated since the date (v6+)

Parameters

Parameter	Description	Type	Parameter Type	Required
`notUpdatedSince`	Updated since date. Requires the date to be in the format YYYY-mm-dd, for example 2017-01-01	string	form	Yes
`description`	A description that is posted as a comment for archiving.	string	form	Yes

Successful Response:

```
HTTP/1.1 200 OK

{
  "archivedReviews": [
    {
      "id": 836,
      "author": "swarm",
      "changes": [789],
      "commits": [],
      "commitStatus": [],
      "created": 1461164339,
      "deployDetails": [],
      "deployStatus": null,
      "description": "Review description\n",
      "groups": [],
      "participants": {
        "swarm": []
      },
      "pending": false,
      "projects": [],
      "state": "archived",
      "stateLabel": "Archived",
      "testDetails": [],
      "testStatus": null,
      "type": "default",
      "updated": 1478191607
    }
  ],
  "failedReviews": []
}
```

Archiving reviews inactive since 2016/06/30

To archive reviews not updated since 2016/06/30 inclusive:

```
curl -u "username:password" -d "notUpdatedSince=2016-06-30" \
     "https://my-swarm-host/api/v7/reviews/archive/"
```

Swarm responds with the list of archived reviews and failed reviews if there are any:

```
{
  "archivedReviews":[
    {
      "id": 911,
      "author": "swarm",
      "changes": [601],
      "commits": [],
      "commitStatus": [],
      "created": 1461164344,
      "deployDetails": [],
      "deployStatus": null,
      "description": "Touch up references on html pages.\n",
      "groups": [],
      "participants": {
        "swarm":[]
      },
      "pending": false,
      "projects": [],
      "state": "archived",
      "stateLabel": "Archived",
      "testDetails": [],
      "testStatus": null,
      "type": "default",
      "updated": 1478191605
    },
    {
      "id": 908,
      "author": "earl",
      "changes": [605],
      "commits": [],
      "commitStatus": [],
      "created": 1461947794,
      "deployDetails": [],
      "deployStatus": null,
      "description": "Remove (attempted) installation of now deleted man
pages.\n",
      "groups": [],
      "participants": {
        "swarm": []
      },
      "pending": false,
      "projects": [],
      "state": "archived",
      "stateLabel": "Archived",
      "testDetails": [],
      "testStatus": null,
      "type": "default",
      "updated": 1478191605
    }
  ],
  "failedReviews":[
    {
    }
  ]
}
```

If no reviews are archived, Swarm responds with an empty reviews list:

```
{
  "archivedReviews": [],
  "failedReviews": []
}
```

POST /api/v7/reviews/{id}/changes/

Summary: Add Change to Review

Description

Links the given change to the review and schedules an update.

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Review ID	integer	path	Yes
`change`	Change ID	integer	form	Yes

Successful Response contains Review Entity:

```
HTTP/1.1 200 OK

{
  "review": {
    "id": 12206,
    "author": "bruno",
    "changes": [10667, 12000],
    "commits": [10667, 12000],
    "commitStatus": [],
    "created": 1399325913,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in
r10145\n",
    "participants": {
      "bruno": []
    },
    "pending": false,
    "projects": [],
    "state": "archived",
    "stateLabel": "Archived",
    "testDetails": [],
    "testStatus": null,
    "type": "default",
    "updated": 1399325913
  }
}
```

Adding a change to a review

You may want to update a review from a shelved or committed change that is different from the initiating change. This is done by adding a change to the review.

To add a change:

```
curl -u "username:password" -d "change=124" "https://my-swarm-
host/api/v7/reviews/123/changes/"
```

Swarm responds with the updated review entity:

```
{
  "review": {
    "id": 123,
    "author": "bruno",
    "changes": [122, 124],
    "commits": [],
    "commitStatus": [],
    "created": 1399325913,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in r110\n",
    "groups": [],
    "participants": {
      "bruno": []
    },
    "pending": true,
    "projects": [],
    "state": "needsReview",
    "stateLabel": "Needs Review",
    "testDetails": [],
    "testStatus": null,
    "type": "default",
    "updated": 1399325913,
    "versions": [
      {
        "difference": 1,
        "stream": null,
        "change": 124,
        "user": "bruno",
        "time": 1399330003,
        "pending": true,
        "archiveChange": 124
      }
    ]
  }
}
```

POST /api/v7/reviews/{id}/cleanup

Summary: Clean up a review (v6+)

Description

Clean up a review for the given id.

Parameters

Parameter	Description	Type	Parameter Type	Required
`reopen`	Expected to be a boolean (defaulting to false). If true then an attempt will be made to reopen files into a default changelist	boolean	form	No

Successful Response:

```
HTTP/1.1 200 OK
```

```
{
  "complete": [
    {
      "1": ["2"]
    }
  ],
  "incomplete": []
}
```

Cleaning up a review with id 1.

Cleanup review number 1, reopening any files into the default changelist.

```
curl -u "username:password" -d "reopen=true" \
     "https://my-swarm-host/api/v7/reviews/1/cleanup"
```

Swarm responds with the review and the changelists cleaned. Depending on the completion they will be either detailed in 'complete' or 'incomplete'. Incomplete changelists will have messages indicating why it was not possible to complete:

```
  {
    "complete": [
      {
        "1": ["2"]
      }
    ],
    "incomplete": []
  }
```

PATCH /api/v7/reviews/{id}/state/

Summary: Transition the Review State (v2+)

Description

Transition the review to a new state. When transitioning to approved, you can optionally commit the review. (v2+)

Parameters

Parameter	Description	Type	Parameter Type	Required
`id`	Review ID	integer	path	Yes
`state`	Review State. Valid options: needsReview, needsRevision, approved, archived, rejected	string	form	Yes
`description`	An optional description that is posted as a comment for non-commit transitions. Commits that do not include a description default to using the Review description in the resulting change description.	string	form	No
`commit`	Set this flag to true and provide a state of `approved` in order to trigger the **Approve and Commit** action in Swarm.	boolean	form	No
`wait`	Instruct Swarm to wait for a commit to finish before returning.	boolean	form	No
`jobs[]`	When performing an 'Approve and Commit', one or more jobs can be attached to the review as part of the commit process.	stringArray	form	No
`fixStatus`	Provide a fix status for the attached job(s) when performing an 'Approve and Commit'. Possible status values vary by job specification, but often include: open, suspended, closed, review, fixed.	string	form	No

Examples of successful responses

Successful Response contains Review Entity:

```
HTTP/1.1 200 OK

{
  "review": {
    "id": 12207,
    "author": "bruno",
    "changes": [10667, 12000],
    "commits": [],
    "commitStatus": [],
    "created": 1399325913,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in
r10145\n",
    "participants": {
      "bruno": []
    },
    "pending": false,
    "projects": [],
    "state": "needsRevision",
    "stateLabel": "Needs Revision",
    "testDetails": [],
    "testStatus": null,
    "type": "default",
    "updated": 1399325913
  },
  "transitions": {
    "needsReview": "Needs Review",
    "approved": "Approve",
    "rejected": "Reject",
    "archived": "Archive"
  }
}
```

Successful Commit contains Review and Commit Entities:

```
HTTP/1.1 200 OK

{
  "review": {
    "id": 12208,
    "author": "bruno",
    "changes": [10667, 12000, 12006],
    "commits": [12006],
    "commitStatus": {
      "start": 1399326910,
      "change": 12006,
      "status": "Committed",
      "committer": "bruno",
      "end": 1399326911
    },
    "created": 1399325900,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in
r10145\n",
    "participants": {
      "bruno": []
    },
    "pending": false,
    "projects": [],
    "state": "needsRevision",
    "stateLabel": "Needs Revision",
    "testDetails": [],
    "testStatus": null,
    "type": "default",
    "updated": 1399325905
  },
  "transitions": {
    "needsReview": "Needs Review",
    "needsRevision": "Needs Revision",
    "rejected": "Reject",
    "archived": "Archive"
  },
  "commit": 12006
```

```
}
```

Committing a review

To commit a review:

```
curl -u "username:password" -X PATCH -d "state=approved" -d "commit=1" \
     "https://my-swarm-host/api/v7/reviews/123/state/"
```

Swarm responds with the updated review entity, as well as a list of possible transitions for the review:

```
{
  "review": {
    "id": 123,
    "author": "bruno",
    "changes": [122, 124],
    "commits": [124],
    "commitStatus": {
        "start": 1399326910,
        "change": 124,
        "status": "Committed",
        "committer": "bruno",
        "end": 1399326911
      },
    "created": 1399325913,
    "deployDetails": [],
    "deployStatus": null,
    "description": "Adding .jar that should have been included in r110\n",
    "groups": [],
    "participants": {
      "bruno": []
    },
    "pending": false,
    "projects": [],
    "state": "approved",
    "stateLabel": "Approved",
    "testDetails": [],
    "testStatus": null,
    "type": "default",
    "updated": 1399325913,
    "versions": []
  },
    "transitions": {
      "needsReview": "Needs Review",
      "approved": "Approve",
      "rejected": "Reject",
      "archived": "Archive"
    }
}
```

PATCH /api/v7/reviews/{review_id}

Summary: Update Review Description

Description

Update the description field of a review.

Parameters

Parameter	Description	Type	Parameter Type	Required
`review_id`	Review ID	integer	path	Yes
`author`	The new author for the specified review. (At least one of Author or Description are required.)	string	form	No
`description`	The new description for the specified review. (At least one of Description or Author are required.)	string	form	No
`_method`	Method Override. If your client cannot submit HTTP PATCH, use an HTTP POST with the parameter ?_method=PATCH to override.	string	query	No

Examples of successful responses

Successful Response:

```
HTTP/1.1 200 OK

{
  "review": {
      "id": 12306,
      "author": "swarm",
      "changes": [12205],
      "comments": 0,
      "commits": [],
      "commitStatus": [],
      "created": 1402507043,
      "deployDetails": [],
      "deployStatus": null,
      "description": "Updated Review Description\n",
      "participants": {
        "swarm": []
      },
      "pending": true,
      "projects": [],
      "state": "needsReview",
      "stateLabel": "Needs Review",
      "testDetails": [],
      "testStatus": null,
      "type": "default",
      "updated": 1402518492
  },
  "transitions": {
      "needsRevision": "Needs Revision",
      "approved": "Approve",
      "rejected": "Reject",
      "archived": "Archive"
  },
  "canEditAuthor": true
}
```

Note

Swarm returns `null` for `totalCount` if no search filters were provided. `lastSeen` can often be used as an offset for pagination, by using the value in the `after` parameter of subsequent requests.

When no results are found, the `reviews` array is empty:

```
HTTP/1.1 200 OK

{
  "lastSeen": null,
  "reviews": [],
  "totalCount": 0
}
```

Extended API example

This section includes an extended API example, involving multiple API calls to answer a more complicated kind of question than any single API endpoint can provide: which reviews does a specific userid need to attend to?

The code

```
<?php
/**
 * vim:set ai si et ts=4 sw=4 syntax=php:
 *
 * reviews.php
 *
 * Queries the Swarm API and reports which reviews a specified user
 * needs to attend to.
 *
 * Required attention is determined by the following criteria:
 * - the user is a participant in a review
 * - and the user has not voted on the review
 * - and the user has not commented on the review
 * - or the user's comment on the review is a
 *   task that has been addressed and needs verification
 */

if (ini_set('track_errors', 1) === false) {
    echo "Warning: unable to track errors.\n";
}

# process command-line arguments
$options = getopt(
    'hs:r:v',
    array('help', 'swarm:', 'reviewer', 'verbose')
);

$swarm = '';
if (isset($options['s'])) {
    $swarm = $options['s'];
}
if (isset($options['swarm'])) {
    $swarm = $options['swarm'];
}
if (!$swarm) {
    usage('Swarm API URL not provided.');
}

$reviewer = '';
if (isset($options['r'])) {
    $reviewer = $options['r'];
}
if (isset($options['reviewer'])) {
    $reviewer = $options['reviewer'];
}
if (!$reviewer) {
    usage('Swarm reviewer not provided.');
}
```

```
$verbose = false;
if (isset($options['v']) || isset($options['verbose'])) {
    $verbose = true;
}

if (isset($options['h']) || isset($options['help'])) {
    usage();
}

function usage($message = null)
{
    if ($message) {
        echo "$message\n\n";
    }

    $script = basename(__FILE__);
    echo <<<EOU
$script: -s <Swarm URL> -u <API userid> -p <API user's password> \
  -r <reviewer userid to report on> -h

-s|--swarm     Swarm's URL (e.g. https://user@password:myswarm.url/)
-r|--reviewer  The reviewer to report on.
-h|--help      This help text.
-v|--verbose   Verbose output.

This script queries the Swarm API and reports on reviews that the
specified user needs to attend to.

Note: If your Helix Versioning Engine (p4d) has security level 3 set, you
cannot use a password to authenticate; you must acquire a host-unlocked
ticket from p4d, and use the ticket in place of a password when
communicating with the Swarm API connected to p4d.

EOU;
    exit;
}

function msg($message)
{
    global $verbose;

    if ($verbose) {
        echo $message;
    }
}

function call_api($url, $params)
{
    global $php_errormsg;

    $query    = http_build_query($params);
    $request  = $url . '?' . $query;
    $response = @file_get_contents($request);
    if ($php_errormsg) {
        echo "Unable to call api: $php_errormsg\n";
        exit;
    }

    $json = @json_decode($response, true);
    if ($php_errormsg) {
        echo "Unable to decode api response: $php_errormsg\n";
        exit;
    }

    return $json;
}

# remove trailing / from URL, if it exists
$swarm = rtrim(trim($swarm), '/');

# fetch the list of reviews
```

```
$reviews = call_api(
    "$swarm/api/v4/reviews",
    array(
        'hasReviewers' => 1, # only reviews with participants
        'participants' => array($reviewer), # only review for this
reviewer
        'max'          => 9, # get plenty of reviews, if available
        'fields'       => array('id', 'description', 'commits'), # get
these fields
    )
);

$report = array();
foreach ($reviews['reviews'] as $review) {
    if (is_null($review)) {
        continue;
    }

    $flag = false;
    msg('Review: ' . $review['id'] . ' ');

    # if the review is already committed, it likely does not need
attention
    if (array_key_exists('commits', $review)
        && count($review['commits'])
    ) {
        msg("is committed, skipping...\n");
        continue;
    }

    # if the review has a vote from the reviewer, they are already aware
    if (array_key_exists('participants', $review)
        && array_key_exists('vote', $review['participants'][$reviewer])
    ) {
        msg("has vote from reviewer, skipping...\n");
        continue;
    }

    # if there are no open comments on the review, the reviewer's
    # attention is required
    if (array_key_exists('comments', $review)
        && $review['comments'][0] == 0
    ) {
        msg("has no open comments, skipping...\n");
        continue;
    }

    # fetch the comments for this review
    $comments = call_api(
        "$swarm/api/v4/comments",
        array(
            'topic' => 'reviews/' . $review['id'], # comments for this
review
            'max'   => 9, # get plenty of comments, if available
        )
    );

    foreach ($comments['comments'] as $comment) {
        msg("\n  Comment: " . $comment['id'] . ' ');

        // skip over comments from other reviewers
        if (array_key_exists('user', $comment) && $reviewer != $comment
['user']) {
            msg("is by another user, carry on...\n");
            continue;
        }

        # skip archived comments
        if (array_key_exists('flags', $comment)
            && count($comment['flags']) > 0
            && $comment['flags'][0] == 'closed'
        ) {
            msg("is archived, carry on...\n");
            continue;
        }

        # skip marked tasks
        if (array_key_exists('taskState', $comment)
            && ($comment['taskState'] == 'comment'
                || $comment['taskState'] == 'verified'
                || $comment['taskState'] == 'open'
            )
        ) {
            msg("reviewer's comment needs attention, carry on...\n");
            continue;
        }

        // anything else means that the reviewer's comment needs attention
        // by the reviewer
        $flag = true;
        msg("needs attention!\n");
        break;
    }

    // evaluation is complete. Does this review need attention?
    if ($flag) {
        $report[] = $review;
    }
}

if (count($report)) {
    echo "User '$reviewer' needs to attend to these reviews:\n";
    foreach ($report as $review) {
        $description = trim($review['description']);
        if (strlen($description) > 60) {
            $description = substr($description, 0, 60) . ' ...';
        }
        echo $review['id'] . ": $description\n";
    }
} else {
    echo "User '$reviewer' has no reviews to attend to.\n";
}
```

Executing the example

The example is written in PHP. To use it, download the code, or copy and paste it into a file called `reviews.php`. Then, execute it like this:

```
$ php reviews.php -s https://myswarm.host:port/ -r bob
```

Replace ***http://myswarm.host/*** with the URL to your Swarm installation. Replace ***bob*** with the userid you'd like to report on.

To authenticate, insert `username:password@` before the hostname. If your Helix Versioning Engine's security counter is set to `3` or higher, you need to acquire a ticket and use the ticket in place of the password (see Authentication for details). If your Swarm is installed on a custom port, or is installed in a sub-folder, include those elements in the URL as well. For example:

```
$ php reviews.php -s
https://me:F0FC33068BA244B1BBD8196CC9166F34@my.host:8080/swarm/ -
r bob
```

If you do not specify the URL correctly, you might see an error like:

```
Unable to call api: file_get_contents
(http://...@my.host:8080/swarm/api/v7/reviews?hasReviewers=1&parti
cipants%5B0%5D=bob&max=9&fields%5B0%5D=id&fields%5B1%5D=descriptio
n&fields%5B2%5D=commits): failed to open stream: HTTP request
failed! HTTP/1.1 404 Not Found
```

If there are no errors, and the specified userid does have reviews to attend to, the output might look like:

```
1234: Added grapple-grommit support to woozlewobble class. @bob sh
...
```

`1234` is the id of a review that `bob` should attend to, followed by the first 60 characters of the review's description.

Pending Review Cleanup

This section contains an example script that cleans up pending changelists which are no longer needed. See the review cleanup options for how this can be done automatically when a review is committed.

For pending changelists which were present before this option was available, or for reviews which have been contributed to by multiple authors and so require super user access to tidy up, there is an API which allows the super user to bulk remove such changelists.

The script demonstrates how this API could be used. It isn’t meant as a complete solution, just a guide to demonstrate what is possible.

The code

```
<?php
/**
 * Perforce Swarm
 *
 * @copyright   2017 Perforce Software. All rights reserved.
 * @license     Please see LICENSE.txt in top-level folder of this
distribution.
 * @version     <release>/<patch>
 */
/**
 * This example script can be used to clean up (delete) Perforce Server
pending changelists automatically when run
 * as a super user. It is able to query for reviews based on parameters to
establish which changelists are eligible
 * for clean up. In this way, it can be tailored to run against reviews of
a user's choice.

 * Requirements for this script:
 *   - MUST be a super user
 *   - MUST populate the parameters below
 *   - MUST be using Swarm 2017.1 or later
 *
 * Usage of script
 *   php superUserReviewCleanUp.php max=10 notUpdatedSince=2017-04-01
state=approved
 *   php superUserReviewCleanUp.php max=10 author=bruno state=approved
 *
 * Each of the parameters that you can use are documented at the following
URL:
 *
```

```
https://www.perforce.com/perforce/doc.current/manuals/swarm/api.endpoints.
html#api.endpoints.Reviews.getReviews
 *
 *
 * This returns a JSON object that contains four main objects.
 *
 * {
 *   "error": "",
 *   "help": "",
 *   "results": "[]",
 *   "search_criteria": {
 *     "fields": "id",
 *     "max": "10",
 *     "notUpdatedSince": "2017-04-01"
 *   }
 * }
 *
 * Referencing the example above:
 *
 * The error section will contain any errors that have been encountered
trying to execute the script.
 *
 * The help section will populated if you have run the command php
superUserReviewCleanUp.php help
 *
 * The search_criteria section indicates which parameters have been used
to fetch the reviews list.
 *
 * The results section returns a JSON object of each of the reviews it has
processed, which may
 * include actions that were incomplete.
 *
 * Below is an example of results being processed:
 * "814": {
 *    "complete": [],
 *    "incomplete": {
```

```
 *       "814": {
 *           "813": [
 *               "0": "Command failed: No shelved files in changelist to
delete.",
 *           ]
 *       }
 *    }
 * },
 * "818": {
 *    "complete": [],
 *    "incomplete": []
 * }
 * "820": {
 *    "complete": [],
 *    "incomplete": {
 *       "820": {
 *           "821": [
 *              "0": "Command failed: No shelved files in changelist to
delete.",
 *              "1": "Command failed: Usage: fix [ -d ] [ -s status ] -c
changelist# jobName ...\nMissing/wrong
 *                    number of arguments."
 *           ]
 *       }
 *    }
 * },
 *
 * Some reviews may have no actions or incomplete actions. Incomplete
actions indicate that additional work is required
 * and the review could not be entirely cleaned up. In the example above,
the first message indicates a changelist was
 * not found. This could be because the end user has already deleted it.
 *
 * An error with the fix command can indicate that the pending changelist
doesn't have any jobs linked or that the jobs
 * have already been removed.
 *
 * NOTES:
 * To make the the output print nicely, you can use the python command
like this:
 *
 * php superUserReviewCleanUp.php max=10 notUpdatedSince=2018-04-01
state=approved | python -m json.tool
 *
 */



/* **************************************************** */
/* These values MUST be set before running the script.  */
/* **************************************************** */

// URL to access swarm. Must not include trailing slash.
$swarmURL = 'http://my.swarm.com';
// Username of super user
$username = '';
// Ticket for user, can be created by running "p4 -u $username login -pa"
$ticket = '';

/* **************************************************** */



// If the super user wants to reopen the files of the end user.
$reopen = true;
// Prebuild the the return message helper array.
$help = array( "help" => "", "error" => "", "results" => "", "search_
criteria" => "");
// @codingStandardsIgnoreEnd
/**
 * function that make the GET or POST requests
 *
 * @param $url          Url in which we want to make our request to
 * @param bool $fetch   Set to true for a GET request, otherwise will do a
```

```
POST.
 * @param array $args  These are the parameter that we pass the the GET or
POST request to filter the reviews
 * @return JSON        We return a JSON object back to be worked on.
 */
function request($url, $fetch = false, $args = array())
{
    // Fetch the settings to allow this function to access them.
    global $username, $ticket, $reopen, $help;
    $curl = '';

    // If GET request fetch should be true and args shouldn't be empty
    if ($fetch === true) {
        // If is args is empty just give the url, otherwise build a http
query with args elements
        $curl = empty($args) ? curl_init($url) : curl_init($url."?".http_
build_query($args));
    } else {
        // Assume fetch is false and build a POST request.
        $curl = curl_init($url."/".$args['id'].'/cleanup');
        curl_setopt($curl, CURLOPT_POSTFIELDS, http_build_query(array
('reopened'=>$reopen)));
        curl_setopt($curl, CURLOPT_POST, count($reopen));
    }
    curl_setopt($curl, CURLOPT_USERPWD, "$username:$ticket");
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($curl, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);

    $result = curl_exec($curl);

    // Catch the error code in case Ticket expired or url doesn't return.
    $statusCode = curl_getinfo($curl, CURLINFO_HTTP_CODE);
    if ($statusCode != 200) {
        $help['error'] = array("HTTP code" => $statusCode);
    }
    curl_close($curl);
    return json_decode($result, true);
}

/**
 * Fetch a list of Reviews based on parameters
 *
 * @param $elements  Each of the args passed from command line are treated
as elements
* @return JSON      We return a JSON object back to be worked on.
 */
function fetchReviews($elements)
{
    global $swarmURL, $help;
    $parameters = array();

    if ($elements !== null) {
        // Loop though each of the args we want to fetch reviews based on
        foreach ($elements as $key => $value) {
            // Break each of the element into key and value to allow use
to build a array to pass to url
            $brokenDown = explode("=", $value);
            if ($key !== 0 && sizeof($brokenDown) === 2) {
                $parameters[$brokenDown[0]] = $brokenDown[1];
            }
        }
        // We only require the field id to limit the amount of data.
        $parameters['fields'] = 'id';
        // Helpful for debugging which parameters we have passed in the
queue.
        $help['search_criteria'] = $parameters;
    }
    // Now make the request to the Swarm server with your field options.
    $result = request(
        "$swarmURL/api/v6/reviews",
        true,
        $parameters
    );

    // Return the JSON object back to be worked on.
    return $result;
}

/**
 * Loop though each of the Reviews passed in and run clean up for them.
 *
 * @param $reviews  JSON object of all the reviews we want to run cleanup
on
 * $reviews => array(
 *       'reviews' => array(
 *           0 => array (
 *                   'id' => 134,
 *           ),
 *           1 => array (
 *                   'id' => 136,
 *           ),
 *           2 => array (
 *                   'id' => 143,
 *           ),
 *           3 => array (
 *                   'id' => 158,
 *           )
 *       )
 * )
 *
 * @return $array   return the array of work that has been carried out.
 */
function runCleanUp($reviews)
{
    global $swarmURL;
    $results = array();
    // Check if there is an reviews element of the array
    if (isset($reviews['reviews'])) {
        foreach ($reviews['reviews'] as $review) {
            // Now make the request to the Swarm for each review.
            $results[$review['id']] = request(
                "$swarmURL/api/v6/reviews",
                false,
                $review
            );
        }
    }
    return $results;
}

/**
 * The help function in case a user doesn't set the basic settings.
 *
 * @param $help    Pass the help array from main script to append the
helpful message.
 * @return $array  Return the array that will be presented to the users.
 */
function helpMessage($help)
{
    $help["help"]      = array( "1" => "", "2" => "", "3" => "" );
    $help["help"]["1"] = "Please ensure you have set the Username, Ticket
and Swarm URL before using this script";
    $help["help"]["2"] = "Running the script can be done by using any of
the standard Swarm API fields for reviews";
    $help["help"]["3"] = "Visit
https://www.perforce.com/perforce/doc.current/manuals/swarm/index.html";
    return $help;
}

// check the first argument is not help.
$helpSet = isset($argv[1]) && $argv[1] == "help" ? "help" : null;

// Check if the user has given help as a command to this script.
```

```
if (isset($argv[1]) && $helpSet == "help") {
    // Set the $help array with the help message.
    $help = helpMessage($help);
}

// Check if the basic user ticket and swarmurl are set.
if (!empty($username) && !empty($ticket) && !empty($swarmURL) && $helpSet
!= "help") {
    try {
        $help["results"] = runCleanUp(fetchReviews($argv));
    } catch (Exception $e) {
        $help = helpMessage($help);
    }
} else {
    $message    = "Please ensure you have set the below before using this
script";
    $errorArray = array("message" => $message, "parameter" => "");

    $missingParameter = array();

    // Now check if the basic settings are empty and show the end user.
    empty($username) ? $missingParameter[] = "Username":'';
    empty($ticket) ? $missingParameter[] = "Ticket":'';
    empty($swarmURL) ? $missingParameter[] = "SwarmURL":'';

    $errorArray['parameter'] = $missingParameter;

    $help["error"] = $errorArray ;
}
// Output the end result of the what the script does.
echo json_encode($help, JSON_FORCE_OBJECT);
// @codingStandardsIgnoreEnd
```

Executing the script

The example is written in PHP, and demonstrates how to make use of the APIs which remove unneeded pending changelists. It **must** be run as a super user.

For a full set of instructions on how to use the example script, see the comments in the script itself.

Abbreviated instructions:

1. Set the value of the `$swarmURL`, `$username` and `$ticket` variables.
2. Run the script by using a command similar to the following:

```
$ php pendingReviewCleanUp.php max=10 author=bruno state=approved
```

A | Contact Perforce

We look forward to hearing about your experiences with Swarm, positive or negative, including *must-haves* or *it would be great if Swarm...*. Feel free to contact us:

- **Post a message on our public forums.**

  The forums are a great way to discuss Swarm with other Swarm users and the Swarm development team.

- **Email support@perforce.com and reference "Swarm" in the subject line.**

  Email is preferable when you need direct assistance.

- **Phone us and ask for assistance with Swarm.**

  When you need immediate assistance, calling us is the quickest way to resolve a Swarm problem. We have international offices; call the office closest to you:

  - North America: +1 510.864.7400
  - Europe: +44 (0) 1189 771020
  - Australia: +61 2 8912-4600

Complete contact information is available on the Perforce web site.

License statements

This distribution includes the following Perforce software; please consult the following for the license terms that apply to these pieces of software only:

P4PHP, the Perforce extension for PHP
Location `SWARM_ROOT/p4-bin/bin.platform`
Terms of Use: https://www.perforce.com/downloads/terms-use

This distribution also includes the following third party software; please consult the accompanying license file for the license terms that apply to that software only:

Arimo Font
Location: `SWARM_ROOT/public/vendor/arimo`
License: `SWARM_ROOT/public/vendor/arimo/OFL.txt`

Bootstrap
Location: `SWARM_ROOT/public/vendor/bootstrap`
License: `SWARM_ROOT/public/vendor/bootstrap/LICENSE`

Cousine Font
Location:`SWARM_ROOT/public/vendor/cousine`
License: `SWARM_ROOT/public/vendor/cousine/OFL.txt`

Git Logo
Location: `SWARM_ROOT/public/vendor/git`
License: `SWARM_ROOT/public/vendor/git/CC-LICENSE.txt`

Google Diff-Match-Patch
Location: `SWARM_ROOT/public/vendor/diff_match_patch`
License: `SWARM_ROOT/public/vendor/diff_match_patch/COPYING`

Google Code Prettify
Location: `SWARM_ROOT/public/vendor/prettify`
License: `SWARM_ROOT/public/vendor/prettify/COPYING`

jQuery
Location: `SWARM_ROOT/public/vendor/jquery`
License file: `SWARM_ROOT/public/vendor/jquery/MIT-LICENSE.txt`

jQuery Expander
Location: `SWARM_ROOT/public/vendor/jquery.expander`
License file: `SWARM_ROOT/public/vendor/jquery.expander/license.txt`

jQuery Sortable
Location: `SWARM_ROOT/public/vendor/jquery-sortable`
License file: `SWARM_ROOT/public/vendor/jquery-sortable/LICENSE.txt`

jQuery Timeago
Location: `SWARM_ROOT/public/vendor/jquery.timeago`
License file: `SWARM_ROOT/public/vendor/jquery.timeago/MIT-LICENSE.txt`

JsRender
Location: `SWARM_ROOT/public/vendor/jsrender`
License file: `SWARM_ROOT/public/vendor/jsrender/MIT-LICENSE.txt`

Parsedown
Location: `SWARM_ROOT/library/Parsedown`
License file: `SWARM_ROOT/library/Parsedown/LICENSE.txt`

ThreeJS

Location: `SWARM_ROOT/public/vendor/threejs`
License file: `SWARM_ROOT/public/vendor/threejs/LICENSE`

Zend Framework 2

Location: `SWARM_ROOT/library/Zend`
License file: `SWARM_ROOT/library/Zend/LICENSE.txt`