Search

Gerrit Code Review - Change-Ids

Version 2.16.2-703-G1ff5ae9244 | Privacy (Https://Policies.Google.Com/Privacy)

Table of Contents

Description

Gerrit needs to identify commits that belong to the same review. For instance, when a change needs to be modified, a second commit can be uploaded to address the reported issues. Gerrit allows attaching those 2 commits to the same change, and relies upon a Change-Id line at the bottom of a commit message to do so. With this Change-Id, Gerrit can automatically associate a new version of a change back to its original review, even across cherry-picks and rebases.

To be picked up by Gerrit, a Change-Id line must be in the footer (last paragraph) of a commit message, and may be mixed together with Signed-off-by, Acked-by, or other such lines. For example:

```
$ git log -1
commit 29a6bb1a059aef021ac39d342499191278518d1d
Author: A. U. Thor <author@example.com>
Date: Thu Aug 20 12:46:50 2009 -0700

    Improve foo widget by attaching a bar.

    We want a bar, because it improves the foo by providing more
    wizbangery to the dowhatimeanery.

    Bug: #42
    Change-Id: Ic8aaa0728a43936cd4c6e1ed590e01ba8f0fbf5b
    Signed-off-by: A. U. Thor <author@example.com>
    CC: R. E. Viewer <reviewer@example.com>
```

In the above example, `Ic8aaa0728a43936cd4c6e` assigned to this change. It is independent of the commit id. To avoid confusion with commit ids, Change-Ids are typically prefixed with an uppercase `I`.

Note that a Change-Id is not necessarily unique within a Gerrit instance. It can be reused among different repositories or branches (see below, change-upload).

Creation

Change-Ids are created at commit time on the client side. A standard 'commit-msg' hook is provided by Gerrit, and can be installed in the local Git repository to automatically generate and insert a Change-Id line during `git commit`, when none is defined yet.

To install the hook, copy it from Gerrit's daemon by executing one of the following commands while being in the root directory of the local Git repository:

```
$ curl -Lo .git/hooks/commit-msg http://review.example.com/tools/hooks/commit-msg
```

or:

```
$ scp -p -P 29418 john.doe@review.example.com:hooks/commit-msg .git/hooks/
```

Then ensure that the execute bit is set on the hook script:

```
$ chmod u+x .git/hooks/commit-msg
```

For more details, see commit-msg.

Change Upload

During upload by pushing to `refs/for/*` or `refs/heads/*`, Gerrit will try to find an existing review the uploaded commit relates to. For an existing review to match, the following properties have to match:

- Change-Id
- Repository name
- Branch name

The following applies in the different scenarios:

- Create a new change

  If no matching review is found, Gerrit will create a new change for review.

- Update an existing change

  If a matching review is found, Gerrit will add the new commit as a new patch set on the existing change.

- Close an existing change

  If a matching review is found, and the commit is being pushed directly to refs/heads/*, the existing change is updated with the new commit, and the change is closed and marked as merged.

If a Change-Id line is not present in the commit message, Gerrit will automatically generate its own Change-Id and display it on the web. This line can be manually copied and inserted into an updated commit message if additional revisions to a change are required.

By default, Gerrit will prevent pushing for review if no Change-Id is provided, with the following message:

```
! [remote rejected] HEAD -> refs/for/master (missing Change-Id in commit
message footer)
```

However, repositories can be configured to allow commits without Change-Ids in the commit message by setting "Require Change-Id in commit message" to "FALSE".

For more details on using git push to upload changes to Gerrit, see creating changes by git push.

Git Tasks

Creating a new commit

When creating a new commit, ensure the 'commit-msg' hook has been installed in your repository (see above), and don't put a Change-Id line in the commit message. When you exit the editor, git will call the hook, which will automatically generate and insert a unique Change-Id line. You can inspect the modified message after the commit is complete by executing `git show`.

Amending a commit

When amending a commit with `git commit --a` ed in the commit message. This will allow Gerrit to automatically update the change with the amended commit.

Rebasing a commit

When rebasing a commit, leave the Change-Id line unmodified in the commit message. This will allow Gerrit to automatically update the change with the rebased commit.

Squashing commits

When squashing several commits together, try to preserve only one Change-Id line, and remove the others from the commit message. When faced with multiple lines, try to preserve a line which was already uploaded to Gerrit Code Review, and thus has a corresponding change that reviewers have already examined and left comments on. If you aren't sure which lines Gerrit knows about, try copying and pasting the lines into the search box at the top-right of the web interface.

If Gerrit already knows about more than one Change-Id, pick one to keep in the squashed commit message, and manually abandon the other changes through the web interface.

Cherry-picking a commit

When cherry-picking a commit, leave the Change-Id line alone to have Gerrit treat the cherry-picked commit as a replacement for the existing change. This can be very useful if the project has a fast-forward-only merge policy, and the submitter is downloading and cherry-picking individual changes prior to submission, such as by gerrit-cherry-pick.

Or, you may wish to delete the Change-Id line and force a new Change-Id to be generated automatically, thus creating an entirely new change record for review. This may be useful when backporting a change from the current development branch to a maintenance release branch.

Updating an old commit

If a commit was created before the availability of Change-Id support, or was created in a Git repository that was missing the 'commit-msg' hook, simply copy the "Change-Id: I..." line from the first line of the Description section of the change and amend it to the bottom of the commit message. Any subsequent uploads of the commit will be automatically associated with the prior change.
