# Integrating Zuul with Gerrit

[Gerrit](https://www.gerritcodereview.com/) is a code review system. The Gerrit driver supports sources, triggers, and reporters.

Zuul will need access to a Gerrit user.

Create an SSH keypair for Zuul to use if there isn’t one already, and create a Gerrit user with that key:

**cat ~/id\_rsa.pub | ssh -p29418 review.example.com gerrit create-account --ssh-key - --full-name Zuul zuul**

Give that user whatever permissions will be needed on the projects you want Zuul to report on. For instance, you may want to grant Verified +/-1 and Submit to the user. Additional categories or values may be added to Gerrit. Zuul is very flexible and can take advantage of those.

## **Connection Configuration**

The supported options in zuul.conf connections are:

**<gerrit connection>**

**<gerrit connection>.driver*(required)***

**gerrit**

The connection must set driver=gerrit for Gerrit connections.

**<gerrit connection>.server*(required)***

Fully qualified domain name of Gerrit server.

**<gerrit connection>.port** **Default:29418**

Gerrit server port.

**<gerrit connection>.user** **Default:zuul**

User name to use when logging into Gerrit via ssh.

**<gerrit connection>.sshkey  
Default:~zuul/.ssh/id\_rsa**

Path to SSH key to use when logging into Gerrit.

**<gerrit connection>.keepalive** **Default:60**

SSH connection keepalive timeout; 0 disables.

**<gerrit connection>.password**

The HTTP authentication password for the user. This is optional, but if it is provided, Zuul will report to Gerrit via HTTP rather than SSH. It is required in order for file and line comments to reported (the Gerrit SSH API only supports review messages). Retrieve this password from the HTTP Password section of the Settings page in Gerrit.

**<gerrit connection>.auth\_type  
Default:basic**

The HTTP authentication mechanism.

**basic**

HTTP Basic authentication; the default for most Gerrit installations.

**digest**

HTTP Digest authentication; only used in versions of Gerrit prior to 2.15.

**form**

Zuul will submit a username and password to a form in order to authenticate.

**gcloud\_service**

Only valid when running in Google Cloud. This will use the default service account to authenticate to Gerrit. Note that this will only be used for interacting with the Gerrit API; anonymous HTTP access will be used to access the git repositories, therefore private repos or draft changes will not be available.

**<gerrit connection>.verify\_ssl  
Default:true**

When using a self-signed certificate, this may be set to false to disable SSL certificate verification.

## **Trigger Configuration**

Zuul works with standard versions of Gerrit by invoking the gerrit stream-events command over an SSH connection. It also reports back to Gerrit using SSH.

If using Gerrit 2.7 or later, make sure the user is a member of a group that is granted the Stream Events permission, otherwise it will not be able to invoke the gerrit stream-events command over SSH.

**pipeline.trigger.<gerrit source>**

The dictionary passed to the Gerrit pipeline trigger attribute supports the following attributes:

**pipeline.trigger.<gerrit source>.event*(required)***

The event name from gerrit. Examples: patchset-created, comment-added, ref-updated. This field is treated as a regular expression.

**pipeline.trigger.<gerrit source>.branch**

The branch associated with the event. Example: master. This field is treated as a regular expression, and multiple branches may be listed.

**pipeline.trigger.<gerrit source>.ref**

On ref-updated events, the branch parameter is not used, instead the ref is provided. Currently Gerrit has the somewhat idiosyncratic behavior of specifying bare refs for branch names (e.g., master), but full ref names for other kinds of refs (e.g., refs/tags/foo). Zuul matches this value exactly against what Gerrit provides. This field is treated as a regular expression, and multiple refs may be listed.

**pipeline.trigger.<gerrit source>.ignore-deletes** **Default:true**

When a branch is deleted, a ref-updated event is emitted with a newrev of all zeros specified. The ignore-deletes field is a boolean value that describes whether or not these newrevs trigger ref-updated events.

**pipeline.trigger.<gerrit source>.approval**

This is only used for comment-added events. It only matches if the event has a matching approval associated with it. Example: Code-Review: 2 matches a +2 vote on the code review category. Multiple approvals may be listed.

**pipeline.trigger.<gerrit source>.email**

This is used for any event. It takes a regex applied on the performer email, i.e. Gerrit account email address. If you want to specify several email filters, you must use a YAML list. Make sure to use non greedy matchers and to escapes dots! Example: email: ^.\*?@example\.org$.

**pipeline.trigger.<gerrit source>.username**

This is used for any event. It takes a regex applied on the performer username, i.e. Gerrit account name. If you want to specify several username filters, you must use a YAML list. Make sure to use non greedy matchers and to escapes dots. Example: username: ^zuul$.

## **Reporter Configuration**

Zuul works with standard versions of Gerrit by invoking the gerrit command over an SSH connection, unless the connection is configured with an HTTP password, in which case the HTTP API is used.

**pipeline.reporter.<gerrit reporter>**

The dictionary passed to the Gerrit reporter is used to provide label values to Gerrit. To set the Verified label to 1, add verified: 1 to the dictionary.

The following additional keys are recognized:

**pipeline.reporter.<gerrit reporter>.submit** **Default:False**

Set this to True to submit (merge) the change.

**pipeline.reporter.<gerrit reporter>.comment** **Default:True**

If this is true (the default), Zuul will leave review messages on the change (including job results). Set this to false to disable this behavior (file and line commands will still be sent if present).

A [connection](https://zuul-ci.org/docs/zuul/4.11.0/reference/connections.html#connections) that uses the gerrit driver must be supplied to the trigger.

## **Requirements Configuration**

As described in [pipeline.require](https://zuul-ci.org/docs/zuul/4.11.0/reference/pipeline_def.html" \l "attr-pipeline.require" \o "attr-pipeline.require) and [pipeline.reject](https://zuul-ci.org/docs/zuul/4.11.0/reference/pipeline_def.html" \l "attr-pipeline.reject" \o "attr-pipeline.reject), pipelines may specify that items meet certain conditions in order to be enqueued into the pipeline. These conditions vary according to the source of the project in question. To supply requirements for changes from a Gerrit source named my-gerrit, create a configuration such as the following:

**pipeline:**

**require:**

**my-gerrit:**

**approval:**

**- Code-Review: 2**

This indicates that changes originating from the Gerrit connection named my-gerrit must have a Code-Review vote of +2 in order to be enqueued into the pipeline.

**pipeline.require.<gerrit source>**

The dictionary passed to the Gerrit pipeline require attribute supports the following attributes:

**pipeline.require.<gerrit source>.approval**

This requires that a certain kind of approval be present for the current patchset of the change (the approval could be added by the event in question). It takes several sub-parameters, all of which are optional and are combined together so that there must be an approval matching all specified requirements.

**pipeline.require.<gerrit source>.approval.username**

If present, an approval from this username is required. It is treated as a regular

**pipeline.reject.<gerrit source>.approval**

This takes a list of approvals. If an approval matches the provided criteria the change can not be entered into the pipeline. It follows the same syntax as [pipeline.require.<gerrit source>.approval](https://zuul-ci.org/docs/zuul/4.11.0/reference/drivers/gerrit.html" \l "attr-pipeline.require.%3Cgerrit%20source%3E.approval" \o "attr-pipeline.require.<gerrit source>.approval).

Example to reject a change with any negative vote:

**reject:**

**my-gerrit:**

**approval:**

**- Code-Review: [-1, -2]**

## **Reference Pipelines Configuration**

Here is an example of standard pipelines you may want to define:

- **pipeline**:

**name**: check

**description**: |

Newly uploaded patchsets enter this pipeline to receive an

initial +/-1 Verified vote.

**manager**: independent

**precedence**: low

**require**:

**gerrit**:

**open**: True

**current-patchset**: True

**trigger**:

**gerrit**:

- **event**: patchset-created

- **event**: change-restored

- **event**: comment-added

**comment**: (?i)^(Patch Set [0-9]+:)?( [\w\\+-]\*)\*(\n\n)?\s\*recheck

**success**:

**gerrit**:

*# Note that gerrit keywords are case-sensitive.*

**Verified**: 1

**failure**:

**gerrit**:

**Verified**: -1