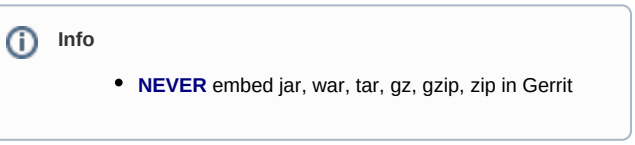
Configuring Gerrit

- Overview
- Tools
- SSH
- HTTP Workaround for Gerrit inside a Corporate firewall/proxy blocking SSH
- A couple of other things specific to Gerrit
  - HTTP Sequence of events
  - SSH Sequence of events
- Things to keep in your radar
  - Merge Conflict
  - Jenkins Intermittent Failure
- Submitting a draft feature
- Running a Command within Gerrit
  - Run-clm
  - Run-Sonar
  - Recheck
  - Remerge
  - Please Release

blocked URL



Overview

Gerrit is an Open Source web-based collaborative Code Review tool that integrates with Git.
For all ONAP projects, Gerrit is available at https://gerrit.onap.org and documentation at https://gerrit.onap.org/r/Documentation/index.html
Gerrit offers an extensive search capability by using a query syntax documented at https://gerrit.onap.org/r/Documentation/user-search.html
To get started, you can watch this 14 minutes video that was done for OPNFV project.

To understand deeper Gerrit UI, this is a good source of info https://gerrit-review.googlesource.com/Documentation/user-review-ui.html
Log into Gerrit using your Linux Foundation ID (LFID).

Anyone who has a valid Linux Foundation ID can submit code as a contributor. You simply need to login into Gerrit using your Linux Foundation ID.
The committer role requires specific permissions per project setup by Linux Foundation. Do not contact directly Linux Foundation's Help Desk to ask for Committer privileges but rather work with the Release Manager who will make it happen through the TSC.

Project committers group are setup in the form of **onap-gerrit-${PROJECT}-committers**. Connect to https://identity.linuxfoundation.org to see which group you are part of.
Committers are nominated according to Configuring Gerrit.

Tools

Whatever OS your are using (Mac, Linux, PC) the following tools must be installed and configured:

1. git client : to perform all the Git task (clone, pull, branch, checkout,...)
2. git-review : to submit your code into Gerrit
3. SSH client : to connect securely to Gerrit server

SSH

Most users use SSH to authenticate with remote servers. To perform such authentication with Gerrit, your have to provide your SSH Public key to Gerrit (User Account->Settings->SSH Public keys).
In case you can't use SSH (because network do not allow SSH on port 29418), you can use HTTPS.

HTTP Workaround for Gerrit inside a Corporate firewall/proxy blocking SSH

A couple of other things specific to Gerrit

- **Gerrit does not allow you push directly to your branch**. If you're not using the git-review plugin then to push a change against your branch it will be as follows:

Pushing Code into Gerrit

```
# in this case
BRANCH=master
git push origin HEAD:refs/for/$BRANCH
```

- Your change will go to Gerrit to be reviewed. It will not be merged onto the branch until someone with committer rights gives it a Code-Review +2. Normally there are verification jobs setup in Jenkins that would vote on the Verified field, but as your project(s) don't just yet (oparent being the exception as they pushed a verify job this afternoon) a committer will also have to flag Verified +1. Once both fields are at max value, then a committer will have the ability to **Submit** the code. It will not be merged until the final **Submit** has occurred.

Once the code is submitted, Gerrit moves the code from the **Open** tab https://gerrit.onap.org/r/#/q/status:open to the **Merged** tabs https://gerrit.onap.org/r/#/q/status:merged.

- Your code import cannot be a historical import. That is, you can't be bringing history from an external SCM tool into a Gerrit repo under Linux Foundation control. This is a policy Linux Foundation had in place for a very long time and is non-negotiable. This means your import will be a squash commit of any code coming in.

HTTP Sequence of events

The HTTPS method does not use your LFID password; you need to have Gerrit generate a password for you (User Account->Settings->HTTPS Password->Generate Password). (The details of managing "git" credentials can be found here: https://git-scm.com/book/en/v2/Git-Tools-Credential-Storage.)

Clone using https syntax

```
export LFID=<YOUR_LFID_HERE>
export REPO=common-services-external-system-registration

# do this once
git config --global credential.helper store

git clone https://${LFID}@gerrit.onap.org/r/a/${REPO}.git
Cloning into '${REPO}'...
Password for 'https://${LFID}@gerrit.onap.org':
  # type or paste your generated password above.
  # this prompt will only appear once, after that, it will store it in ~/.git-credentials
  # and will re-use it for any future repos

cd ${REPO}
# acquire the commit hook - do this once
curl -Lo ./.git/hooks/commit-msg https://gerrit.onap.org/r/tools/hooks/commit-msg
chmod +x ./.git/hooks/commit-msg
# configure remote - do this once
git remote add gerrit https://${LFID}@gerrit.onap.org/r/a/${REPO}

# create your code commit
cp ${codeblob} ./
git add .
git commit -asm 'Initial code import'
git push origin HEAD:refs/for/master

# as an alternative to the above "git push" line:
git review
```

A couple of things to note here:

The URL will be /r/a/${REPO}.git (the .git is optional) /r == the web url that Gerrit lives on.
/a == authenticated https.
Without /a it will try to do an anonymous http connection and it will fail for pushes, at least when we open the repos to public access.

The commit has a -s which gives you the 'Signed-off-by: Name <email>' footer in your commit message. Your 'Name <email>' portion must match what Gerrit has registered. And it's case sensitive. (Commit message example)

If you do not have the Gerrit commit hook installed you'll get an error when you push telling you how to get it. Once you've obtained it you'll want to run the following operation before trying to push again:

Amending commits

```
git commit --amend
```

Just resave your commit message. After you do that if you do a 'git log' you should notice that a 'Change-Id:' line was added above your 'Signed-off-by' footer. This Change-Id is required by Gerrit.

SSH Sequence of events

Clone using ssh syntax

```
export LFID=<YOUR_LFID_HERE>
export REPO=common-services-external-system-registration

git clone ssh://${LFID}@gerrit.onap.org:29418/${REPO}.git

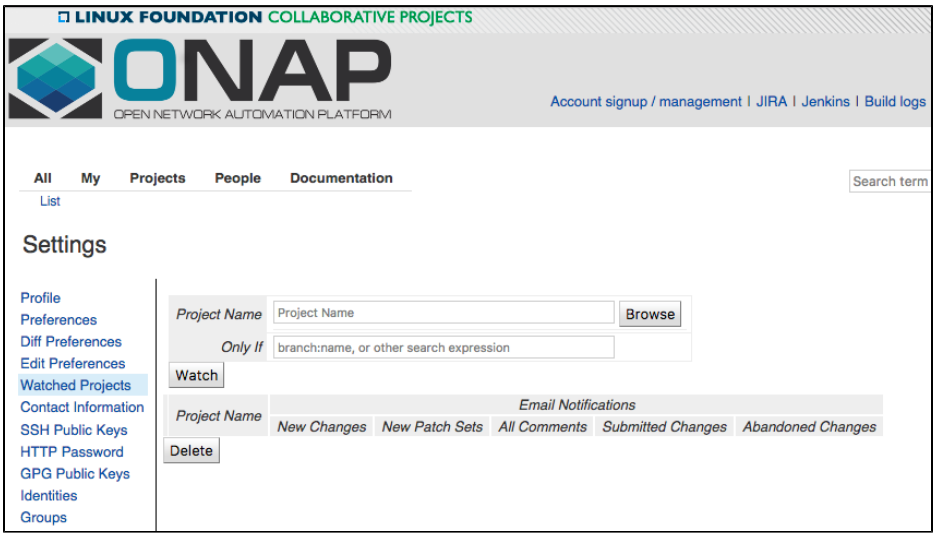
cd ${REPO}
# acquire the commit hook
curl -Lo .git/hooks/commit-msg https://gerrit.onap.org/r/tools/hooks/commit-msg
chmod +x .git/hooks/commit-msg
# creat your code commit
cp ${codeblob} ./
git add .
git commit -asm 'Initial code import'
git push origin HEAD:refs/for/master
```

You'll note that this is essentially the same. The primary difference is that you won't be getting prompted for a password as it only operates with SSH keys.

Committer

In order to avoid delays merging the code, it is expected that committers or contributors **review the code within the next 36 business hours** after the contributor/committer has submitted his code.
There are 2 ways for a Committer/contributor to be notified by email on the code to review:

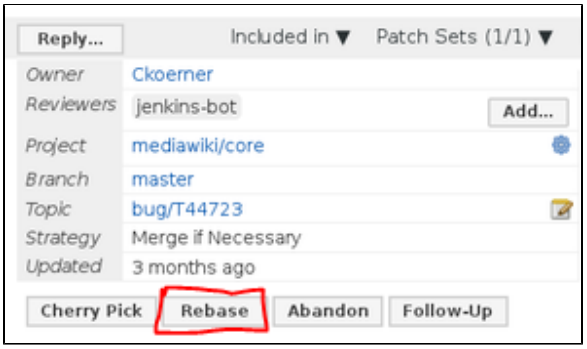
1. the contributor/committer specifically enters the committer/contributor name in the submission form.
2. any contributor/committer add their self to the list in gerrit
3. the contributor updates his Gerrit's settings to **Watch** his projects.



Things to keep in your radar

Merge Conflict

It may happen that you cannot merge into the upper branch. Gerrit will render this behavior by indicating in the Status column the message **Merge Conflict**.
You can also run the following query to display all the submissions in status **Merge Conflict.**



To address the **Merge Conflict** issue 3 solution scenarios are possible:

1. A change that has already been merged is in conflict with the current change, try to rebase the change by clicking the "Rebase" button in Gerrit UI.
2. A simple rebase can't fix the problem because it is fundamentally unsolvable without making actual changes to the submission. The solution to this sort of error is to try to do the merge locally and find out what the actual problem is and then re-submit an update.
3. Abandoning and bringing in a new patch. This is of a last resort because you can't figure out what the real problem is.

Jenkins Intermittent Failure

For some yet unknown reason it may happen that Jenkins have intermittent failure and thus impact your build.
Look for typical message:

Jenkins error example

```
[ssh-agent] FATAL: Could not find a suitable ssh-agent provider
[ssh-agent] Diagnostic report
[ssh-agent]  Java/JNR ssh-agent
[ssh-agent] hudson.remoting.RequestAbortedException: java.io.IOException: Unexpected termination of the
channel
.............
FATAL: [ssh-agent] Unable to start agent
hudson.util.IOException2: [ssh-agent] Unable to start agent
.............
Finished: FAILURE
```

To address the issue just hit the reply button in Gerrit add keyword 'recheck' or 'reverify' (do not enter the ') to the submission that failed. This will automatically retrigger the **verify** job.
If the job was a merge job, then use the 'remerge' keyword. It will automatically retrigger the **merge** job.

blocked URL

Submitting a draft feature

As part of the Development Best Practices, it is suggested to commit code multiple times a day. One may argue on the value of submitting code multiple a day if the code you are submitting does not bring value (working functionality) to the community. Developing a complete feature may take days or weeks and until it is complete it may not bring value - but will if work is being shared. Consider submitting unfinished code as an opportunity to show work in progress and get early feedback. However, as we do not want to break the CI paradigm of "Don't break the Build" by committing+merging unfinished work, Gerrit has a way to show work in progress by using the "Draft Feature" capability - however you can commit outside of the draft as long as you mark the review as WIP or "do not merge".

We are making the assumption that developers are working in a feature branch (named story-1 for the sake of the example)

To submit draft code in Gerrit, from a feature local branch, enter from your terminal the following:

git review -D

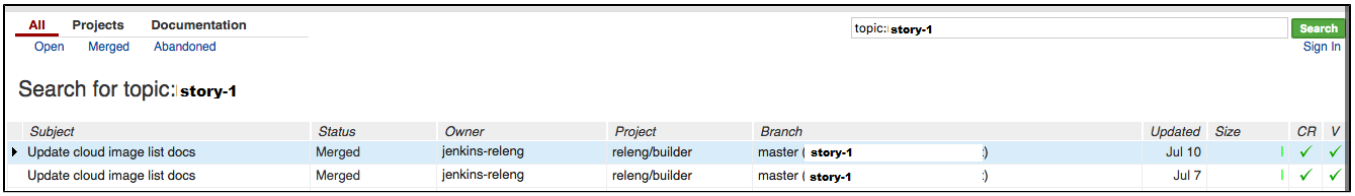
The name of the local branch (story-1) is used by Gerrit to set the topic.

You can perform the above command as often as you need until the developer has not completed his work. The CI build system is automatically triggered.

These commits CAN'T be +2 and merged in Master.

Once a developer has finished developing the functionality and need all these commits to be merged into Master, the developer will need to use the Gerrit UI and click for each commit the "Publish" button. The committers will then have 36 business hours to complete the review and merge the code into Master branch.

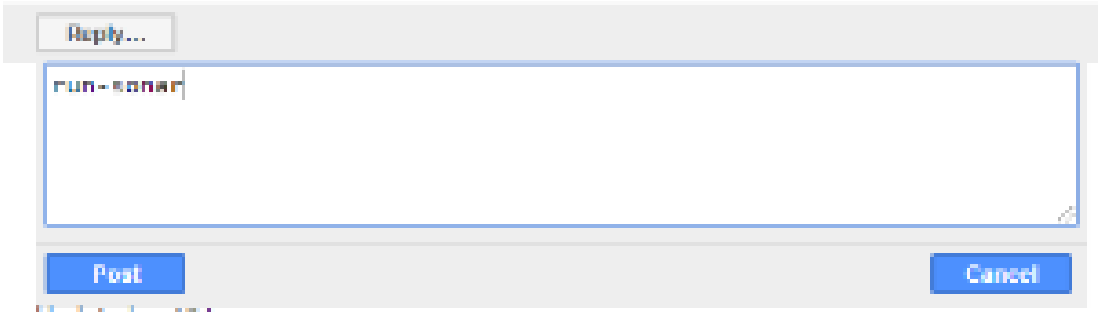
This Gerrit screenshoot illustrates 2 "Draft Feature" of topic: story-1



Running a Command within Gerrit

Gerrit provides capabilities to run some command on the tip of the branch.

Note: all the command mentioned below must be entered without the double quote ".



Run-clm

CLM jobs can be triggered on demand after posting the comment "run-clm" in your Gerrit change.

The CLM jobs are still scheduled to run every Saturday, this feature can be useful for debugging on demand.

Commenting "run-clm" in a gerrit that is not merged, will not trigger the CLM job based on that revision but will trigger the job based on the tip of the branch.

This job is designed to always run on the latest tip of the branch to avoid inconsistencies on the reports.

Run-Sonar

Sonar job can be triggered on demand after posting the comment "run-sonar" in your Gerrit change.

These jobs are scheduled to run every day, this is used to follow up closely on code coverage progress.

Recheck

In the case Jenkins is facing a failure or your need to re-run the Jenkins Verify job you can do so on demand by posting the comment "recheck" in your Gerrit change.

Remerge

In the case Jenkins is facing a failure or your need to re-run the Jenkins Merge job you can do so on demand by posting the comment "remerge" in your Gerrit change.

Please Release

Release jobs can be triggered on demand after posting the comment "please release" in your Gerrit change.

Release jobs can be seen in Jenkins by looking at job name containing this string "release-version-java-daily". The log file file contains the keyword "autorelease-xyz" with xyz a random number.

Autoreleased job are available in Nexus at https://nexus.onap.org/content/repositories/autorelease-zyz

The link to the autoreleased job is a mandatory input for LF to release the binary into Nexus Release.

These jobs are scheduled to run every day.