Development Procedures and Policies

Links

SDN-C Development Environment Setup#CDevelopmentEnvironmentSetup-Fixnochange-idHowtoFixaCommitwhichDoesNothaveaChange-Id

Configure git for http only clients - as in developers behind a firewall proxy that blocks SSH (in this case use git push origin HEAD:refs/for/master instead of git review - Configuring Gerrit

Prerequisite: Set Up

The actions described in this section depend on having the requisite set of tools and settings on your development machine. See Setting Up Your Development Environment.

Note: replace openecomp.org with onap.org in all screen captures - reason: https will authenticate against the actual onap.org domain but ssh will redirect from either.

Reference: Gerrit Code Review for Git

ONAP uses Gerrit to automate the process of reviewing all code changes before they are committed to the Git repository. Here is a tutorial and reference on using Gerrit: https://gerrit.onap.org/r/Documentation/index.html.

Using the Command Line to Clone the Repository

1. Move to the folder at the root of your development directory. For example:
   `cd C:\Users\<user>\Documents\work`

2. Add a remote pointer to the Gerrit server that hosts your repository using either SSH or HTTPS. (Note: <REPONAME>s are given in various Development Guides.
   Using SSH:
   `git remote add origin ssh://USERNAME@gerrit.onap.org:29418/<REPONAME>`

   Using HTTPS:
   With HTTPS, first you will need the Gerrit HTTP-generated password for each HTTPS operation with Gerrit/Git.

   `git remote add origin https://USERNAME@gerrit.onap.org/r/a/<REPONAME>`

3. Now clone the remote repository. Since we clone in the current folder, it will create a subfolder with the remote copy.
   Using SSH:
   `git clone ssh://USERNAME@gerrit.onap.org:29418/<REPONAME>`

   Using HTTPS:
   With HTTPS, first you will need the Gerrit HTTP-generated password for each HTTPS operation with Gerrit/Git.

   `git clone https://USERNAME@gerrit.onap.org/r/a/<REPONAME>`

Using IntelliJ to Clone From, and Commit To, Gerrit/Git

Visit https://gerrit.onap.org and log in, using your Linux Foundation identity. Here, you will be able to view the names of Projects to which you have access.

On top of the project listing, the web site provides you the command to clone the whole project, such as:

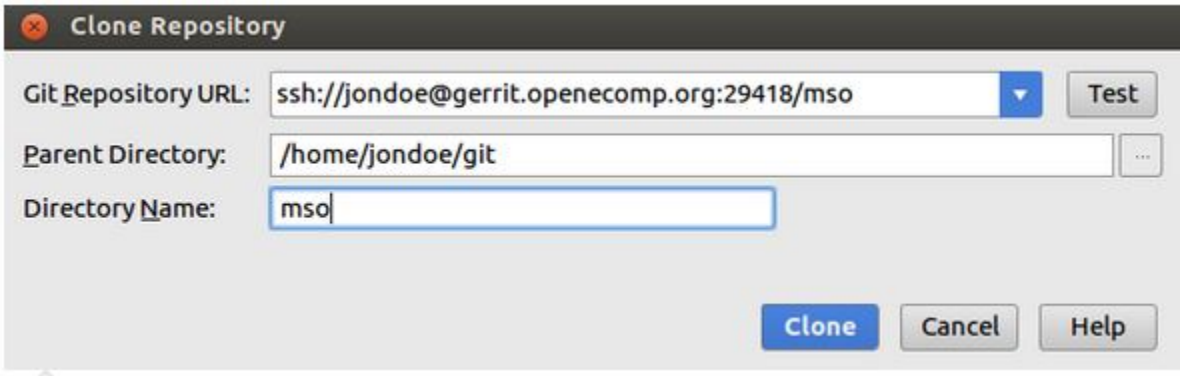
The project can be downloaded in either of two ways, via HTTP url or SSH url.

IntelliJ and SSH URL (Recommended)

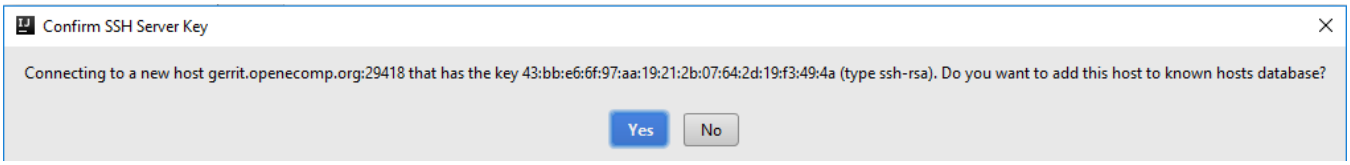
To download the project using an SSH URL, you need to have already added your SSH key into the Gerrit website.

Having added your key, visit https://gerrit.onap.org/r/#/admin/projects/ and select the project you are working on. The Gerrit site will show the full URL for the project:

Next, within IntelliJ, select **File** -> **New** -> **Project from Version Control** -> **Git** to open the **Clone Repository** window. Enter the SSH URL from above, including the project name (**mso** in this example), and chose your local **Parent Directory**. IntelliJ will fill in the **Directory Name** for you. Click **Clone** to clone the project.



For the first downloading, a window will appear to authenticate the host.Click **Yes** to continue.

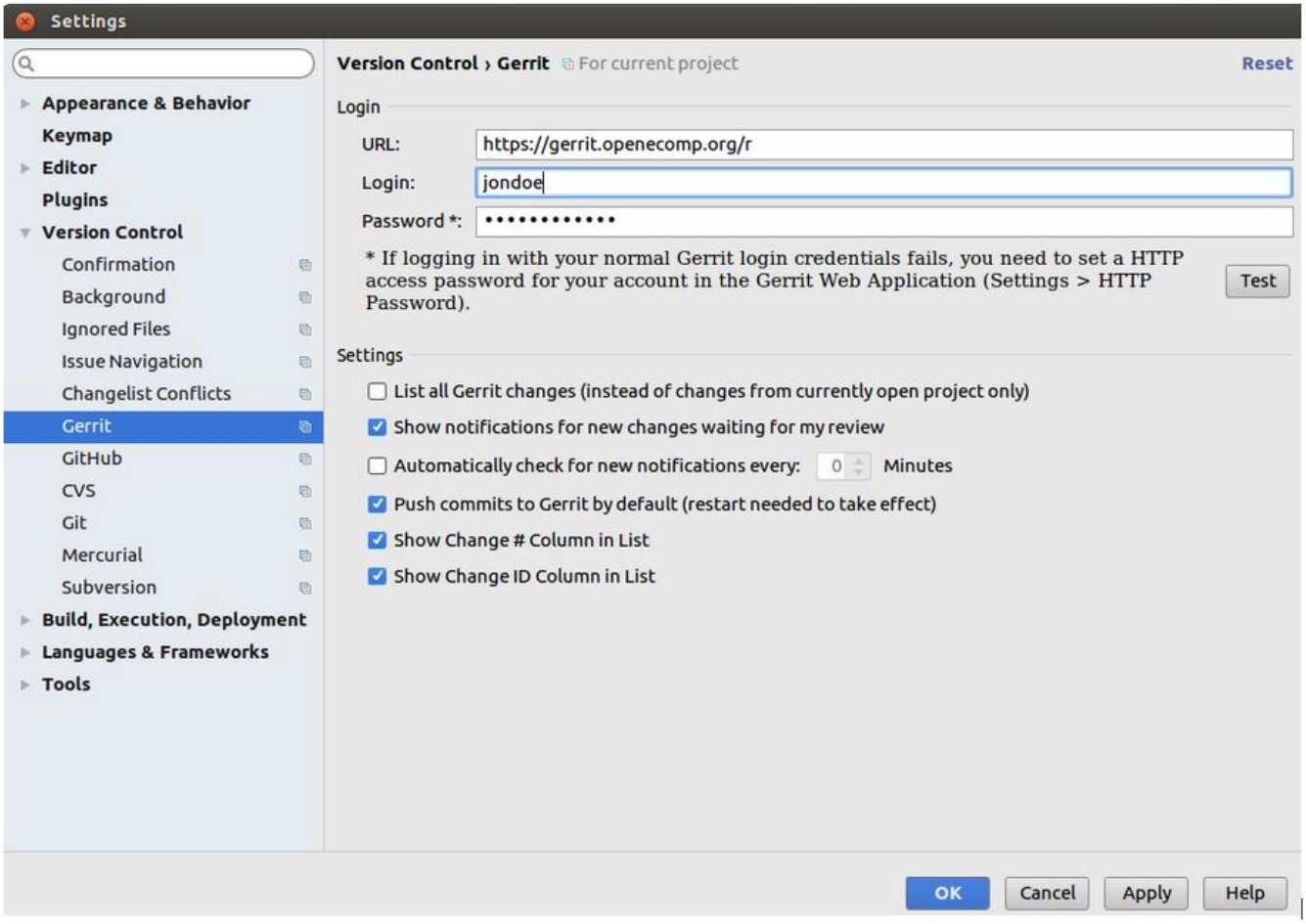


IntelliJ and HTTP URL (Alternate Method)

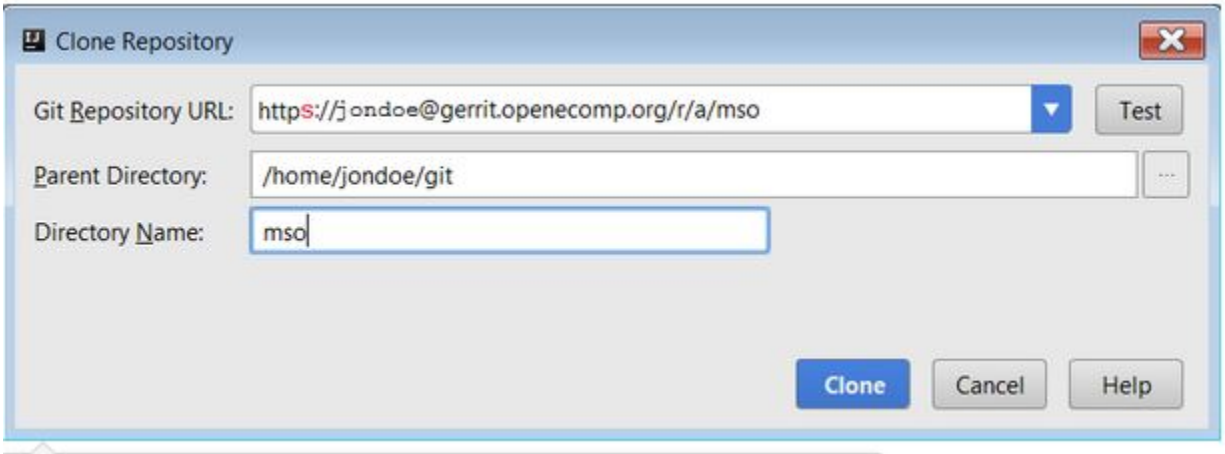
Open IntelliJ and click **File** -> **Settings** -> **Version Control** -> **Gerrit**. In the resulting dialog box, enter the Gerrit URL as shown above, along with your Linux Foundation username and password. To test whether the connection is good, you can click the **Test** button:

> **https: Required**
>
> The URL must use **https:**, not **http:**.

Once you have entered this information, click **OK**:



Then open **File** -> **New** -> **Project from Version Control** -> **Git** to open **Clone Repository** window. Enter the http url you have found above, but with **https** not with **http**, chose your local **Parent Directory** and **Directory Name** and click **Clone** to clone the project.

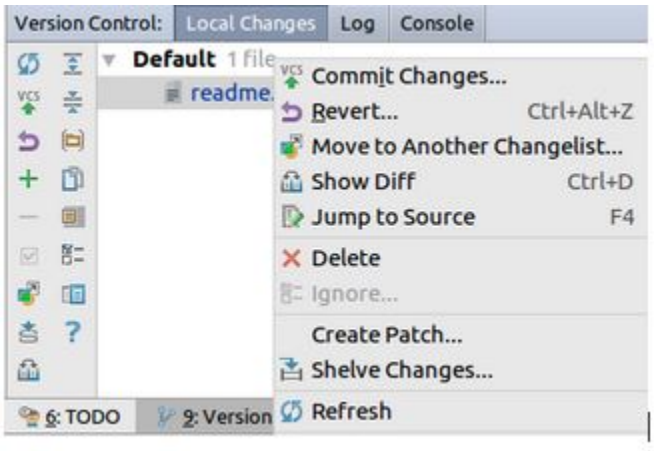


A window will pop up to input the user name and password. The user name is the username of your Linux Foundation account, and the password is the Gerrit HTTPS password generated during setup.

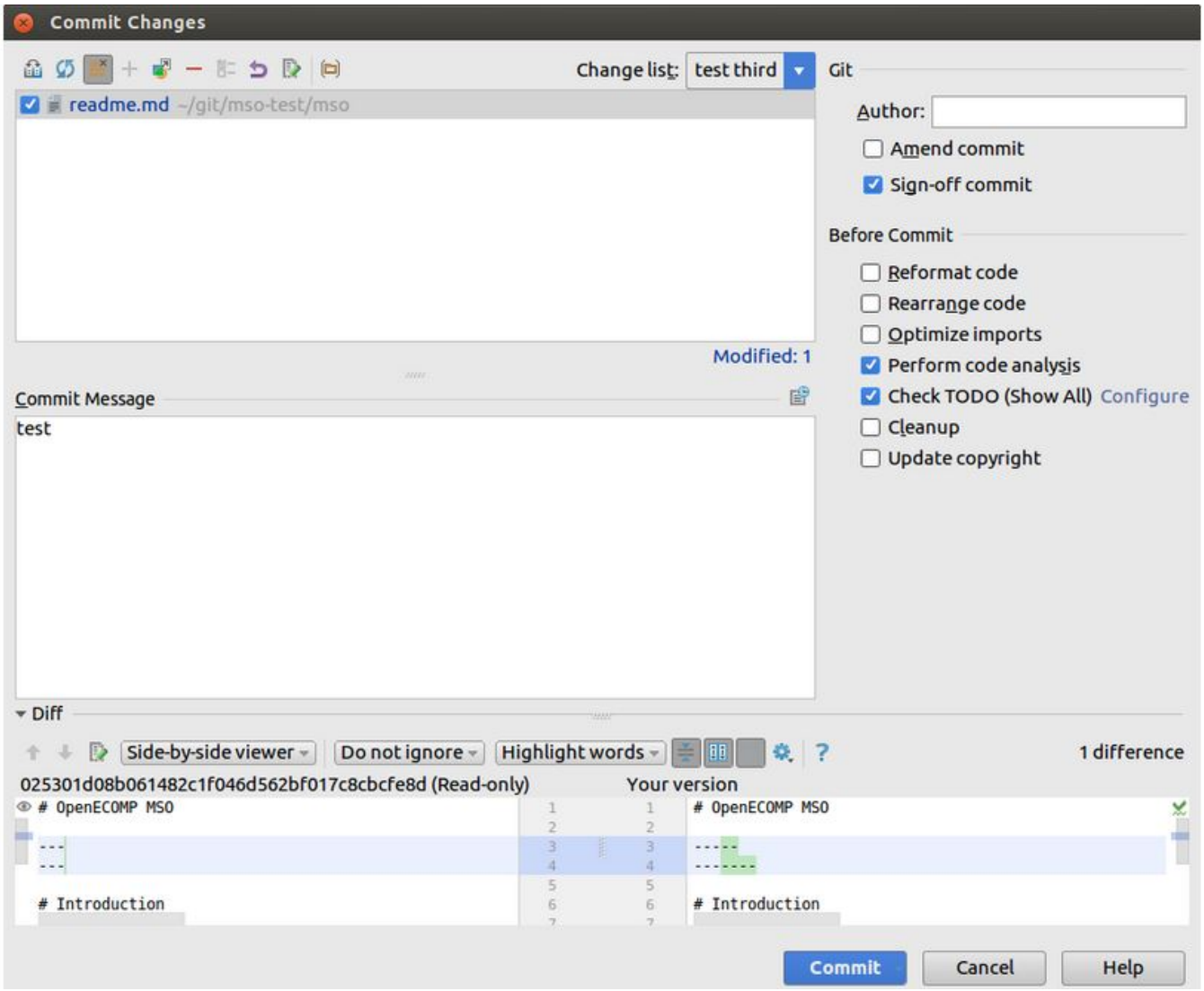
Commit Changes Locally

Once you have made changes in the code, before you push the changes, you should commit the code locally.

To do that, go to the **Version Control** window, under **Local Changes**, right click the changed files and click **Commit Changes...**.



In the Commit Changes window, enter the **Commit Message**, select **Sign-off commit** option and click **Commit** button.



Pushing Changes for Review Using a Command Line (optional)

Here are the Git commands to add some files and make a commit with an associated message:

```
git add somefiles

git commit -m "My first Awesome commit"
```

Note that this is just an example and the commit message should be more explicit than that.

Now that everything is ready, you can sign off your commit:

```
git review -s
```

if asked for a remote gerrit run

git config --global gitreview.remote origin

You may be prompted for your Linux Foundation account password. To verify that the commit worked, this command will show you the commit message and the sign off entry:

See the exact commit message format at Commit Messages

```
git commit -s --amend
```

Automatic signature mode

You can automatically add the signature for all your commits by setting the git configuration:

```
git config --global format.signoff true
```

Verify you are setup with the correct account

vi ~/.gitconfig

[gitreview]

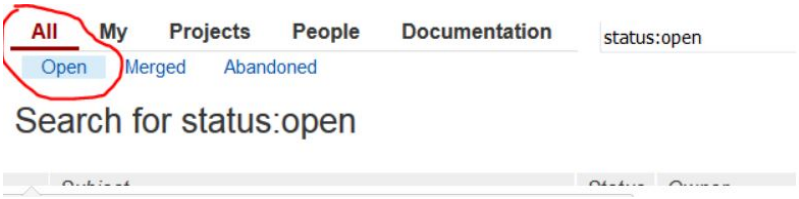
username = youruser

Eventually, we can push the change to the gerrit server:

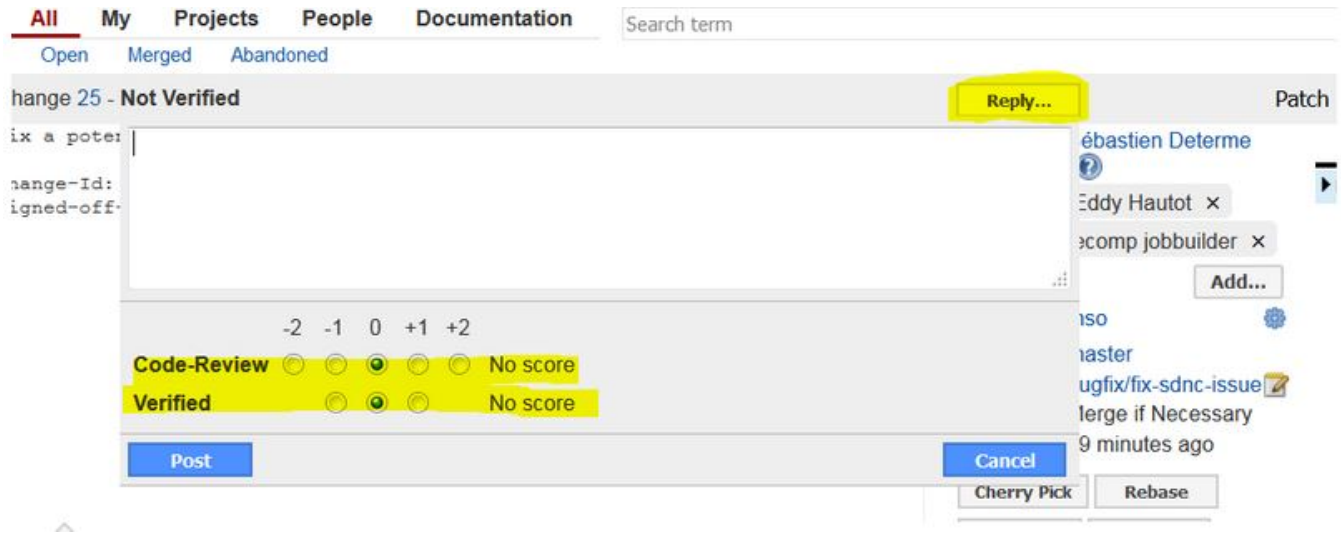
```
git review
```

Committer/Contributor Actions

The code should now appear in the gerrit web interface. A contributor or committer can review it with a -1/0/+1 and mark it as verified with test results if available. It needs a committer to approve it with a +2 rating, and to also move the verifed flag to +1. The committer does this by visiting the gerrit.onap.org site and logging in.



The committer may take any of several actions, such as clicking on the "Reply" button, adding reviewers, adding a review comment, and moving the flags to +2 and +1



Once a committer/contributor approves it, the code can be merged to the master branch.

Then go to the **Terminal** window, use command **apt-get install git-review** to install git-review plugin in the machine, if not already installed.

```
root@ubuntu:~# apt-get install git-review
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  git-review
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 18.1 kB of archives.
After this operation, 120 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu/ trusty/universe git-review all 1.23-1 [18.1 kB]
Fetched 18.1 kB in 0s (59.2 kB/s)
Selecting previously unselected package git-review.
(Reading database ... 204588 files and directories currently installed.)
Preparing to unpack .../git-review_1.23-1_all.deb ...
Unpacking git-review (1.23-1) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up git-review (1.23-1) ...
root@ubuntu:~#
```

Once installed, under root location of the project enter command **git config --list**, make sure the email address listed is **exactly** the same as the one you used in your Linux Foundation account.

```
root@ubuntu:~/git/mso-test/mso> git config --list
http.sslverify=false
user.name=jondoe
user.email=jondoe@intl.att.com
core.autocrlf=input
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=ssh://jondoe@gerrit.openecomp.org:29418/mso
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
remote.gerrit.url=ssh://jondoe@gerrit.openecomp.org:29418/mso.git
remote.gerrit.fetch=+refs/heads/*:refs/remotes/gerrit/*
```

If your address is not present or not the same as the one defined in Linux Foundation account, enter command **git config --global user.email "jondoe@somewhere.com"**.

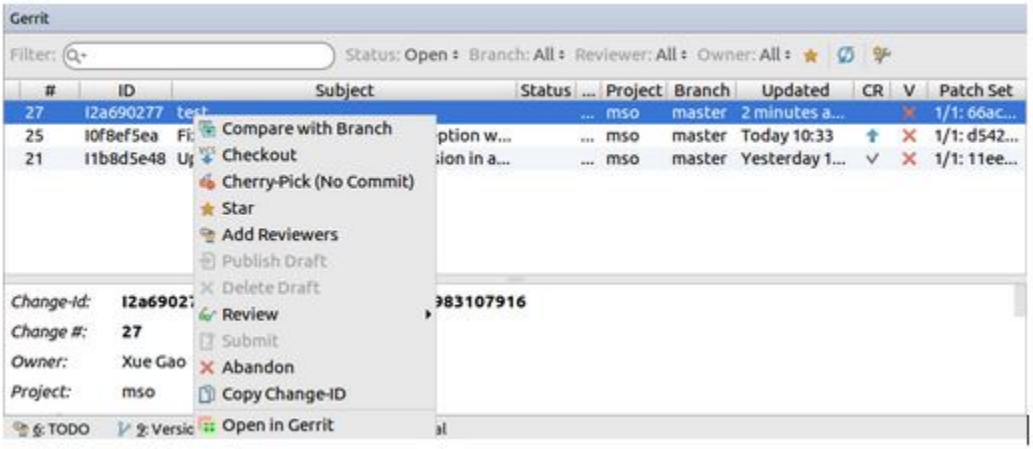
Then enter command **git commit -s --amend** to show the commit message and the sign off entry, as shown below. Make sure the email address is the **exactly** the same as the email you used in the Linux Foundation account. If not, update the address.

```
test

Change-Id: I2a6902770c9f0172c5fd139e6a2fa29983107916
Signed-off-by: jondoe <jondow@intl.att.com>

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Your branch is ahead of 'origin/master' by 2 commits.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#       modified:   readme.md
#
```

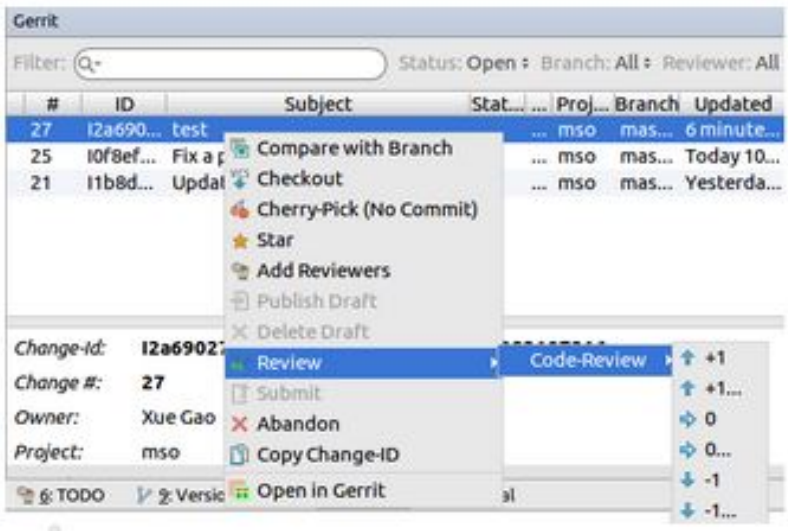
Then enter command **git review** to upload the changed code into the Linux Foundation for reviewing.

```
root@ubuntu:~/git-lf/mso-lf$ git review
remote: Processing changes: new: 1, refs: 1, done
remote:
remote: New Changes:
remote:   http://gerrit.openecomp.org/r/31 well formated commit message for tests
remote:
To ssh://jondoe@gerrit.openecomp.org:29418/mso.git
 * [new branch]      HEAD -> refs/publish/master
root@ubuntu:~/git-lf/mso-lf$
```

Go to **Gerrit** window and **refresh**, the committed changes will be shown in the window. Right click the commit, you can find the tab to **Add Reviewers** for reviewing or **Abandon** the changes if you don't want it any more.



If you have been selected as the reviewer of the changes. You can right click the corresponding commit and click **Review**, **Code-Review** to give your score and comments for the change.



JBL Templates for on-demand Gerrit recheck or remerge

~~https://lists.onap.org/pipermail/onap-discuss/2018-January/007781.html~~

https://lists.onap.org/g/onap-discuss/message/7773?p=,,,20,0,0,0::relevance,,recheck,20,2,0,22459587

Thanks Alexis de Talhouët - you can type "recheck" or "remerge" in the comment to kick in another JobBuilder run if required

A rebase will also kick in a build

see the ci-management ONAP project

https://github.com/onap/ci-management/blob/master/jjb/global-macros.yaml#L343

Failed Commit / Resubmit

If issues are found, both contributors and committers can amend the review (use the top-left download link on the review screen to get the direct clone commands that would download the reviewed code in a local repository)

Similarly to the above steps, do the needed modifications, and push the changes (again after signing them) using git review

Note that you can also work on local branch, Gerrit will automatically use your local branch name as a Topic for the review, allowing you to share branches between team members.

Example Session

ssh-add onap_rsa

git clone ssh://username@gerrit.openecomp.org:29418/portal

or via https (note openecomp.org now redirects to onap.org)

obrienbiometrics:onap4 michaelobrien$ git clone https://gerrit.onap.org/r/a/mso

Cloning into 'mso'...

Username for 'https://gerrit.onap.org': michaelobrien

Password for 'https://michaelobrien@gerrit.onap.org':

remote: Total 4596 (delta 0), reused 4596 (delta 0)

Receiving objects: 100% (4596/4596), 2.94 MiB | 1.68 MiB/s, done.

Resolving deltas: 100% (1985/1985), done.

Magic Words

Sometimes automatic verification/build/deploy doesn't happen as expected. In that case, use the gerrit Reply feature to insert the magic word into the message body.

Action Desired	Magic Words	Result
trigger the verify job	recheck	re-runs the Jenkins Job Builder (JJB) on an unmerged set of code
trigger check of self release	recheck|reverify	will trigger the "{project-name}-release-verify-{stream}" job. Will make sure the release file contains the needed information and that the candidate exists. (NOTE: possible confusion here over syntax...is that bar (|) to be included or can you use either of these words in the Reply?) Ref: Self Releases Workflow (Nexus2)
trigger the helm verify job	verify-helm	re-runs the JJB verify-helm job used in the OOM project
trigger the merge build Jenkin job	remerge	re-runs the JJB deploy on a merged set of code
trigger self release of artifact	remerge	runs the "{project-name}-release-merge-{stream}" job to push the release and tag the repo. Ref: Self Releases Workflow (Nexus2)
trigger daily release job	please release	re-runs the JJB daily release build
	stage-release	re-runs the JJB daily maven-stage build (replaces "please release")

trigger CLM jobs	run-clm	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. Source: Configuring Gerrit
trigger Sonar job	run-sonar	this is used to follow up closely on code coverage progress.
rebase	N/A	There is no magic word for this desired action. Instead, hit the button in the review page.
trigger OOM CD deployment / healthcheck for component under review	run-helm-deploy	20181122 - in progress **TSC-25** - Getting issue details... STATUS meetings https://lists.onap.org/g/onap-discuss/topic/cd_task_force_tsc_25_meetings/29001640?p=,,,20,0,0,0::recentpostdate%2Fsticky,,,20,2,0,29001640 will run an OOM deploy of the component and any dependent component on request (before a merge) for example if the current helm-verify job is +1 - a full deploy of the component via --set <pod>.enabled=true will be done, healthcheck run and reported back as a +1 if we get a pass for the pod Example of what the jjb job will run when deploying pomba - requires dmaap to function. `sudo helm deploy onap local/onap --namespace onap -f onap/resources/environments/disable-allcharts.yaml --set pomba.enabled=true --set robot.enabled=true --set dmaap.enabled=true`
Re-run CI Deployment test	oom_redeploy	Re-runs the remote CI oom deployment job in Orange Lab
Create Jenkins job in Sandbox	jjb-deploy <pattern>	Posting this on a ci-management change creates jobs matching the pattern (for example "myproject-*") to the Jenkins sandbox, https://jenkins.onap.org/sandbox/. See https://docs.releng.linuxfoundation.org/en/latest/jenkins-sandbox.html.

Links

ONAP Development