Gerrit Code Review

Version 1.18-SNAPSHOT

2018-12-18

Table of Contents

Table 1. History

Date	Author	Detail
2018-12-18	bcouetil	Moved Gerrit plugin configuration from Jenkins to Gerrit page + Enhanced HTML CSS + Cropped some images + Improved PlantUML skin
2018-12-11	bcouetil	- Added reveal plugins and background - Fixed reveal css following change in structure in asciidoc-reveal master (from previous version : 1.1.3) - Implemented Zenika layout in HTML and PDF - Reported back reveal-js enhancements
2018-09-19	bcouetil	- Sample asciidoctor maven project published on Github - Github & LinkedIn links - Sample project tree - new images + resizing and positioning
2018-08-29	bcouetil	Asciidoc HTML look & feel changes
2018-08-23	bcouetil	Initial commit

1. History

The initial development of Gerrit started when a code review system for **Android** was requested. Because many Google developers were involved in the development of Android, the new system needed to have a feature set similar to the Google internal review system **Mondrian**. For this purpose, **Rietveld** was started. Because the development of Rietveld was not fast enough, Rietveld was forked and developed separately, then as **Gerrit**.

The name originates from that of the Dutch architect Gerrit Rietveld.

2. User documentation

Official user documentation is here.

3. Installation

With the openfrontier/gerrit image, you can start a ready-to-use dockerized server with persistant data in a local folder, SMTP and replication configuration. More information/examples here.

Using docker

```
$ docker run -d --name gerrit-of -p 8080:8080 -p 29418:29418 \
-e AUTH_TYPE=DEVELOPMENT_BECOME_ANY_ACCOUNT \
-e WEBURL=http://my.public.url:8080 \
-e SMTP_SERVER=my.smtp.server \
-e USER_NAME="Gerrit CodeReview" \
-e USER_EMAIL=gerrit@my-provider.com \
-v ~/gerrit_volume:/var/gerrit/review_site \
-e GERRIT_INIT_ARGS="--install-plugin=analytics" \
-e GERRIT_INIT_ARGS="--install-plugin=replication" \
-e REPLICATION_REMOTES=gitlab \
-e GITLAB_REMOTE=https://oauth2:ACCESS_TOKEN@my.gitlab.server:MY_GROUP/${name}.git \
-e GITLAB_REPLICATE_ON_STARTUP=true \
docker.io/openfrontier/gerrit
```

4. Configuration

You can switch from new UI (PolyGerrit) to old UI with the link at the bottom of any page, and you can switch back to PolyGerrit by adding `?polygerrit=1` in the URL.

4.1. Initialization

Updated with new UI

- Connect to Gerrit homepage
  - First person to connect is the administrator
  - Without LDAP, other users are added at first connection

So, as the administrator :

- Skip the plugin configuration
- Click on the image in the top right corner → **[Settings]**, and set

  Full name

  Changes per page = 100 rows

  Date/time format

  ☑ Set new changes to "work in progress"

  - Click **[Save changes]**

  Ignore Whitespace = All

  - Click **[Save changes]**

  New email address = your@mail.com

  - reload page
  - select your mail in list
  - delete email@example.com
  - Click **[Save changes]**

  Click **[Generate New Password]**

  - save it for later

4.1.1. Permissions

In further sections, actions are for old UI

Jenkins user push

- Click on **[People]** → **[List Groups]** → **[Non-interactive Users]**
- Add Jenkins' technical account in the list

Deleting tags

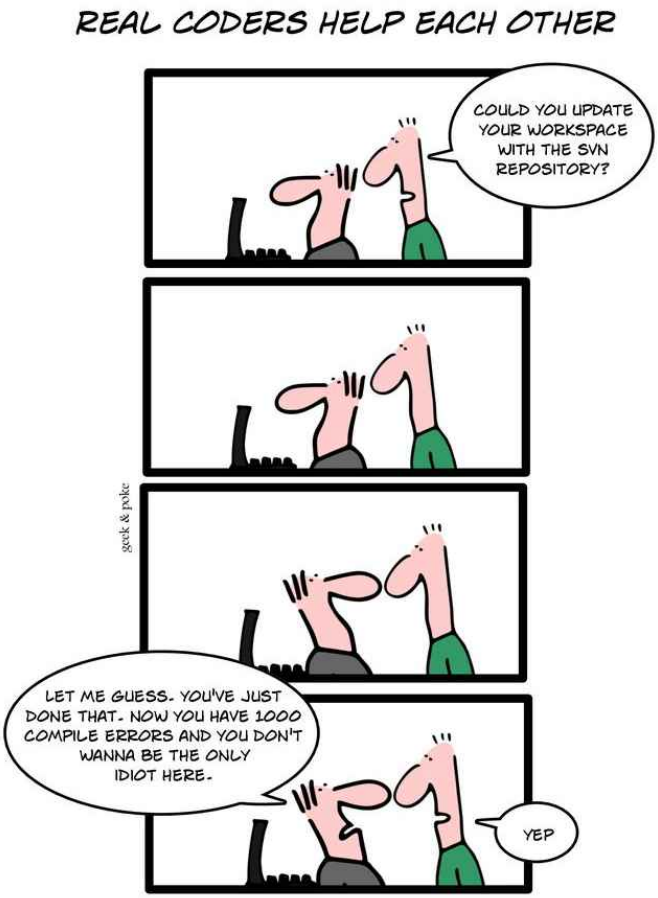
- Click on [**Projects**] → [**List**] → [**All-projects**] → section [**Access**] → [**Edit**]
- Under `Reference: refs/tags/*`
  - Click on [**Add Permission...**] and select [**Push**]
  - Select group [**Administrator**] and click [**Force Push**]
  - Save Changes

Now you can delete tags from your projects, for ex :

```
git push --force --delete origin cg-wm-1.17.6
```

Allow (gitweb) access for everyone

- Click on [**Projects**] → [**List**] → [**All-projects**] → section [**Access**] → [**Edit**]
- Under `Reference: refs/meta/config`
  - Under `Read`
    - Click on [**Add Group**]
    - Enter `Registered Users`
    - Click [**Add**]
    - Save Changes



4.1.2. Verified status

- Click on **[Projects]** → **[List]** → **[All-Projects]** → section **[General]** → **[Edit Config]**
- Add this

```
[label "Verified"]
   function = MaxWithBlock
   value = -1 Fails
   value =  0 No score
   value = +1 Verified
```

- Click on **[Save]**, then **[Close]**
- Click on **[Publish Edit]**, then **[Publish]**, **[Code-Review+2]**, **[Submit]**
- Click on **[Projects]** → **[List]** → **[All-Projects]** → **[Access]** → **[Edit]**
- Under **[Reference: refs/heads/*]**
  - Click on **[Add Permission...]** and select **[Label Verified]**
  - Select group **[Administrator]**
  - Select group **[Non-Interactive Users]**
  - Save Changes

4.1.3. Fast Forward

By default, when projet submissions are not fast forward, final submitting a change will create a merge commit. The history is potentially doubled.

- Click on **[Projects]** → **[List]** → **[All-Projects]** → **[General]**
- Under **[Submit Type]**, select **[Rebase if Necessary]**

Figure 1. Life without code review

4.2. User preferences

Click on [**YourName**] → [**Settings**] → [**Diff Preferences**] and set **columns** = `120` (you will probably have to paste it due to a GUI bug)

4.3. Project creation

4.3.1. Old UI

Create your GIT project by clicking on [**Projects**] → [**Create New Project**]

- Project Name = `your-project-name`
- Rights Inherit From = `All-Projects`
- Check that it has inherited correctly "Rebase if necessary", else change and save

4.3.2. PolyGerrit

Create your GIT project by clicking on [**Browse**] → [**Repositories**] → [**Create New**]

- Repository Name = `your-project-name`
- Rights Inherit From = `All-Projects`
- Owner = `the-code-owner`
- Click [**Create**]
- Check that it has inherited correctly "Rebase if necessary", else change and click [**SAVE CHANGES**]

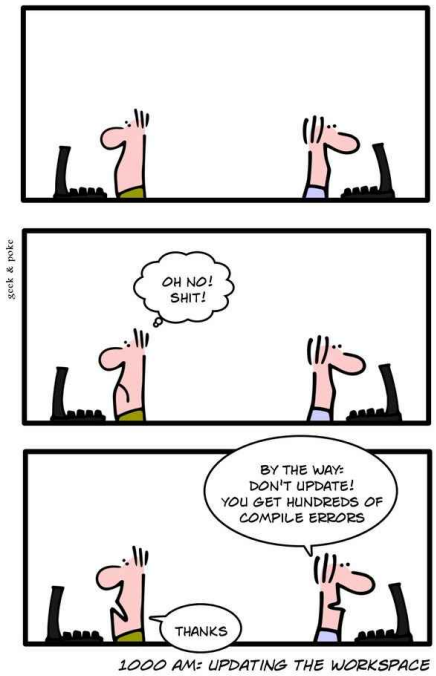
4.3.3. Project git address/URL

To get the repo address of your project under Gerrit :

- Navigate to [**Projects**] → [**List**]
- in front of [**your-project**], click on (gitweb)
- Take the .git text next to **URL**

Example : https://my-url.com/gerrit/my-project.git

If you don't have access to Gitweb interface ("Not Found"), ask your admin to do Allow (gitweb) access for everyone.



4.3.4. Users groups creation

For each project, create a reviewer list and a validator list.

- Go to **[Projects]** → **[Create New Group]**
- Reviewers list
  - Give a name, for example **[dge-reviewers]**
  - Add every developers / primary reviewers on the project
  - Click on **[General]**
  - Description = Reviewers (+1)
  - Click **[Save Description]**
  - Check **[Make group visible to all registered users]**
  - Click **[Save Group Options]**
- Validators list
  - Give a name for example **[dge-validators]**
  - Add technical responsible and a backup
  - Click on **[General]**
  - Description = Validators (+2)
  - Click **[Save Description]**
  - check **[Make group visible to all registered users]**
  - Click **[Save Group Options]**

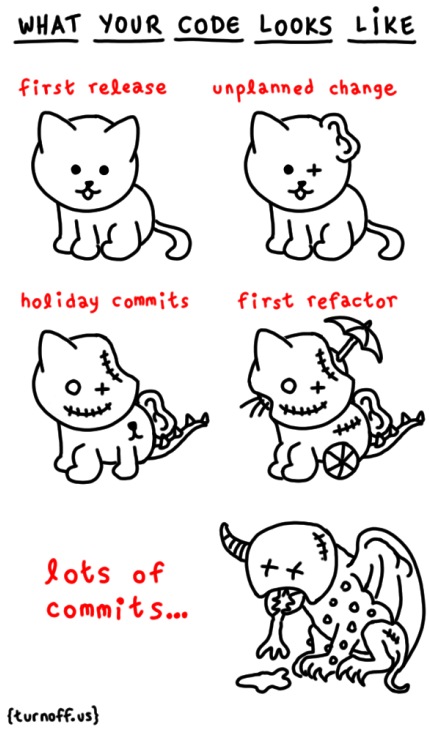


Figure 2. Repository without code review

5. Jenkins configuration

5.1. Plugin installation

- Go to **[Jenkins]** → **[Administration Jenkins]** → **[Gestion des plugins]**
- Install Gerrit Trigger

5.2. Plugin Configuration

- Create the console-master job if not already existing

  Create a new freestyle job.

  Name it console-master

  General

  ☑ **[Restreindre où le projet peut être exécuté]**

  - master

  In **[Build]** → **[Ajouter une étape au build]** → **[Exécuter un script shell]** → paste this and save :

```
ssh-keygen -y -f /root/.ssh/id_rsa > /root/.ssh/id_rsa.pub
ls -lart /root/.ssh/
more /root/.ssh/id_rsa.pub
```

- Add 1 executor on the master node
  - **[Home]** → **[État du lanceur de compilations]** → **[maître]** → **[Configurer]**
- Execute the console-master
- Keep track of what the execution gave for later Gerrit configuration, example :

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDKGER5oLwkNhcCYtTzmUQooA+1mdrjIGi84AVsOHyNpsMqFBhkpxfImvopvKlYiztXUA15dwwDsPWq1tUcy/4N
WqKnMTQA57xxxT2r8suF/DVlH6fNn8T73mGz9+kT77FXHuaMfmDTqrwPngUYQMm2Y9kTjGhIcH/jseq6jCUawITAOs/6EUbs7jtJ/S+jMb6Ed60S7S/n
R3IzQwVrXMiQjDdFsL8RWEBQ54T4cNia/HMI8MK7mEEF5K008g4Ru3Bidk+VSisPUYFPmNc/tE12RyAjvkcwWxrYqFEB5h6RlS0yWXAjCUzjv8T0ov4W
us+ZqNgqUMYtBBf+zQvQC1ub
```

- When finished, remove the executor from master node
- Create a local trigger server
  - **[Home]** → **[Administrer Jenkins]** → **[Gerrit Trigger]** → **[Add New Server]**
    - Gerrit Connection Setting
      - Name = local_server
      - Hostname = your.gerrit.server
      - Frontend URL = http://your.gerrit.server:8080/
      - SSH Port = 29418
      - Username = your-technical-user
      - SSH Keyfile = /root/.ssh/id_rsa
    - Gerrit Reporting Values

        - Verify = <vide>, 1, -1, -1, -1
        - Code Review = <vide>, 1, -1, -1, -1
    - Gerrit Verified Commandes
        - Started = vide
        - Successful =

```
gerrit review <CHANGE>,<PATCHSET> --message 'Build Successful (      ) <BUILDS_STATS>' --verified
<VERIFIED>
```

        - Failed =

```
gerrit review <CHANGE>,<PATCHSET> --message 'Build Failed (  _  ) <BUILDS_STATS>' --verified
<VERIFIED>
```

        - Unstable =

```
gerrit review <CHANGE>,<PATCHSET> --message 'Build Unstable (   °       °)  <BUILDS_STATS>' --verified
<VERIFIED>
```

        - Not Built =

```
gerrit review <CHANGE>,<PATCHSET> --message 'No Builds Executed (      ,)  <BUILDS_STATS>' --verified
<VERIFIED>
```

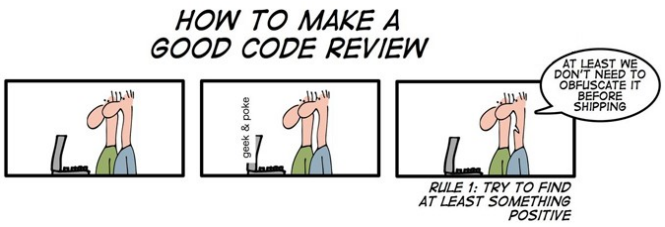
    - Save

On Gerrit

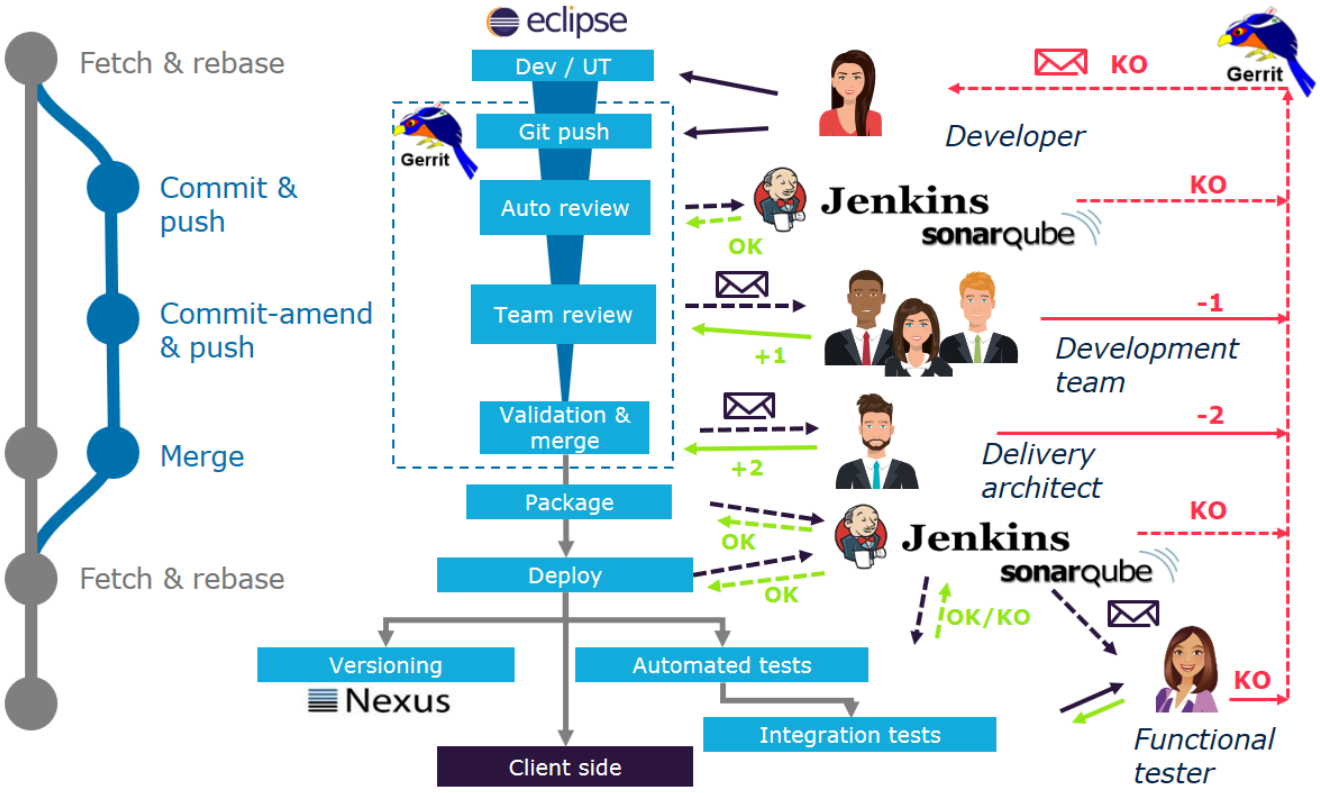
- Connect with the technical user (svc-fr-bpmfact / Bpm-fact0ry)
- Click on the user top right → **[Settings]** → **[SSH Public Keys]** → **[Add Key...]**
- Add the public key content from Jenkins server (the one asked to be kept track earlier), starting with `ssh-rsa`

On Jenkins

- Test the earlier configured connection of the trigger with **[Test Connection]** while editing local_server
- Restart jenkins with : https://bpmfactory.s2-eu.nvx.com/jenkins/safeRestart
- The Gerrit trigger should be up and running



6. Code review golden rules (using Gerrit)



These are rules to be followed for a smooth overall development process :

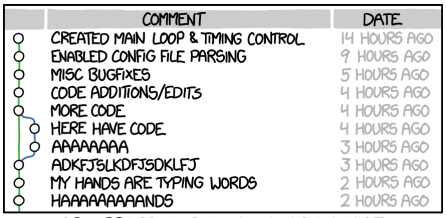
1. The team has to know that you are taking responsability of the current development task.
2. If not sure of what to achieve, confirm with task responsible
3. Target a complete realization in delays estimated by team leader. Alert on time shortage.
4. Update documentation along with code whenever it's needed.
5. Do not group functionnalities in commit, to avoid long run reviews.
    a. It is possible to handle multiple review in parallel.
6. Commit text has to be explicit, complete, and synthetic.
    a. Commit text must be one line for the sake of history and documentation readability (replace `:` with `()` and `-` with `+`). No limit to the lengh of the line.
    b. If the commit include documentation, set a first line commit text suitable for documentation. Put other information on other lines (they won't appear in documentation history)
7. Commit often, at least on tuesdays and thursdays (even on unfinished current task).
8. No `related changes` should appear on the change in Gerrit, or you did not handle multiple review properly.
9. Fixing Jenkins failures is always a top priority.
10. On `Cannot merge` Gerrit message, you have to pull/commit/push to rebase properly
11. When Jenkins give a `+1`, add the `reviewers` list as reviewers, this should add all reviewing people.
12. When added as a reviewer, try to give a review in the next half day, knowing that it blocks the process.
    a. You don't have to be an expert to do a review. At least try to spot pieces of code not well explained and missing javadoc. Try to imagine yourself as a future bug fixer who needs clean code to work properly.

b. If suitable, test the application or check the auto IT tests.

c. If any, check that the generated documention looks good in PDF.

d. Check that there is UT specifically testing the new/modified code.

13. When at least 25% of the team gave a `+1`, add the `validators` list of reviewers for a final `+2` review followed by a submit to the master branch.



AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

7. Troobleshootings

7.1. Submit replaced with Submit including parents

If the **[Submit]** button is replaced with **[Submit including parents]**, there is obviously a problem in the git tree. You have to rebase. You can ask the developer to do so in his IDE and push again. But you can also do it in the UI :

Click **[Rebase]**, select

☑ Change parent revision

leave the field blank

Click **[Rebase]**

Figure 3. Peer review is better for ego than peer refactoring

8. Appendix

8.1. Revision marks

Differences since last tag