Eclipse Configuration & Best Practices

Version 1.18-SNAPSHOT

2018-08-31

Table of Contents

Table 1. History

Date	Author	Detail
2018-08-31	bcouetil	Minor changes
2018-08-29	bcouetil	Asciidoc HTML look & feel changes
2018-08-24	bcouetil	Icones added for download + favicon added for webpage
2018-08-23	bcouetil	Initial commit

1. Installation

- Install the latest Eclipse JEE. This documentation is not guaranteed to work below Neon version.
- Install SonarLint plugin
  - in order to connect to its local SonarQube server and be warned of violations in Eclipse, in real time
- Install Enhanced Class Decompiler plugin
  - Enhanced Class Decompiler integrates JD, Jad, FernFlower, CFR, Procyon seamlessly with Eclipse and allows Java developers to debug class files without source code directly.
- Install Git if you don’t already have it on your computer.

2. Configuration

Go to **Window → preferences**

2.1. Formatter

- Click on **formatter**
- Import this file

```
src/docs/CG.Eclipse.Preferences.Java.CodeStyle.Formatter.xml
```

2.2. Members Sort Order

- **Java → Appearance → Members Sort Order**
- Organize as follow
  - Static Fields
  - Fields
  - Types
  - Static Initializers
  - Initializers
  - Constructors
  - Static Methods
  - Methods

2.3. Save actions

- **Java → Editor → Save actions**
- Check as follow :
  - ☑ Perform the selected actions on save
    - ☑ Format source code
      - ☐ Format all lines
      - ☑ Format edited lines
    - ☑ Organize imports
    - ☑ Additional actions
      - Add missing @Override annotations
      - Add missing @Override annotations to implementations of interface methods
      - Add missing @Deprecated annotations

        - Remove unnecessary casts
        - Sort members excluding fields, enum constants, and initializers
        - Remove redundant type arguments

2.4. Encoding

- Go to **Window** → **Preferences** → **General** → **Workspace**
- At the bottom, choose
    - Other : UTF-8
    - Other : Unix

2.5. Unnecessary Code Detector plugin

- Install Unnecessary Code Detector plugin to see unused declarations or wrongs declarations (public when it could be private for example), with this configuration :
    - Window > preferences > UCDetector
    - ActiveMode = MyMode
    - Ignore > [ ] Bean methods
    - Detect > File name pattern to search =

```
*.xml,MANIFEST.MF,*.tld,*.properties,*.jsp,*.jspf
```

2.6. JAutoDoc plugin

- Install JautoDoc plugin to be able to :
    - add Javadoc with right click → **Add Javadoc** on a class or a package
    - add Javadoc on a function by selecting its name and pressing `Ctrl+Alt+j`

2.7. AutoDeriv plugin

The target folder should always be marked as derived, not to appear in searches. Install the AutoDeriv plugin for this purpose, and configure it by creating the file below.

It can be placed in the project's root directory, then it would be shared to others through the source code repository, but this works only for the current projects, not submodules !

<WORKSPACE>/.derived

```
# set the 'target' and 'ext' folders as derived
target
#ext

# but don't affect the 'keep' sub-folder
!target/generated-sources

# all files with a '.dep' extension are generated
#*.dep

# this specific file is also generated
#src/include/version.h
```

2.8. Commit timeout

- Navigate to **Team** → **Git**
- set **Remote connection timeout (seconds)** = 120

2.9. XML Line size

The Eclipse XML Formatter split lines, and this is an issue for wagon plugin in pom.xml.

- Navigate to **XML** → **Editor**
- Set **Line width** = 200

2.10. Disable stealing of focus by Console in Eclipse

There is one very annoying issue in Eclipse: stealing of focus by Console window.

Console window is displayed when you run application.

When you set focus to some other window like Search results and application prints something on output then Eclipse will automatically switch to Console window. Your search results are gone.

There is simple way how to get rid of such a behavior.

Go to Run/Debug → Console.

Uncheck options :

- ☐ Show when program writes to standard out
- ☐ Show when program writes to standard error

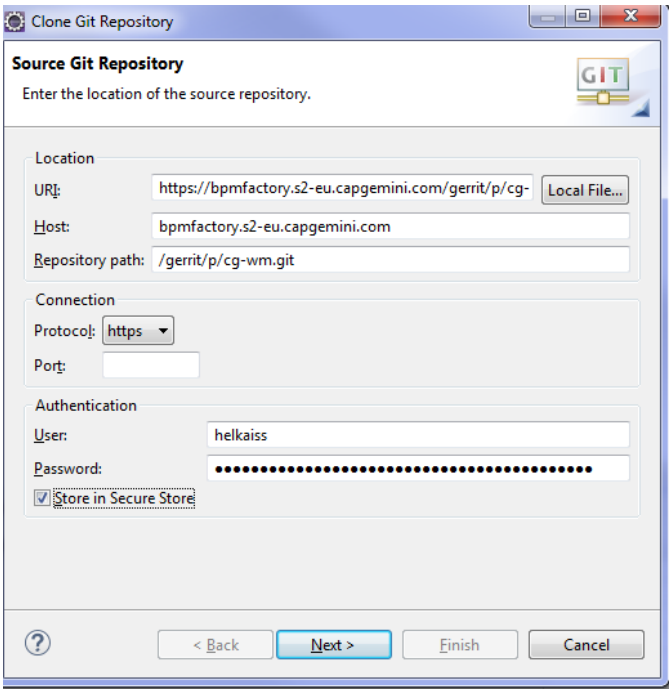
3. Clone a Git repository

Git without Eclipse and Gerrit

To generate your password in Gerrit, right click on your username → **Settings** → **HTTP password** → **Generate Password**

On Eclipse

- Open **Git perspective**
- Click on **Clone a Git repository**
- Fill the required data :



For Gerrit, the process to get the URL is described in Gerrit BP.

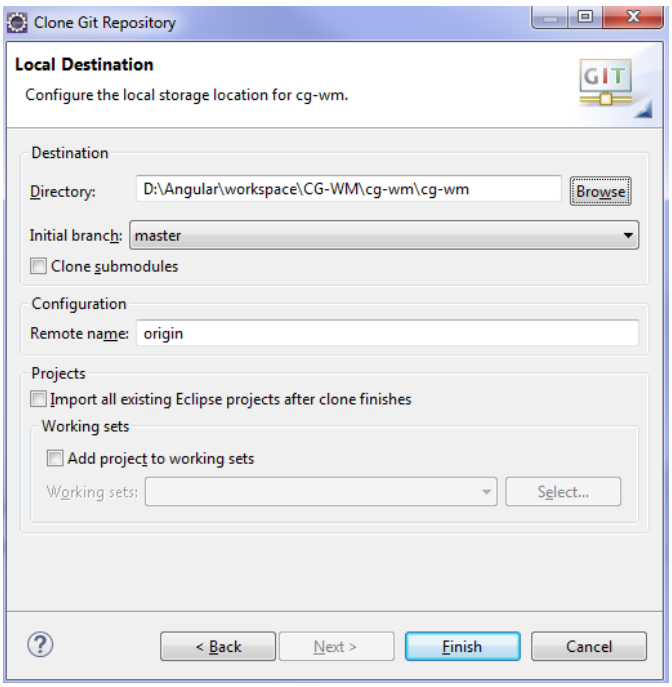
URL : https://your-server.com/gerrit/p/your-project.git

Protocol = https

User = your user

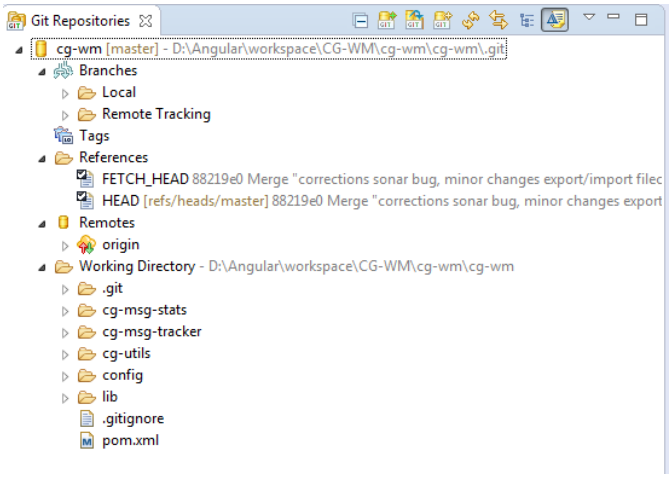
Password generated from Gerrit

☑ Store in Secure Store



Enter your workspace in directory and then click on finish

After creating a new local Git repository, the Git repository of the project should have this structure:



To configure Git, go to **Window** → **Preferences** → **Team** → **Git** → **Configuration**, on **User setting** click on **Open** and add the following configuration :

These configurations are inherited to your repositories. You might have the [user] here too, if you have the same login/password for all your repositories

user .gitconfig settings

```
[gerrit]
    createchangeid = true
[branch]
    autosetuprebase = always
[pull]
    rebase = true
[rebase]
    autoStash = true
```

The **autosetuprebase** = **always** will allow you to do a fetch and a rebase by simply click on "pull".

On **Repository Settings** click on **Open** and enter the following configuration :

repository config settings

```
[core]
    repositoryformatversion = 0
    filemode = false
    logallrefupdates = true
[remote "origin"]
    url = https://host.com/gerrit/p/cg-wm.git
    fetch = +refs/heads/*:refs/remotes/origin/*
    push = HEAD:refs/for/master
[branch "master"]
    merge = refs/heads/master
    rebase = true
    remote = origin
[user]
    name = <user>
    email = <email>
```

After these Git configuration, you may need to restart Eclipse.

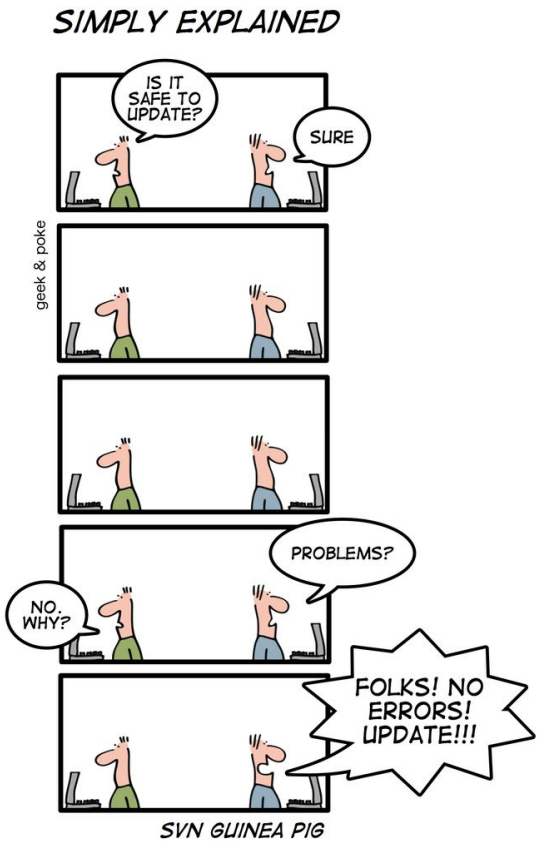
4. Import a project in Project Explorer

Once the repository is cloned, import the project in your Java JEE view. (**File** → **Import** → type **maven** → **Existing Maven Project**).

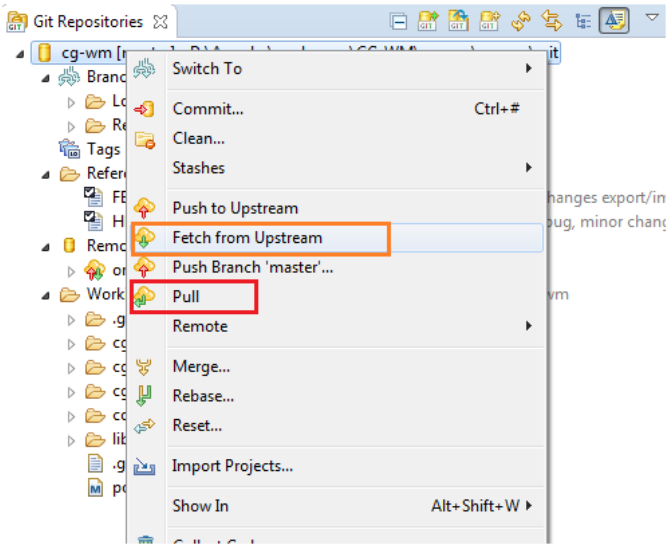
5. Git common usages

5.1. Keep your local repository up to date

Life without an unbreakable build



- To update your git repository, right-click on parent project and select
  - "Pull" (If your git is configured to do a fetch and rebase on "Pull")
  - or "Fetch from upstream", and then "Rebase"

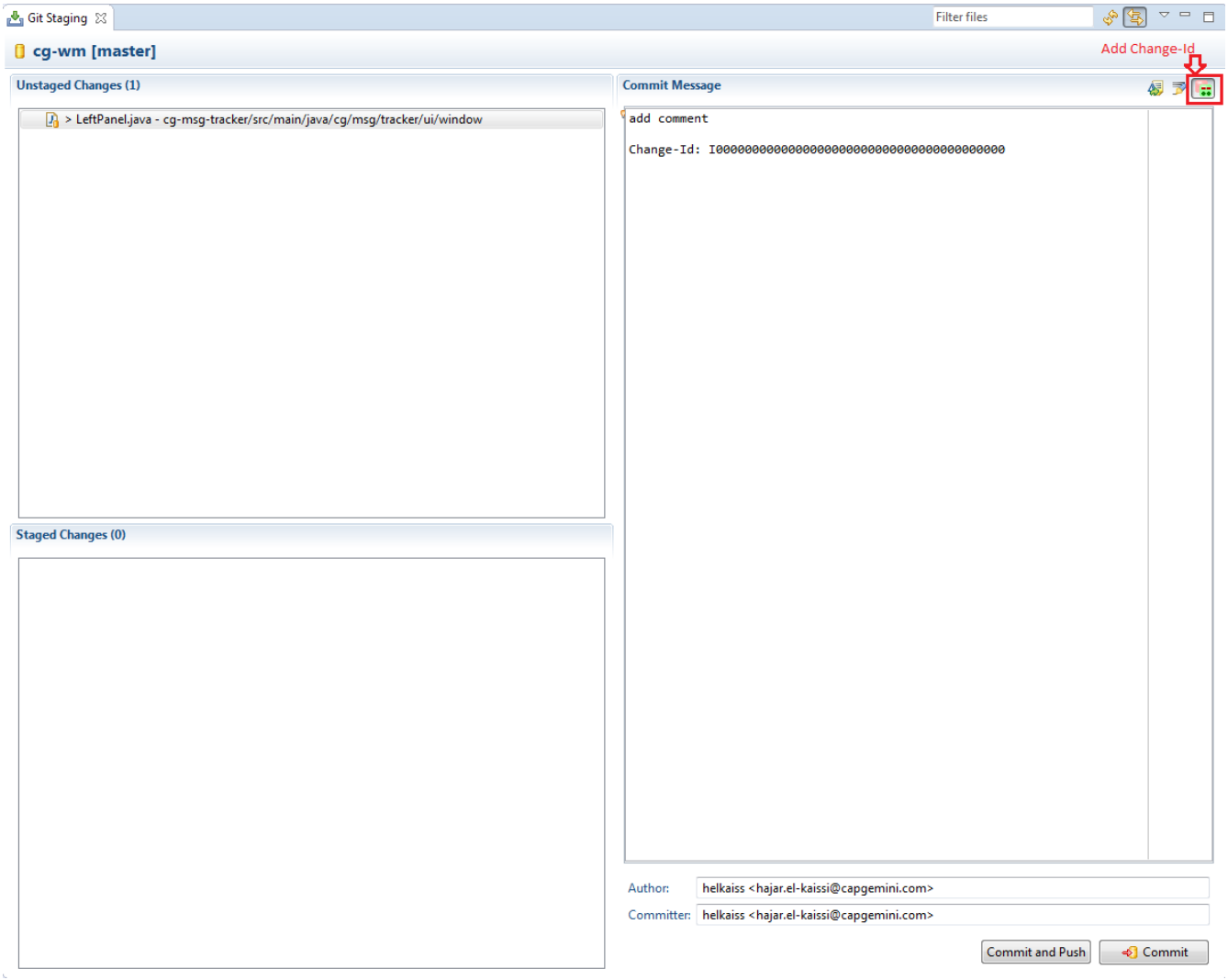


5.2. Commit / push

Life before Gerrit

First of all, you have to understand the commit/push/review process described in Gerrit Best Practices.

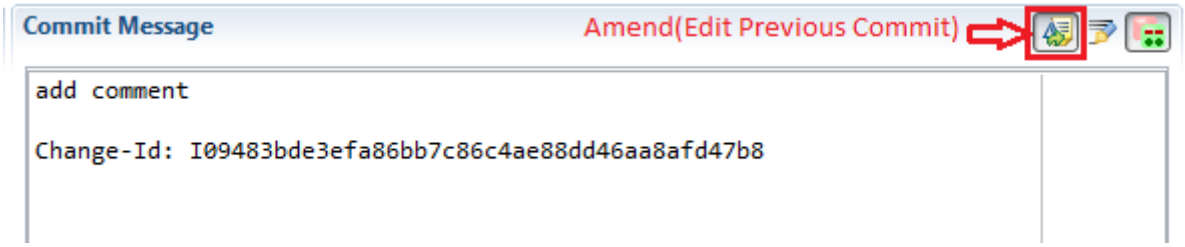
In order to commit a change follow these steps:



- Show view **Git Staging** (or go to perspective Git and select **Git Staging** tab)
- Drag your file not committed from Unstaged Changes to Staged Changes
- On Commit message
  - Click on amend if you want to amend a commit already started add your message and click

on **Add Change-Id** to add an ID for your commit (you add a new change ID only when your previous is validated by the tech lead on Gerrit)

- Click on :
  - **Commit** If you want to do a local commit (changes will be visible only by you)
  - or **Commit and push** If you want to do a commit on the shared repository



5.3. Rebase

In order to get the last version of the git repository, or if one of your commit is in **Merge Conflict** on gerrit, you need to do a rebase. Before doing it, commit all your uncommited changes, otherwise you might lose it. After that you can click on team → fetch from upstream, then team → rebase. If your git is configured to do a fetch and rebase on pull, just select pull. If there is some conflicts during the rebase, correct them, then team → rebase → continue rebase. The rebase is done, you have the lastest version of the git repository and your modifications. You can amend your commit and the "Merge Conflict" on gerrit will disapear.

5.4. Reset Mixed

git reset soft/mixed/hard explained : https://git-scm.com/docs/git-reset

In team → reset, you can find a usefull functionnality called Reset Mixed. The reset mixed will allow you to get the last version of the git repository (after a fetch from upstream), while keeping your modifications. It works like a rebase, except that your commit will be cancelled on your local workspace. You will be branched on the last commit on the git repository. Use it if you want to abandon a commit, do a fresh commit, or merge two of your commits in one for example. Reset Hard can be used if you want to cancel all your modifications. All modifications will be erased.

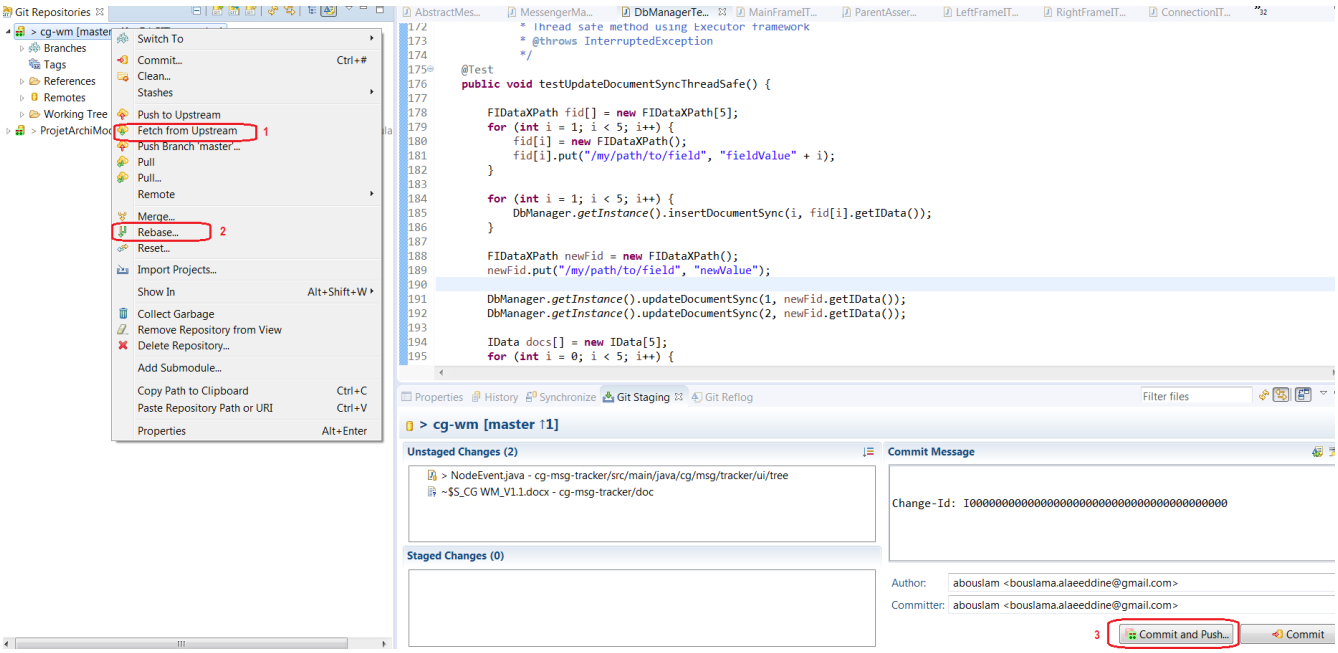
6. Troubleshooting

6.1. Two commits at the same time

If you have a commit on gerrit, and if you want to change a file that is already in this commit, you should amend this commit instead of making a second commit.

You can have two or more commits at the same time if they are not dependants. To do so, you have to do a reset mixed on the state before your first commit in order to commit the second one. It works like this : If A is the last commit on the git repository, you can do a first commit B. To make a second commit C, reset mixed on commit A, then commit. If you want to amend your commit B,

reset mixed on A before doing it, and copy the description and change id for your commit on gerrit, to paste it like it is a new commit. You also need to select again the files that were in this commit.



6.2. Pull from another developer

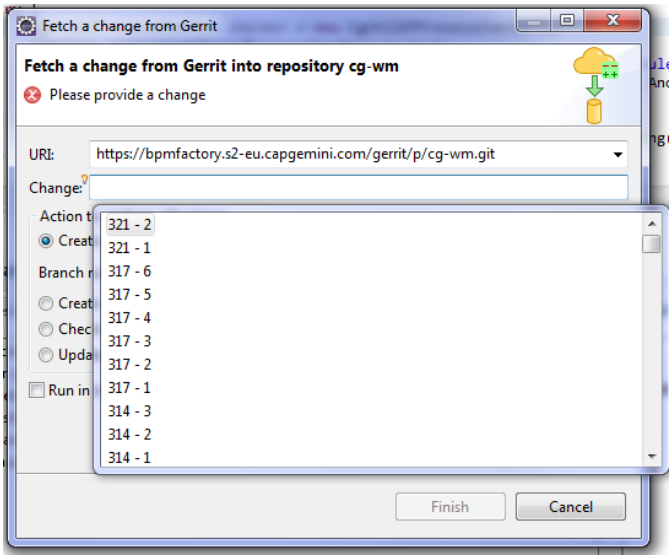
To download a change of another developper from gerrit to your local :

- go to **Git Repositories** on your Eclipse
- right click on your repository
- select **fetch from Gerrit**
- on **Change** press `Ctrl+Space`
- select your change
  - To change is of the form :

```
<(small) change number> - <patchset number>
```

  - The small change number is visible in gerrit page in the upper left area (clickable)
- Click on **finish**

If you don't have the option **fetch from Gerrit** : Go to the **Git repositories** view, under your repo, in **Remotes**, right click on **origin** and select **Gerrit configuration...**. Check that everything is OK and click **Finish**. Now you should have the option.



To keep this branch up to date with the master :

- Go to **Git Repositories**
- Right click on your repository > **Pull**
- Navigate in you repository → **Branches** → **Local**
- Rigth click on the **master** branch → **Rebase On**
- Resolve the conflicts if any

To switch back to your master, right click on your repo → **Switch To** → **master**.

6.3. Missing Change-Id in commit message footer

On your first push, you might get this error from Gerrit

```
 [f57ea8b] missing Change-Id in commit message footer
Processing changes: refs: 1
Processing changes: refs: 1, done
ERROR: [f57ea8b] missing Change-Id in commit message footer
 Hint: To automatically insert Change-Id, install the hook:
  gitdir=$(git rev-parse --git-dir); scp -p -P 29418 yremila@cdsdouane.pl.s2-eu.nvx.com:hooks/commit-msg
${gitdir}/hooks/
 And then amend the commit:
  git commit --amend
```

Doing a **reset mixed** on the initial commit would solve the problem.

If not, here is another solution :

- Download this file https://gerrit-review.googlesource.com/tools/hooks/commit-msg
- Put it in myproject/.git/hooks/
- Restart Eclipse
- Retry the commit

Last solution, requiring Cygwin :

```
  gitdir=$(git rev-parse --git-dir); wget -P ${gitdir}/hooks/ http://bcouetil@cric.pl.s2-
eu.nvx.com/gerrit/tools/hooks/commit-msg && chmod +x ${gitdir}/hooks/commit-msg
```

6.4. .classpath and other not-to-push files

If there are files in git repository not to be pushed, remove them in your next push :

- Update the .gitignore file, if not already done
- Go to **Git Staging** view
- Right click on the file → **Team** → **Untrack**
- The file should be moved to **Staged changes** (it can take a few seconds)
- This will be taken into account in next **Commit and Push…**

6.5. See SonarQube for an older build

SonarQube does not have history available for browsing, you can only see the last build. So you have to retrigger the gerrit patch to see specific data associated to your change.

You can do an empty **commit amend** from Eclipse. But you can also retrigger from Jenkins.

- Go to Jenkins homepage
- Navigate to the pipeline/job
- If your build is still in the history
  - Open it
  - Select **Retrigger**
- Else if your build has been deleted
  - Go to Jenkins homepage
  - Click on **Query and Trigger Gerrit Patches**
  - In **Query String**, put your change-id
  - Click **SEARCH**
  - Select the change
  - Click **TRIGGER SELECTED**

6.6. Double remote : get latest commits from gitlab into gerrit

In the case of a gitlab to gerrit migration, you may have a common ancestor but commits on Gerrit and commits on Gitlab. It’s easy if you can replace the whole gerrit master, but here is the way to go for a clean merge :

- have your gerrit clone in Eclipse

- Navigate to **Git** view
- Right click on your **Remotes** where there is already a Gerrit **origin**
- Choos* **Create Remote...**
- Create your gitlab remote while configuring fetch with only the **master** branch. Name it **gitlab**
- right click on the new remote and choose **fetch**
- right click on your repository and select **pull**
- Finish by this in command line :

```
git push origin master
```

6.7. Get rid of ORIG_HEAD / FETCH_HEAD in Git history

ORIG_HEAD is last value of HEAD before dangerous operation. This reference and FETCH_HEAD are not real commits, and can be hidden in **Preferences → Team → Git → History** by deselecting **Additional Refs**

7. Appendix

7.1. Revision marks

Differences since last tag