

How To use Gerrit

Developer

FAQ

[Q: How do I get the git repository from Gerrit initially?](#)

[Afterwards, you have to specify the new remote repository as a Gerrit repository \(Git Repositories view > woped > Remotes > origin > Gerrit Configuration\)](#)

[*](#)

[-](#)

[Q: How do I create/insert a ssh key?](#)

[Q: Where/in which branch do I create new code/modify existing code?](#)

[Q: How do I commit new changes?](#)

[Q: Where can I see the latest changes on the project?](#)

[Q: How do I delete file/s?](#)

[Q: How do I get the changes from other developers?](#)

[Q: What is the difference between fetch and pull?](#)

[Q: How can I see recently pushed changes by other developers?](#)

[Q: How can I reset my HEAD or my workspace?](#)

[Q: What is the best workflow to submit code?](#)

[Q: What is the best workflow to review code?](#)

[Q: How do I get notified of new changes?](#)

[Q: How can I adopt changes from the remote repository in my workspace by using the 'Team Synchronizing' view?](#)

[Q: What is an alternative workflow to submit code w/o using automated merge/rebase?](#)

Troubleshooting

[Issue: Fetch failed](#)

[Issue: Checkout conflict \(while doing a checkout, merge or rebase\)](#)

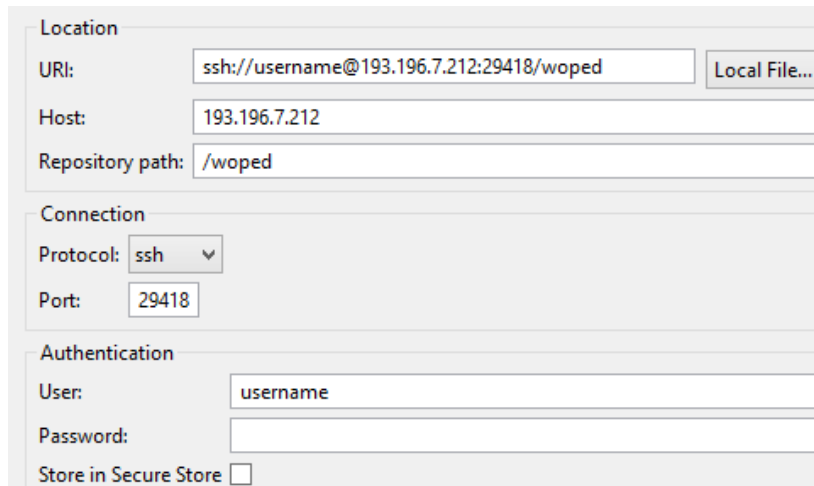
[Issue: Merge conflict](#)

[Issue: Rebase conflict](#)

FAQ

Q: How do I get the git repository from Gerrit initially?

A: In order to get started with WoPeD's source code, you have to clone the Gerrit repository. Go to eclipse's 'Git Repositories' view > 'Clone' and insert the repository's URL (replace 'username' with your Gerrit user name).



Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

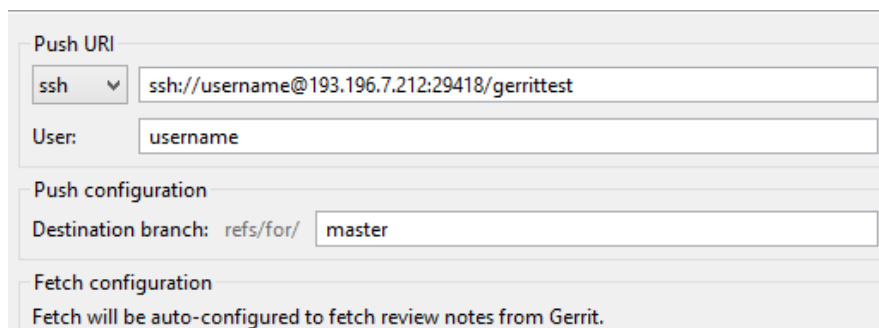
Authentication

User:

Password:

Store in Secure Store ☐

Afterwards, you have to declare the cloned repository as a Gerrit repository (Git Repositories > WoPeD > Remotes > right-click on the remote's name (regularly 'origin') > Gerrit Configuration > Finish.



Push URI

User:

Push configuration

Destination branch:

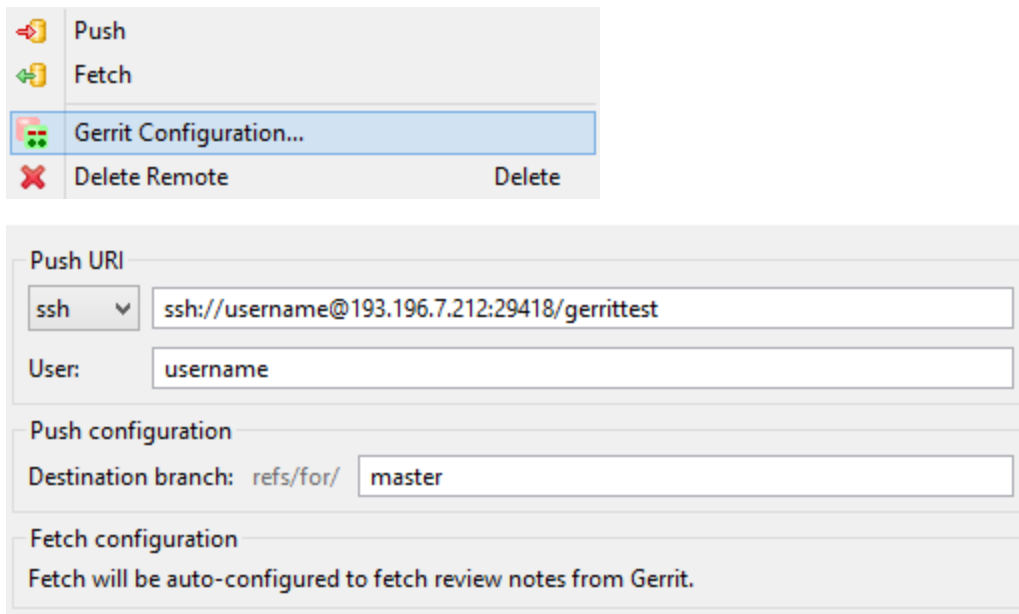
Fetch configuration

Fetch will be auto-configured to fetch review notes from Gerrit.

Annotation:

- Using ssh requires providing a valid ssh key, instead of specifying a password.
- If you want to use http instead of ssh, you have to replace port 29418 with 8080. In this case, you also have to provide the password of your Gerrit account.
- If you do not have a (pre-)installed EGit plugin for eclipse, you have to download it from the Eclipse Marketplace (Help > Eclipse Marketplace).

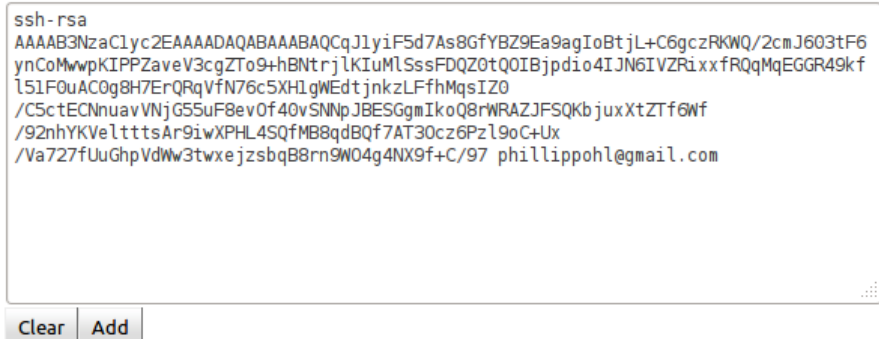
Afterwards, you have to specify the new remote repository as a Gerrit repository (Git Repositories view > woped > Remotes > origin > Gerrit Configuration)



Q: How do I create/insert a ssh key?

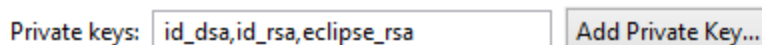
A: Go to Window > Preferences > General > Network Connections > SSH2 > Key Management and let eclipse create a ssh key for you (alternatively insert an existing, valid ssh key here). Afterwards, you enter your Gerrit user account settings and specify the ssh key that you have just created in eclipse. Please make sure you copy exactly the same key (w/o any spaces, etc.).

Add SSH Public Key (GitHub's Guide to SSH Keys)



```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAj1yIF5d7As8GfYBZ9Ea9agIoBtjL+C6gcZRWQ/2cmJ603tF6
ynCoMwWpKIPPZaveV3cgZTo9+hBNtrjLKIuMLSSsFDQZ0tQ0IBjpdio4IJN6IVZRixxfRQqMqEGGR49kf
151F0uAC0g8H7ErQRqVfN76c5XH1gWEdtjnkzLFfhMqsIZ0
/C5ctECNnuavVNjG55uF8ev0f40vSNnpJBESGgmIkoQ8rWRAZJFSQKbjuxXtZTf6Wf
/92nhYKVeltttsAr9iwXPHL4SQfMB8qdBQf7AT30cz6PzL9oC+Ux
/Va727fUuGhpVdWw3twxejzsbqB8rn9W04g4NX9f+C/97 phillippohl@gmail.com
```

Afterwards, go to Window > Preferences > General > Network Connections > SSH2 > General and add the previously created private key.

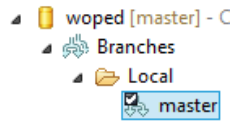


Q: Where/in which branch do I create new code/modify existing code?

A: We strongly recommend only modifying the source code within the local master branch:

Git Repository View > WoPeD > Branches > Local > master > Checkout.

Afterwards, you should see a little check mark beside the branch's name.



Q: How do I commit new changes?

A: Committing changes takes multiple steps:

1. Commit

Commits the selected changes from your workspace to your local git repository. When you are using EGit's 'commit' command, you are able to choose among all individual files that have been modified since the last commit.

2. Push

Pushes the committed changes from the local git repository to the remote repository, which is provided by your chosen source code hosting service, i. e. Gerrit, Sourceforge, GitHub, etc.

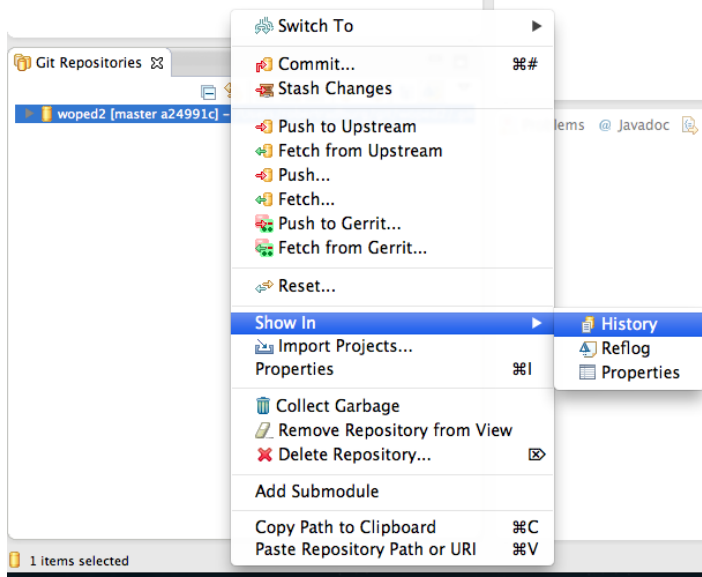
Note:

Regularly, you can ignore the 'index' stage while doing your work. An important exception is resolving merge conflicts (http://wiki.eclipse.org/EGit/User_Guide#Resolving_a_merge_conflict).

Q: Where can I see the latest changes on the project?

A: In the project history

1. Select the git repository you would like to see the history of
2. Right click on it and go to 'Show in...'->'History'



3. In the now opening history tab you see the projects' history.

Q: How do I delete file/s?

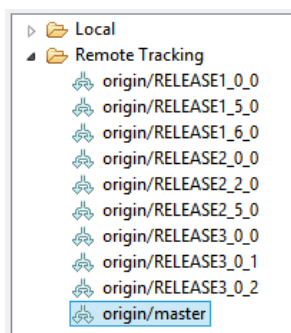
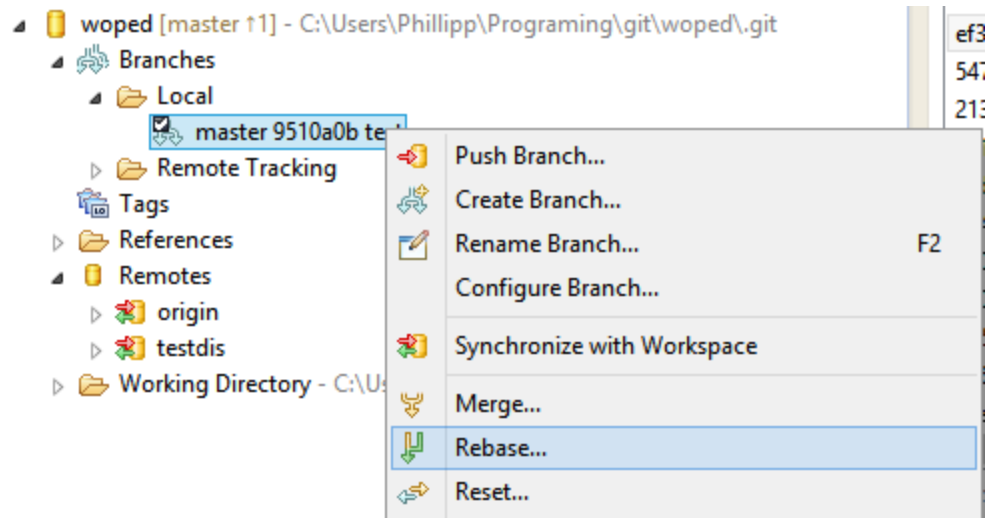
A: You simply delete the desired file/s in your workspace and commit/push the changes to the Gerrit repository.

Q: How do I get the changes from other developers?

A: In order to update your local repo with changes from other developers, you ought to fetch the current HEAD from the Gerrit repository regularly (using 'Fetch from Upstream') and merge/rebase them with your local repository. Alternatively you can use the 'pull' command.

Q: What is the difference between fetch and pull?

A: Fetch simply gets the latest version of the git repository from the Gerrit repository (but does not overwrite anything in your workspace or local repository). In order to bring the changes to your local repo, you have to merge or rebase your local master branch with Gerrit's master branch.

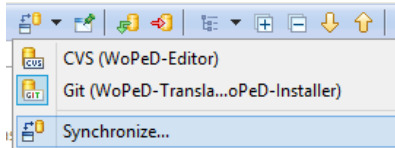


'Pull' does the same as 'fetch', but also merges automatically the changes with your local repository at the same time, i. e. pull = fetch + merge.

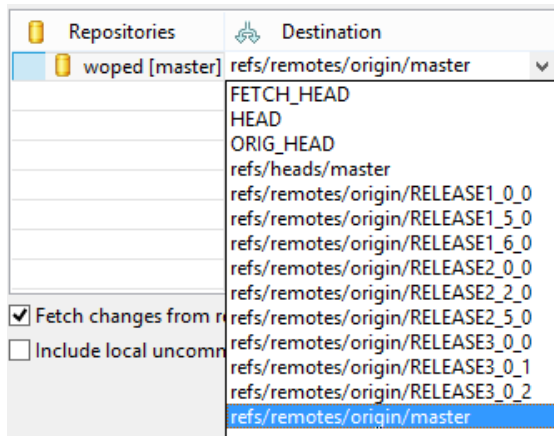
Q: How can I see recently pushed changes by other developers?

A: There is more than one way to achieve this goal:

1. Compare a certain file with the remote's current HEAD branch:
Select a file > Compare With > Branch, Tag or Reference > Remote Tracking > **origin/master**
2. Compare the whole workspace with the remote's current HEAD branch:
Enter the 'Team Synchronizing' view > press 'Synchronize'



Choose the remote's master branch



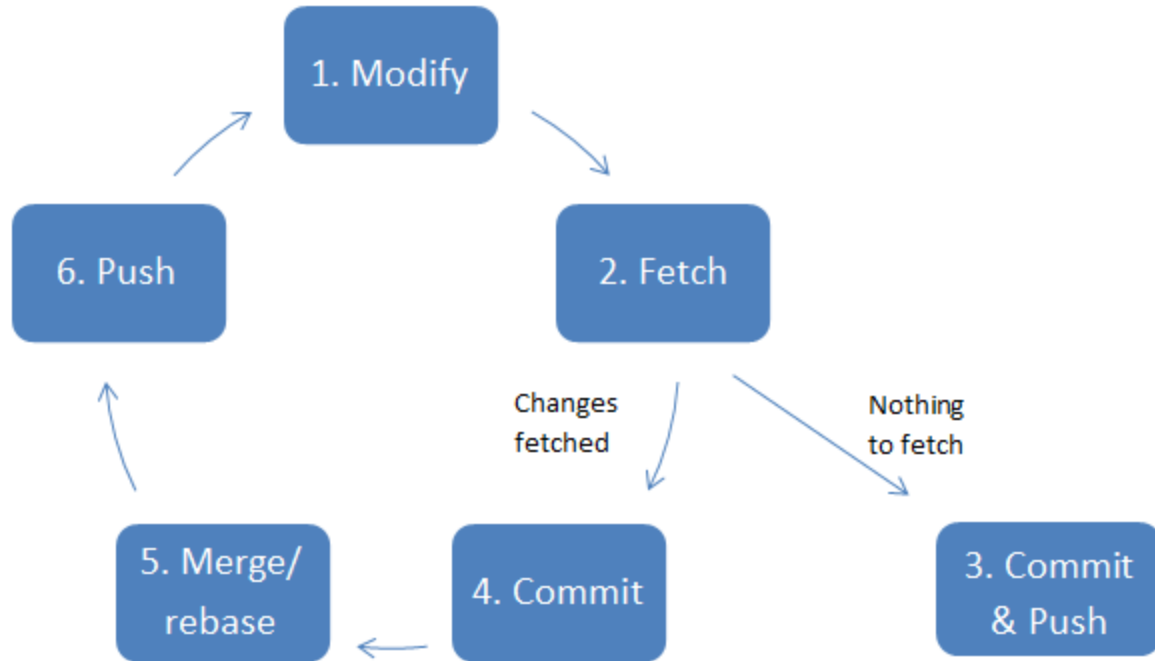
Q: How can I reset my HEAD or my workspace?

A: Show in your 'history' view and select the commit upon you wish to reset your HEAD and/or your workspace. When you click 'Reset' you have 3 options to choose (especially soft and hard resets are useful):

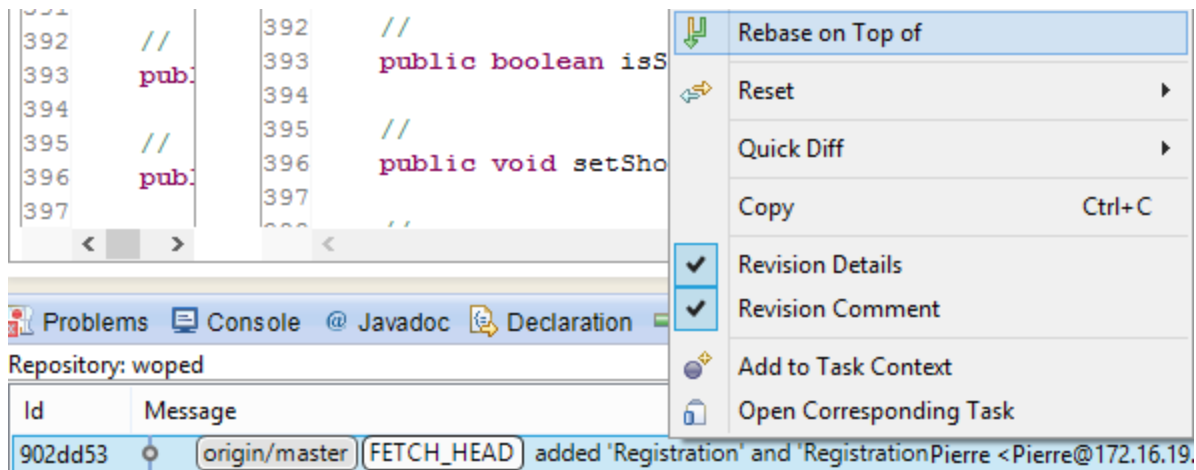
- **soft** - the HEAD points now to the new commit, the index and the working tree are unchanged
- **mixed** - the HEAD points now to the new commit, the index is updated, the working tree is unchanged
- **hard** - the HEAD points now to the new commit, the index and the working tree are updated

Q: What is the best workflow to submit code?

A: We recommend a fetch - commit - merge/rebase - push workflow



1. Modify code in your workspace.
2. 'Fetch from Upstream' to get the current HEAD from the Gerrit repository.
3. If there are **no** changes to fetch, you have already had the latest changes from the remote included in your local repository. Just commit and afterwards push your changes to the local ('Commit') respectively Gerrit ('Push to Gerrit or Push to Upstream') repository. Skip the following steps.
If there are changes to fetch, you will see an arrow (pointing down) beside your local repository. The number next to the arrow indicates how many commits your local repository is behind the remote repository.
4. Update your local repository with the changes from your workspace by using 'Commit'.
5. Now you have to incorporate the changes from the Gerrit repository by doing a merge or rebase with Gerrit's master branch.



Doing so will bring the latest changes to your local repository such that it now contains both, the latest changes from the Gerrit repository and the new changes from your workspace.

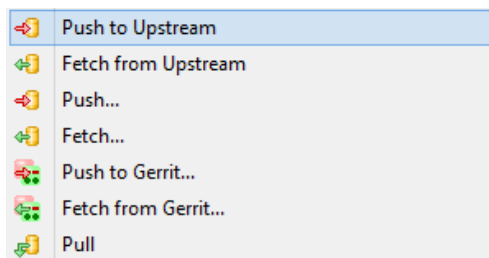
Further information concerning difference between merge and rebase:

<http://www.fiveminutes.eu/git-merge-and-rebase-the-simple-explanation/>

Note:

If there are any merge/rebase conflicts occurring, you have to resolve them manually (see chapter [Troubleshooting](#)).

6. Last but not least, you have to push the changes from your local repository to the Gerrit repository by using 'Push to Gerrit'/'Push to Upstream'.



Q: What is the best workflow to review code?

A: Gerrit knows two separate stages that new code has to pass in order to be integrated in the source code: 'Code-Review' and 'Verification'.

After somebody committed a new piece of code, this commit has to be reviewed by either two registered user or one integrator and afterwards verified by an integrator:

1. Log in onto Gerrit and go to All > Open. In this dashboard, you will see all unreviewed commits. Identify the oldest commit and start reviewing it (therefore you have to be registered and signed in).

The screenshot shows the Gerrit web interface. At the top, there is a search bar with the text "Search for status:open". Below it is a table with two columns: "ID" and "Subject". The first row shows the ID "169f99866" and the subject "Added *.jar to .gitignore". Below the table, there is a section for commit details. It shows the "Author" as "phillippohl <phillipp-ohl@web.de> Jan 31, 2013 4:07 PM" and the "Committer" as "phillippohl <phillipp-ohl@web.de> Jan 31, 2013 4:07 PM". The "Parent(s)" section shows a merge of "15c0e61a2055c15ec12e39307f6be35f3d9412e3 Merge 'Added .project'". The "Download" section shows links for "checkout", "pull", "cherry-pick", "patch", and "Anonymous HTTP". Below the download links, there is a red box around the "Review" button, and other buttons for "Abandon Change", "Rebase Change", and "Diff All Side-by-Side".

ID	Subject
169f99866	Added *.jar to .gitignore

Author: phillippohl <phillipp-ohl@web.de> Jan 31, 2013 4:07 PM
Committer: phillippohl <phillipp-ohl@web.de> Jan 31, 2013 4:07 PM
Parent(s): 15c0e61a2055c15ec12e39307f6be35f3d9412e3 Merge "Added .project"
Download: checkout | pull | cherry-pick | patch | Anonymous HTTP
git fetch ssh://phillippohl@193.196.7.212:29418/gerrit

Review Abandon Change Rebase Change Diff All Side-by-Side

2. Optional: If you are not sure about whether the new code will work or not, you are able to test the change by using the 'Fetch from Gerrit' command in eclipse and specifying the change you want to test (press CTRL + Space in the 'Change' text field to see a list of all changes that are available for fetching).

As a result, you will have a new 'change' branch in your local repository that includes the changes by this commit. Check if everything works and delete the local 'change' branch after finishing the review.

Fetch a change from Gerrit into repository gerrittest

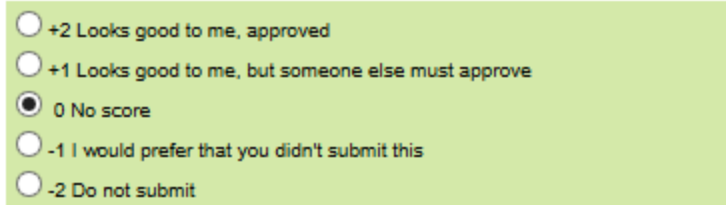
Please select a Gerrit URI and change to fetch

The screenshot shows a dialog box titled "Fetch a change from Gerrit into repository gerrittest". It has a text input field for "URI" with the value "ssh://phillippohl@193.196.7.212:29418/gerrittest". Below it is a text input field for "Change" with the value "refs/changes/02/2/1". Under the heading "Action to perform after fetch", there are four radio button options: "Create and checkout a local branch" (selected), "Create and checkout a tag", "Checkout FETCH_HEAD", and "Update FETCH_HEAD only". A text input field for "Branch name" has the value "change/2/1".

URI: ssh://phillippohl@193.196.7.212:29418/gerrittest
Change: refs/changes/02/2/1
Action to perform after fetch
☒ Create and checkout a local branch
Branch name: change/2/1
☐ Create and checkout a tag
☐ Checkout FETCH_HEAD
☐ Update FETCH_HEAD only

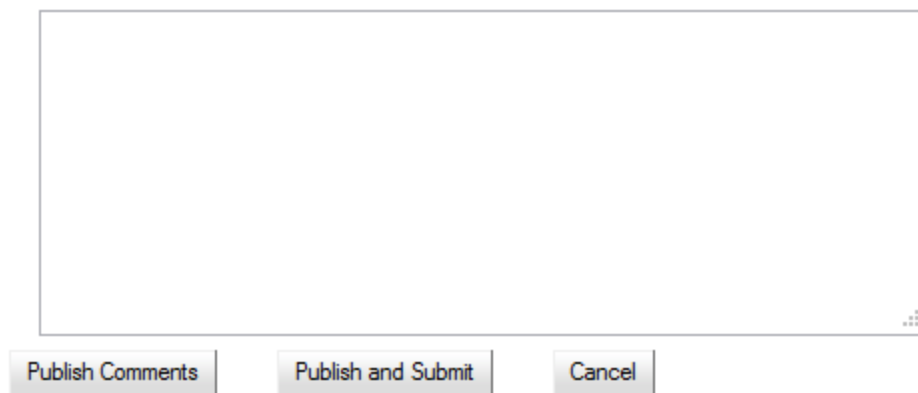
3. If you are done with reviewing the content of the commit, you have three different possibilities to vote:
 - a. +1/+2
You are satisfied with the new code and want to integrate it in WoPeD.
 - b. 0
You are indifferent about the new code.
 - c. -1/-2
You are not happy with the new code and do not want to integrate it in WoPeD.

Code Review:




☐ +2 Looks good to me, approved
☐ +1 Looks good to me, but someone else must approve
☒ 0 No score
☐ -1 I would prefer that you didn't submit this
☐ -2 Do not submit

Cover Message:



A commit's review is finished successfully when it reaches a sum of at least +2.
If you want to add a second reviewer, you can Gerrit notify him/her by sending an email.



Note:

+2 and -2 votes are only available if you are a member of the 'Integrator' group.

4. Only as Integrator: Before Gerrit is able to merge the reviewed commit into the master branch, an integrator has to verify the commit (All > Open > Review).

Verified:



☐ +1 Verified
☒ 0 No score
☐ -1 Fails

5. After the commit is verified, Gerrit merges it automatically into the project's master branch.
6. In case the commit fails on one of the two stages (Code-Review or Verification), it is not automatically rejected by Gerrit.
Therefore, one of the integrators has to abandon the commit manually. Subsequently, the responsible developer is notified automatically and has to improve the code.
Afterwards, the modified code has to run through both, the submit and review workflow, again.

Q: How do I get notified of new changes?

A: Assumed you are on Gerrit, go to Settings > Watched Projects and specify of which kind of changes you want Gerrit to inform you (via e-mail).

<i>Project Name</i>	<input type="text" value="Project Name"/>
<i>Only If</i>	<input type="text" value="branch:name, or other search expression"/>
<input type="button" value="Watch"/>	<input type="button" value="Browse"/>

	<i>Project Name</i>	<i>Email Notifications</i>		
		<i>New Changes</i>	<i>All Comments</i>	<i>Submitted Changes</i>
<input type="checkbox"/>	gerrittest	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	woped	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Q: How can I adopt changes from the remote repository in my workspace by using the ‘Team Synchronizing’ view?

A: You can either overwrite or merge single changes within the ‘Team Synchronizing’ view.

1. Overwrite will overwrite the selected file in your workspace with the file from the remote repository.
2. Merge will try to incorporate changes into your file, i. e. it will assemble one file, which integrates changes from both files (file from your workspace and file from the remote repository).

Important:

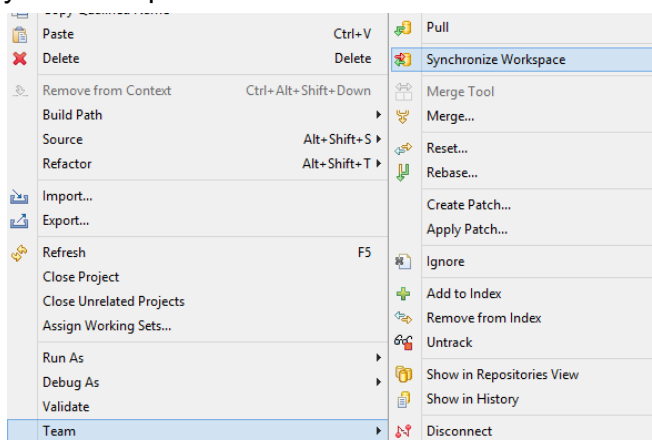
You will not see any changes in the current ‘Team Synchronizing’ view, since you have just modified files in your **workspace**. Because of that, the changes are currently only in your workspace and **not in your local WoPeD** repository. To bring the changes to your local repository, you have to commit the changes.

Q: What is an alternative workflow to submit code w/o using automated merge/rebase?

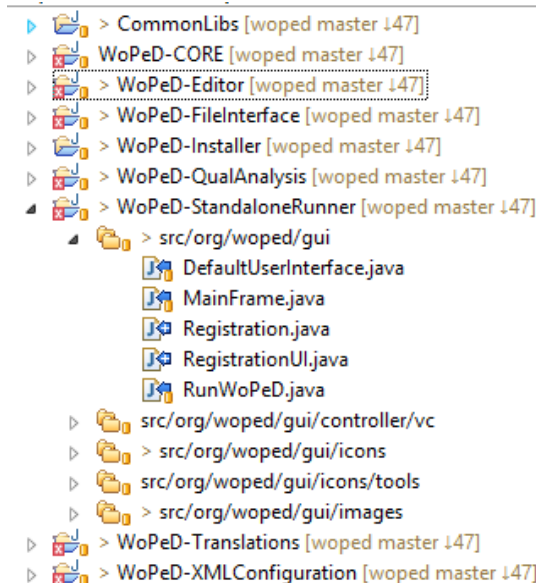
A: We do recommend using the automated workflow, described before. Do the following workflow only if you are aware of what you are doing.

This workflow might especially be important if you are facing merge/rebase errors during the automated workflow.

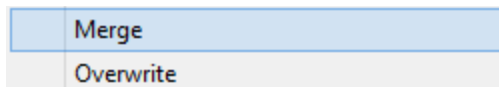
1. Modify code in your workspace
2. Synchronize your workspace with Gerrit’s HEAD branch (regularly the master branch) → you will fetch the current HEAD files such that you can compare them with the files in your workspace



3. In the 'Team Synchronizing' view you are able to see the differences between the files in the workspace and the latest files from the remote repository.

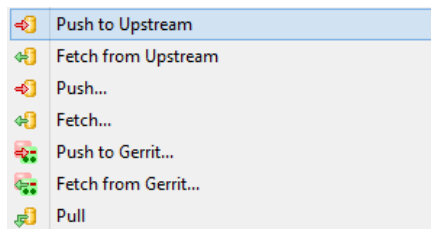


4. Manage the changes between the files either automatically by using merge or overwrite (as described a question before) or manually by editing the files on your own.



When you have resolved the differences between the files, you now have all the updated files in your **workspace**.

5. Now you have to update your local repository with the latest version from the remote repository in order to include the latest commits. To achieve this goal you have to merge/rebase your local repository with the remote repository.
6. Afterwards, you have to bring the previously edited files from your workspace to the git repositories. Therefore you have to commit the changes to the local repository.
7. Last but not least, you have to push the changes from your local repository to the Gerrit repository by using 'Push to Gerrit'

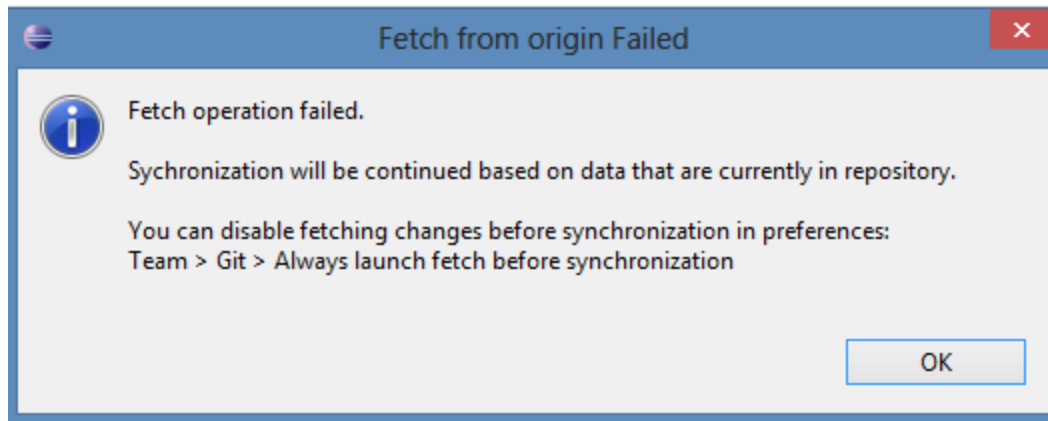


Annotation:

Please be aware that it is likely for 'Checkout Conflict' errors to happen if you are following this workflow (see [Troubleshooting](#)).

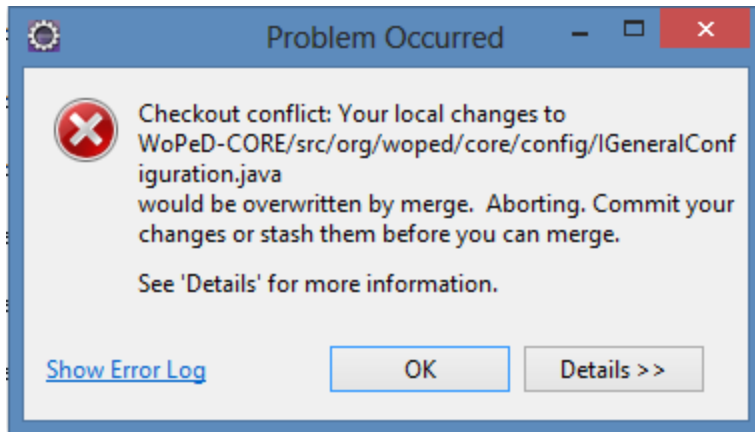
Troubleshooting

Issue: Fetch failed



Reason: Most likely issue is your network connection.

Issue: Checkout conflict (while doing a checkout, merge or rebase)



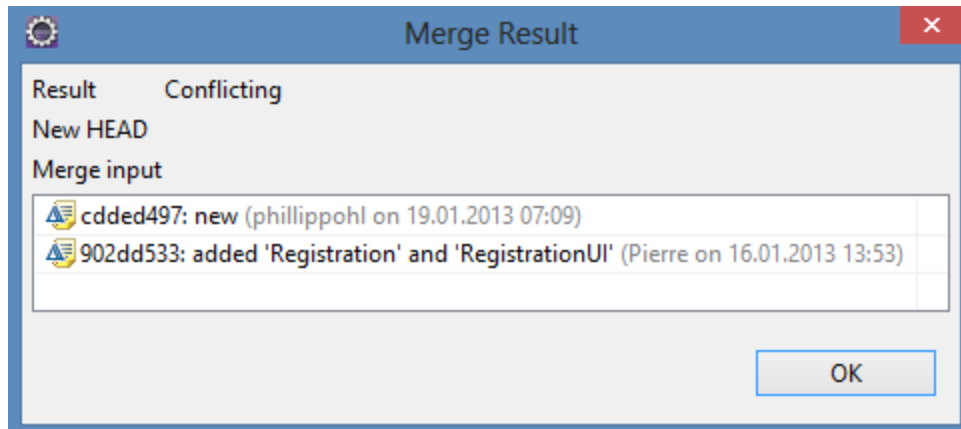
Reason:

This error indicates that local, uncommitted changes in your repository would be overwritten if you would continue with your operation. Therefore the operation is aborted.

Solution:

To resolve the conflict you simply have to commit the changes from your workspace to your local repository or reset your workspace if you do not want to keep your local changes (so they will be overwritten).

Issue: Merge conflict



Reason:

Error while merging your local with the remote repository.

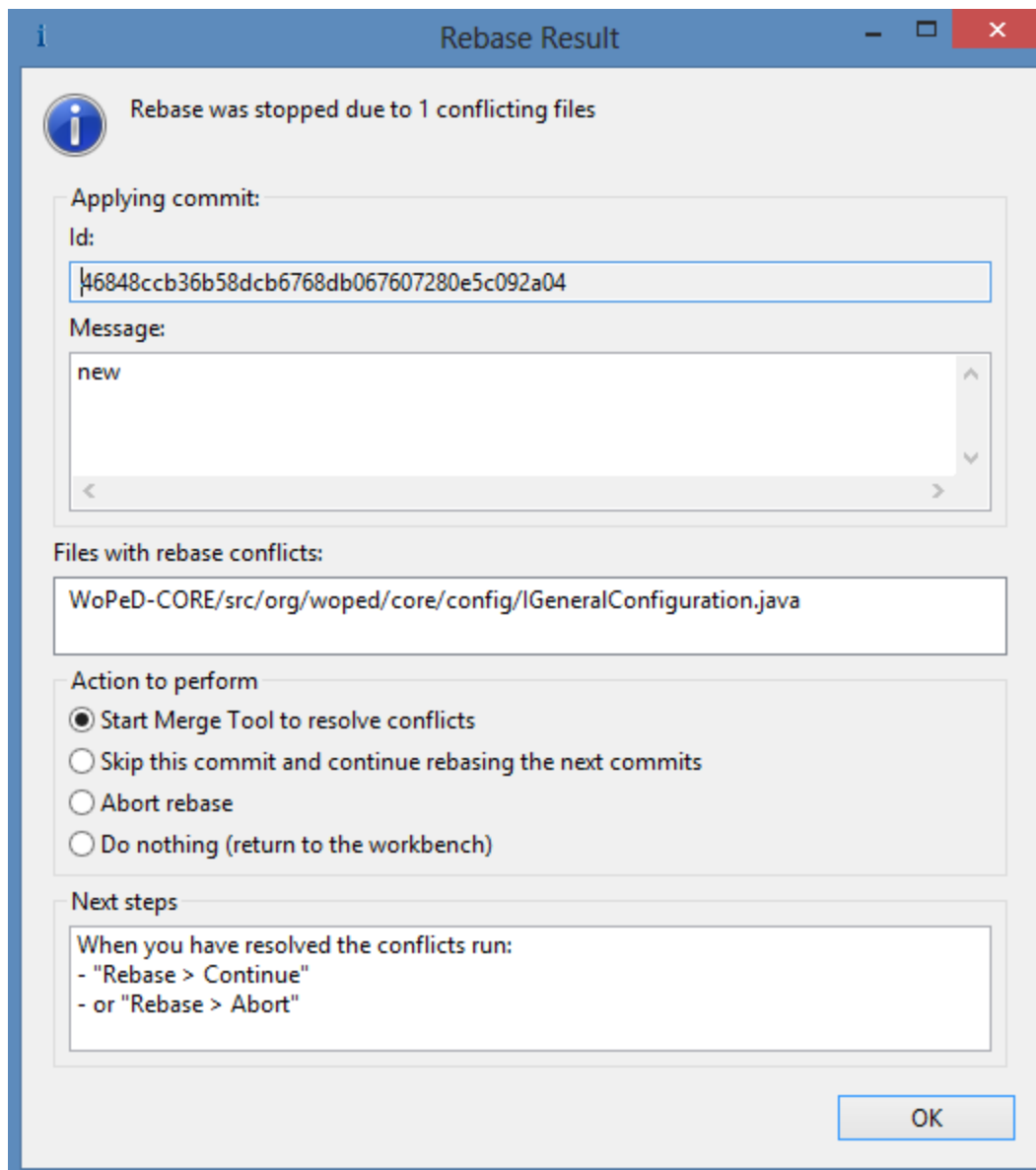
Solution:

- Navigate to the conflicting resource
- Edit the content of the conflicting resource
- Tell EGit that the conflict is resolved with **Team > Add to Index** (red conflict icon should change)
- Commit the conflict resolution with **Team > Commit**

Further information:

http://wiki.eclipse.org/EGit/User_Guide#Resolving_a_merge_conflict

Issue: Rebase conflict

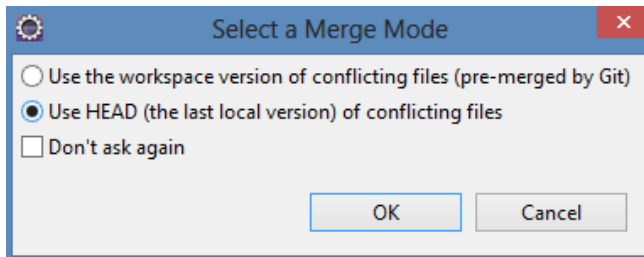


Reason:

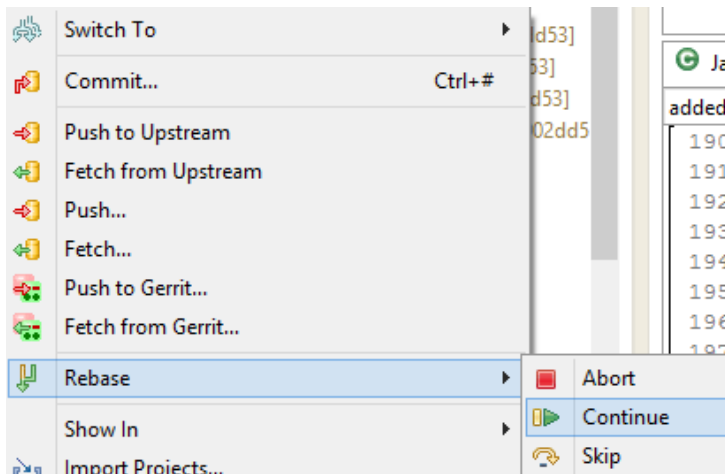
Error while rebasing your local with the remote repository.

Solution:

- Start Merge Tool while using the HEAD version of conflicting files (rebase interactive mode entered)



- Navigate to the conflicting resource
- Edit the content of the conflicting resource
- Tell EGit that the conflict is resolved with **Team > Add to Index** (red conflict icon should change)
- Commit the conflict resolution with **Team > Commit**
- Continue the rebase (repository > Rebase > Continue)



Note:

If you want to exit the rebase interactive mode, go to your repository and abort the rebase.

