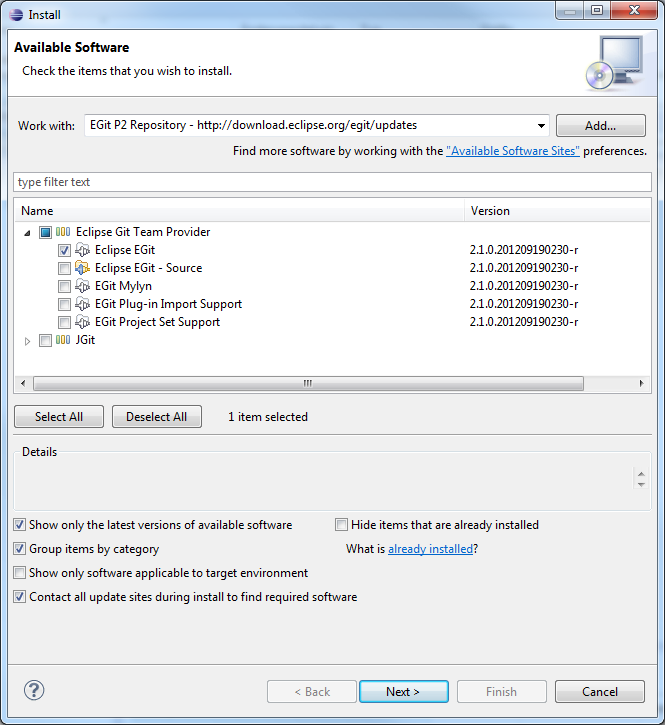
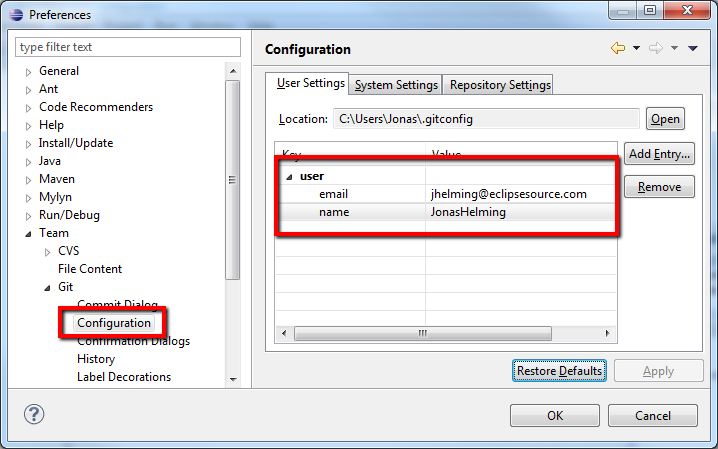
**Installing EGit in Eclipse**

If you use an older version of Eclipse, open the Eclipse Wizard to install new software Help => Install New Software. Insert <https://download.eclipse.org/egit/updates> after Work with: and hit Return. Select Eclipse EGit as a child from Eclipse Team Git Provider. You don’t have to install any other plugins. Click Next and confirm your selection in the following window by pressing Next again. Finally, accept the terms of use and the license agreement and click Finish to start the installation. After the installation has finished restart Eclipse.



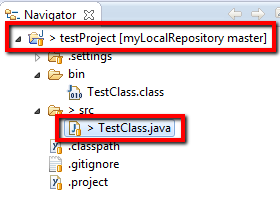
## EGit Configuration

Every commit in EGit will include the user’s name and his email-address. These attributes can be set in the Preferences-window Window => Preferences. Navigate to Team => Git => Configuration and hit the New Entry… Button. Enter user.name as Key and your name as Value and confirm. Repeat this procedure with user.email and your email address and click OK in the Preferences window. The username and email should be the same you use for your Git account

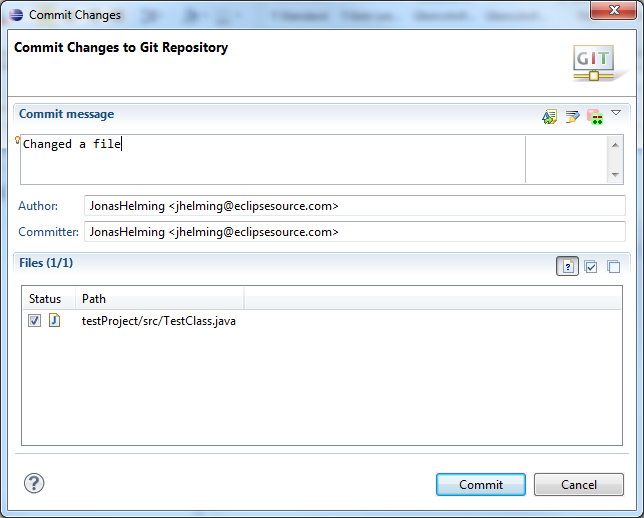


## Commit

To save changes made in your workspace to your repository, you will have to commit them. After changing files in your project, a “>” sign will appear right after the icon, telling you the status of these files is dirty. Any parent folder of this file will be marked as dirty as well.



If you want to commit the changes to your repository, right click the project (or the files you want to commit) and select Team => Commit… . This will open a new window, allowing you to select the files you want to commit. Before you can commit the files, you will have to enter a commit message in the upper textbox. After you’re done, click Commit to commit the selected files to your repository.

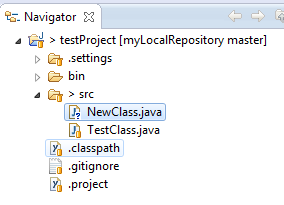


Note that the status of the changed file is Mod., not staged. By staging the files before you commit (see the section “Additional Information”), you can change the status to Modified (and the dirty sign to a staged icon).

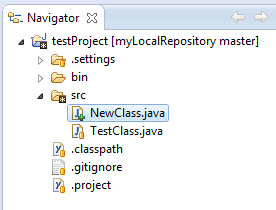
If you later realize that your previous commit was incomplete (e.g. you forgot to commit a file) or your commit message was wrong, you might want to use Amend previous commit. This will merge the current commit and the previous commit into one,

## Adding Files

To add a new file to the repository, you will have to create it in your shared project first. The new file will, again, appear with a question mark.



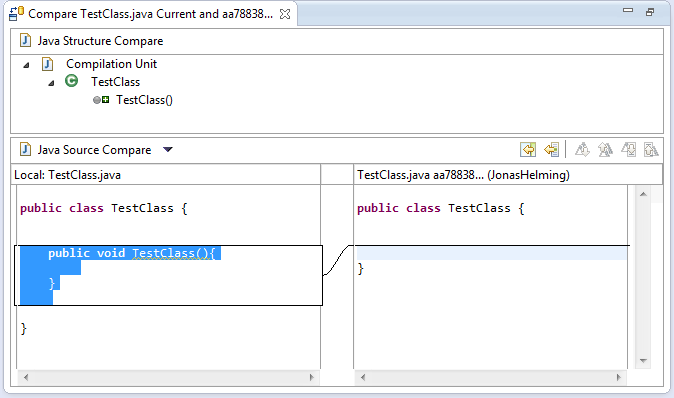
Right click it and navigate to Team => Add. The question mark will turn into a plus symbol and the file will be tracked by Git, but it is not yet committed. All of the file’s parent folders should now have a symbol that looks like an asterisk indicating that it is ‘staged’. In the next commit, the file will be added to the repository and the plus symbol will turn into a repository icon. The repository icons of all the file’s parents (packages/project…) will turn into staged icons. EGit also allows selecting untracked files to be added in the commit dialog if you turn on the option “Show untracked files”. In this case, they will be added and committed at the same time.

[](https://eclipsesource.com/wp-content/uploads/2012/12/8.png)

## Reverting Changes

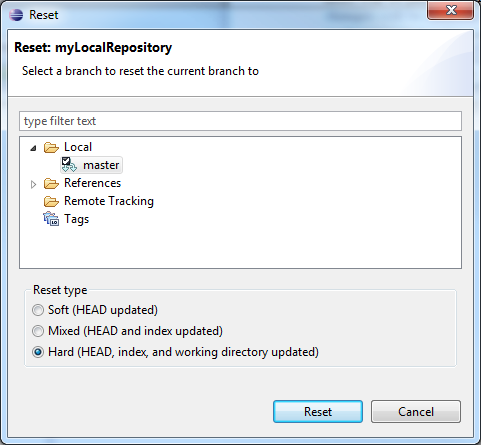
### Revert via Compare

Right click the file you want to revert and select Compare With => HEAD Revision. This will open a comparison with the HEAD Revision, highlighting any changes done. If you want to completely revert your file, hit the Copy All Non-Conflicting Changes from Right to Left-button in the Java Source Compare toolbar. If you only want to revert several lines, select each line individually and hit the Copy Current Change from Right to Left button (in the toolbar) for each line. To complete the Revert operation, you will have to save either the comparison or your local copy of the file.



### Revert via Reset

To reset all changes made to your project, right click the project node and navigate to Team => Reset… . Select the branch you want to reset to (if you haven’t created any other branches, there will be just one) and choose Hard as a reset type. By confirming this operation, all changes will be reset to this branch’s last commit, including all changes done in the workspace (and index, more on that in the section “index”). Be careful with this option as all changes in your workspace will be lost.

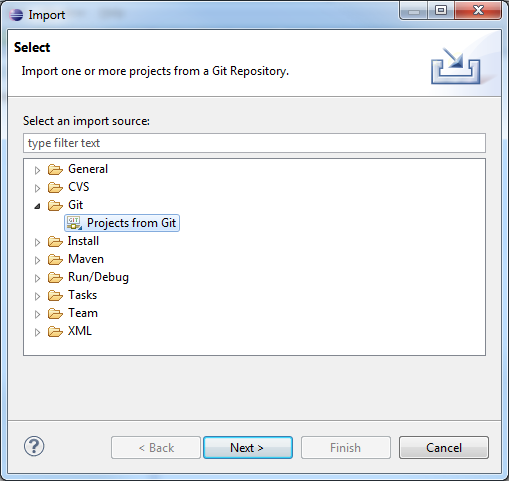


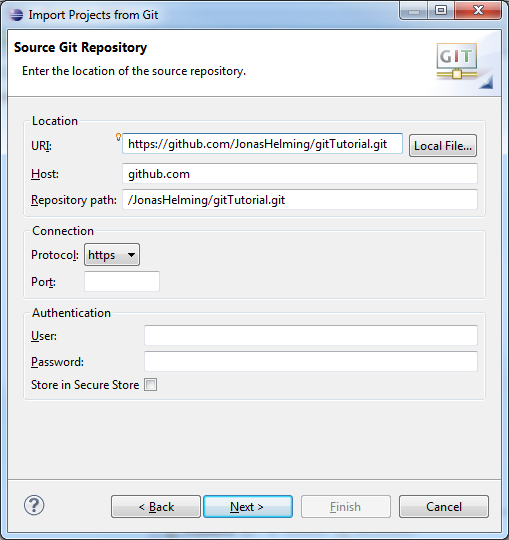
## Cloning Repositories

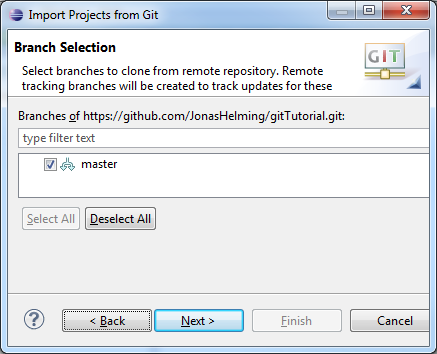
In order to checkout a remote project, you will have to clone its repository first. Open the Eclipse Import wizard (e.g. File => Import), select Git => Projects from Git and click Next. Select “URI” and click next. Now you will have to enter the repository’s location and connection data. Entering the URI will automatically fill some fields. Complete any other required fields and hit

Select all branches you wish to clone and hit Next again.

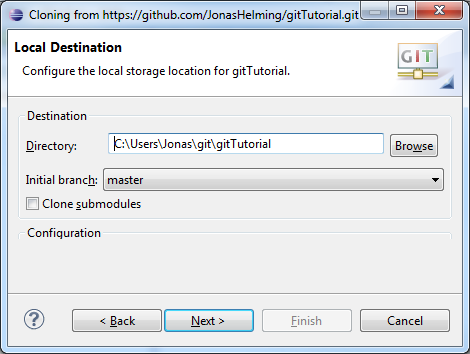
Hit the Clone… button to open another wizard for cloning Git repositories.



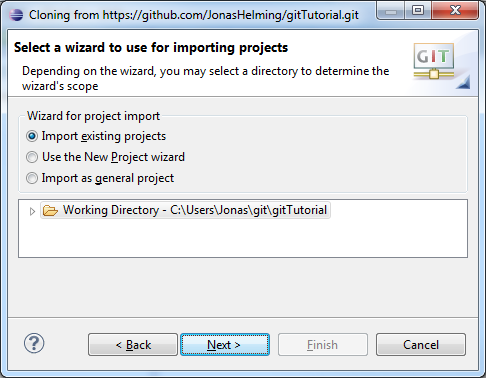




Choose a local directory to save this repository in.

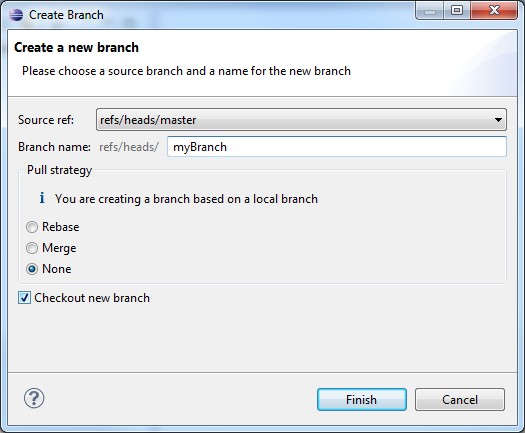


To import the projects, select the cloned repository and hit Next. Select Import Existing Projects and hit Next. select all projects you want to import and click Finish. The projects should now appear in the Navigator/Package Explorer.

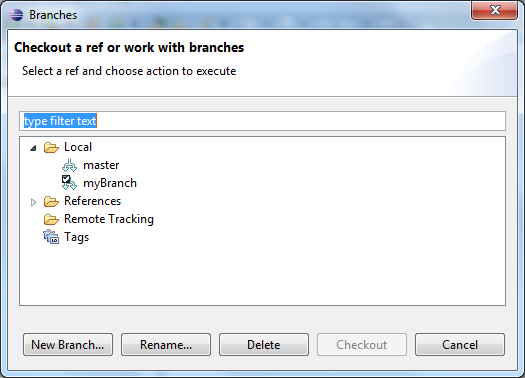


## Creating Branches

To create a new branch in your repository, right click a shared project and navigate to Team => Switch to => New Branch… from the context menu. Select the branch you want to create a new branch from, hit New branch and enter a name for the new branch.



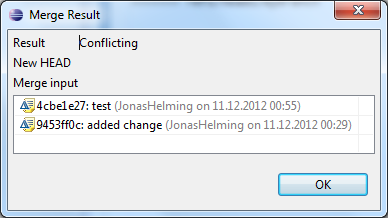
The new branch should appear in the branch selection window. If you would like to checkout the newly created branch, select it and click Checkout.



## Merge

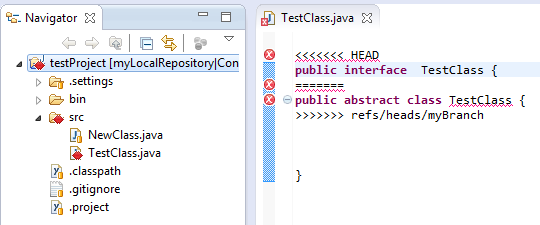
To merge one branch into another, you will have to check out the branch you want to merge with. Right click the project node and navigate to Team => Merge… . Select any branch (other than the checked out branch) and hit Merge.

The merge will execute and a window will pop-up with the results. The possible results are Already-up-to-date, Fast-forward, Merged, Conflicting, Failed. A Conflicting result (see image below) will leave the merge process incomplete.



## Resolving Conflicts

If your merge resulted in conflicts (note the red symbols on the file icons), you will have to resolve these manually. Open the conflicting files and scroll to the conflicting changes marked with “<<<<<<<”.



After you are finished the manual part of the merge, you will have to tell Git that the conflicts are resolved. To do so, Add the files and Commit to complete your merge.

## Fetch and Pull

When cloning remote repositories, Git creates copies of the branches as local branches and as remote branches. A Fetch operation will update the remote branches only. To update your local branches as well, you will have to perform a Merge operation after fetching. The operation Pull combines Fetch and Merge. To perform a Fetch, select Team => Fetch From… from the project’s context menu. Enter the repository you want to fetch branches from. (If you cloned this repository, the remote branch will be selected as default.) In the following window you will have to select what you want to fetch. As default, all branches are selected. The result of the Fetch-operation will be shown in a final confirmation window. Follow the same steps to apply a Pull.

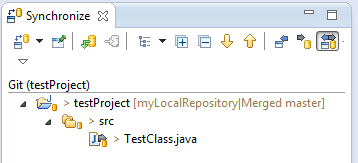
## Push

Local changes made to your local branches can be pushed to remote repositories causing a merge from your branches into the branches of the remote repository (X pulls from Y is the same as Y pushes to X). The Push wizard is pretty much the same as the Fetch wizard. First, right click the project node and navigate to Team=> Push… . Enter the repository you want to push your branches to (the default for this will be the same as the Fetch default if you didn’t configure a Push default) and hit Next. Choose the branches you want to push or click Add all branches spec if you want to push all branches. You can also select branches you want to delete from the remote repository. If you are done hit Finish. A final window will show the results of the Push.

## Synchronize

Comparisons between your workspace and the local repository or between the current branch and others and are done via the Synchronize operation. If you right click Team => Sychronize Workspace, your local workspace will be compared with the current branch showing uncommitted changes. If you select Team => Advanced => Synchronize… . , you can select other branches to compare your current branch with. In this case you can also include local uncommitted changes.

To compare the branches you may want to switch to the Synchronizing perspective, where you can get a more detailed view of several changes.



## History View

To show any shared file’s history, right click it and select Team => Show in History. This will open the History View, giving an overview of the commits and allowing you to perform several actions (compare, creating branches/tags, reset…). Every commit you select comes up with a revision comment and revision details. The revision comment (bottom left corner) includes parents, children, commit message and changes whereas the revision details (bottom right corner) name the changed files and the actions performed upon them (A=ADD, M=MODIFY, D=DELETE). Selecting a file in the revision details will scroll the revision comment to the changes to that file.

