# Git Cheat sheet for command-line and Eclipse plug-in

### Computer Science 316

### Spring 2015

This cheat sheet only contains the basic commands. There are many more commands. Check on-line help for more information. For more detail in using the command-line, I recommend Pro Git, available online at http://git-scm.com/book/en/v2. For more details on using the Eclipse plug-in, see its User Guide at http://wiki.eclipse.org/EGit/User_Guide.

## Getting started

### Creating a repository

One person in your group should create a repository on github.com. To do this:

- login to github
- Click on the + next to your name and select New Repository
- Give your project a name and description. The repository should be Public. Check the box next to "Initialize this repository with a README"
- I recommend that you select the GNU Affero GPL v3.0. You can read more about the license at http://www.gnu.org/licenses/agpl-3.0.html.

### Adding teammates as collaborators

Next, the person who created the repository should add her teammates as collaborators on the project. This will allow them to push their changes to the repository. While on the project page, do the following:

- Click on Settings on the right side of the page
- Click on Collaborators in the left column.
- Enter each of your teammates github usernames in the box and click Add Collaborator.

### Cloning a repository

Every person working on the project needs to clone the repository to get a copy of it on her computer. You will only need to do this once (unless you get a new computer or delete it from your computer, etc.)

To get a copy of a repository that exists on github onto your computer, you **clone** it.

Command-line: git clone https://github.com/blernermhc/MyTest

Eclipse:

- File menu -> Import -> Git -> Projects from Git -> Clone URI.
- Paste in the URI to your github repository.
- Authenticate to github. Select Store in Secure Store you will not need to reauthenticate to github later. Next.
- Select master. Next.

- Browse to the directory where you want to keep the checked-out project. This should be within the Eclipse workspace. Finish.
- Use the New Project wizard - Java - Java project (assuming you are coding in Java).
- Type in the project name. **Be sure that the location shown here is the same as the one you entered in the Browse step above.** Finish.

## Changing a project

The normal workflow for making changes in a project that you have cloned is this. Each of these steps is outlined below.

1. Create a branch where you will make your edits.
2. Repeatedly edit, test, commit and push. Every time that you get even a small piece working, you should commit and push. That will keep your latest work on the github server just in case your laptop stops working.
3. When you are ready to make your changes part of the official version, you will merge it into the master and commit and push the master.

### Creating a new branch to work on

Do this to keep your changes isolated from others on your team. In this example the branch name is “gui”.

Command-line: git checkout -b gui

Eclipse:
- Right-click on the project. Select Team — Switch to… — New Branch
- Be sure the checkout new branch box is checked.

### To do a commit

Command-line:
1. Run the “git status” command
2. Notice which files are listed as “not staged for commit” but that you intend to commit.
3. Run the “add <filename>” command for the files you want to commit.
4. “git -v commit” - opens a text editor showing you the changes to the files.
5. Add a commit comment.

Eclipse:
- Open the Window Menu. Select Show view… Other…. Git - Git Staging
- In the Git Staging view, clicking on an an Unstaged or Staged file will show you how the file has changed. Right-clicking an Unstaged file lets you stage it. There is also a pane to add your commit comment. When all done, click Commit to commit locally, or Commit and Push to share the changes on the github server.

### To put your changes on the github server

Command-line: git push

Eclipse:
- Right-click on project or file. Team menu - Push to upstream

- Note above, that you can commit & push in one step from the Staging View.

### Merging a branch into master

You need to switch to the master branch, pull down any changes others may have pushed to master and then merge in the changes you made. This example assumes you want to merge in the gui branch.

Command-line:
1. git checkout master
2. git pull
3. git merge gui

Eclipse:
1. Right-click on the project. Select Team — Switch to…
2. Select master from the menu.
3. Right-click on the project. Team Menu - Pull
4. Right-click on the project. Team Menu - Merge…
5. Select the Local branch gui.

**Remember to commit and push the changes after merging them into the master!**

## Other useful commands

### Checking the status of your files

Command-line: git status

Eclipse: look at the icons attached to the files in the Package Explorer. ? means untracked. * means modified. Repository icon (gold rectangle) means committed.

### Finding out the changes in each file since the last commit

Command line: git diff `HEAD`

Eclipse: Right-click the project. Compare With menu - HEAD revision

### To review the log history

Command-line: git log

Eclipse: Right-click on project or file. Team menu - Show in history.

### To bring changes from github to your repository copy

Command-line: git pull

Eclipse: Right-click on the project. Team Menu - Pull

### Tagging a version

It is a good idea to tag the version of a system that you show to the client so that you can be certain which version their comments refer to. In this example, we are creating the tag “v1.0”.

Command-line:

1. git tag -a v1.0 -m “Version shown to Julie on Feb. 14”
2. git push origin v1.0

Eclipse:
1. Open the Window Menu. Select Show View — Other… — Team — History
2. Right-click on a Commit and select Create tag…
3. Use the Create tag and start push button.

### To work on a specific tagged version

Command-line: git checkout -b Version2 v1.0

Eclipse:
- Right-click on the project. Select Team — Switch to… — Other… — Tags

### Switching to a branch that already exists

Command-line: git checkout gui

Eclipse:
- Right-click on the project. Select Team — Switch to…
- Select the desired branch from the menu.

### Resolving merge conflicts

When you merge two branches, it is possible that there may be conflicting changes. In that case, git will add lines to the files with conflicts to highlight the changes, showing you both versions.

A file that contains merge conflicts will look like this. The <<<< and >>>> lines show you the beginning and end of the conflict. The ==== line separates the conflicting lines of the two versions. You will need to use an editor to decide which lines you want to keep in the merged file and remove the others, and removing the <<<<, >>>> and ==== lines.

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

You can also use “git mergetool” to help you find and resolve the conflicts. The mergetool is also available in the Team menu in Eclipse.

After resolving the conflicts, you need to add and commit the changed files.

To determine which files have conflicts:

Command-line: git status

Eclipse: Files with merge conflicts will have a red arrow icon on them.