GIT Cheat Sheet

Make changes

Workspace -> Index –> Local Repository -> Remote repository



In terminal

Git init

Git add *filename*

Git add filename\_1 filename\_2

Git add => Sends files to staging

Git reset HEAD filename => Resets file in staging to the same as in the head commit. Does not delete changes, but removes the file from the staging area.

Git commit (add –m “This is the comment for the commit”) => Sends files to final repository

To go back to an earlier stage, go to the log and type:

Git Log to see previous commits

Git reset *first 7 letters/numbers of log*

HEAD is the most recent commit.

After reset, HEAD moves to an earlier commit.

“TIME TRAVEL”

* git checkout HEAD filename: Discards changes in the working directory.
* git reset HEAD filename: Unstages file changes in the staging area.
* git reset SHA: Can be used to reset to a previous commit in your commit history.

You can use different branches for different outcomes. Now I’ve only used the master branch.

To find out which branch you’re on, use:

Git branch

Git branch *branch\_name* => Make new branch

Git checkout *branch\_name* => Change to other branch. This does not affect master branch

Git merge branch\_name => Merges branch\_name to branch you are currently on(master)

There can be a *merge conflict* if you try to merge two branches with changes in the same area. Git does not know which changes to take into account.

Git adds >>> MASTER and other text to the file to show which one you want to keep.

Branches are used to add new features, which can then be added to the master branch.

Git branch –d *branch\_name* => delete branch name



Working with a team

To pull from a server you write

Git clone *remote\_location clone\_name*

Git remote –v => Find the origin of cloned repositories

Git fetch => Gets files from remote repository and puts them on a new, remote branch in your local repository. Used if the code has been changed and you want to know what’s been changed.

Git merge origin/master => Get your code up to date with first fetch from server then merging your master with it.

**Typical approach:**

* Fetch and merge changes from the remote
* Create a branch to work on a new project feature
* Develop the feature on your branch and commit your work
* Fetch and merge from the remote again (in case new commits were made while you were working)

Push your branch up to the remote for review

Git push origin <branch name> pushes branch to the origin remote