**git clone <some git url>**

-makes a copy of repository

-stores in your computer

-fork a repo (make your own copy of someone’s repo)

**Git add <file name>**

-adds a file to the “staging area”

-tells git to include the file in the next revision to the repo

-**git add \*** adds all changes to all files

**git diff**

-shows all the changes you made to the file

**git commit -m “message”**

-saves the changes to the repo as a new revision (in a commit)

-records a message

**git commit -am “message”** adds and commits in the same step

**git status** to check the current status

**git push**

-sends committed changes to the remote repo

-more explicitly could write

**git push origin master**

**git pull**

**-**pull files from github (receives changes from remote repository)

**merge Conflicts**

-when two different commits can’t be automaticlally merged

-need to be resolved

**git log**

-shows a history of commits and messages

**git reset**

**-git reset --hard <commit>**

reverts code back to a previous commit

**-git reset --hard origin/master**

reverts code back to a remote repo version

Branching

-branch is a version of the repo

-Each branch has its own commit history and current version

**git branch**

* Shows all the branches of code
* Create a branch with

**git branch <branch\_name>**

* Switch to (”checkout”) a new branch with

**git checkout <branch\_name>**

-to checkout and create a new branch

**git checkout -b <branch\_name>**

**git merge**

**-git merge <branch\_name>** merges

the branch branch\_name with the current branch

**git merge branch\_name** merges the new branch\_name file to the master

If you want to delete the branch\_name you created

**git branch -D branch\_name**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**#codeacademy (level 1-Real Git)**

**Setting up git:**

**git config --global user.name “Ghezae Fisseha”** who gets credit for the change

**git config --global user.email “**[**ghezae15@gmail.com**](mailto:ghezae15@gmail.com)**”** what email you use

**git config --global color.ui true** pretty command line colors

**git add file1.txt file2.txt**

**git add --all**

**git add <list of files>** add a list of files

**git add \*.txt** add all txt files in the current directory

**git add docs/\*.txt** add all txt files in docs directory

**git add “\*.txt"** add all txt files in the whole project

For adding both files to the staging area before commit

**git status** to check the files ready to be committed

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**git lot --stat** gives statistics which files changed in each commit

**git reset --soft HEAD^** keeps the changes without committing ( if you want to change the commit you made in the file without- no change is done to the file)

**git reset --hard HEAD^** revert back to the file and avoid the changes you made

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Let us say we two branchs “**master**” and “**unicorn**”:

we wanna get out unicorn branch up-to-date

we first checkout the unicorn branch **git checkout unicorn** (switched to branch ‘unicorn’

and then we run **git rebase master** (to check how **rebase** works)

It will move the unique commits in **unicorn** branch to a temp area

And the and then the **unicorn** gets updated from **master** branch and then the commits from the temp area are moved back to the **unicorn** branch

\*If you want to alter the commits that you did in the same branch you use

**git rebase -i HEAD~3** (**i** stands for interactive and **~3** for 3 commits before the current head)

**git rebase -i HEAD** - does nothing

**git rebase -i HEAD^**  got the parent head (checks the last commit)

**git rebase -i HEAD~3 (swap the order of commits in a branch)**

**CHANGE MESSAGES**

**Change the keyword ‘pick’ to ‘reword’**

If we want to split a single commit to 2 or 3

Inside the rebase **‘pick’** to **‘edit’**

**SQUASH THE COMMITS**

**git rebase -i HEAD~3** - change the keyword **‘squash’** instead of pick