If you’re creating new repository and then pushing to github, then

* Git init .

// this will initialize a new git repository

* Git remote add {remote\_name} {url}

// url of your git repo will be added as a remote

// remote\_name = origin, if your/forked repo url

// = upstream, if your team’s repo

If you’re cloning repository

* Git clone {url}

// will create folder of repo from github url

// git is automatically initialized

General

* git remote –v

//check remotes

* git branch

//names all branches

* git checkout –b {newbranch} {branch}

// will create new branch which is at par with branch

//Note: changes in branch wont be pushed directly to github unless same named branches exist there

* git checkout {branch}

// switches to that branch

// Note: commit changes in current branch before checking out, or changes will be reflected in switched branch

* git branch –d {branch}

// deleted branch

* git diff {file}

//shows changes made

* git add {file} or git add .

//add file/all files for commit

* git status

// shows added/ deleted/ modified files staged for commit

// if red in font color => add files for commit then check status

* git commit –m “name of commit”

//changes committed

// you can keep committing your changes , and at the end push them

//Note: name commits sensibly, helpful when you want to rollback in future

* git push {remote\_name} {branch}

eg. git push origin master

* this will push changes on remote origin – master branch

git push origin feature

* this will push changes to feature branch
* note: if feature branch doesn’t exist on github repo, throws error
* git pull {remote\_name} {branch}

eg. git pull upstream prod

* this will pull everything from prod branch of remote upstream to your branch
* git log --oneline

// shows all commits till now along with where HEAD and remote is

* To merge branch feat2 into feat1

Git checkout feat2

Git fetch --all // this will fetch and keep all

commits of feat2

git checkout feat1

git merge feat2 // this merges feat2 into feat1

* Git reset --hard {commit} or HEAD

// you use this when you want to rollback to a previous commit

eg . git reset --hard a2b2a2

// this will rollback your branch to commit number a2b2a2

//you can get the commit name/number from git log --oneline

// HEAD is where your last commit was.

Suppose you want to discard all changes you made since your last commit, recheck where your HEAD points , then simply reset head