Using git on your machine

Before you do any git:

git config --global user.name [your username as it will appear in logs]

git config --global user.email [email address where pull requests and comments will be sent to]

Setting up a git repo on your machine:

git init — this command is only for when you start

git add — for adding new or modified files or directories

git status — makes sure we've got everything staged that we want to pack

git diff — for when we want to see the exact changes

git commit — we start to zip up the suitcase

iesc:wq — command for leaving a note about what's inside the suitcase (what's been changed)

Adding to your existing repo:

make files, changes, etc

git status – shows us what we have done and what is being tracked or not tracked

git add – alternatively, if you have only updated files and not created new ones, you can instead use `git add .`
git ignore – tells git to mind its own business and not nag you about your password file

git rm – removes a file from being tracked

git commit

iesc:wq — leaving a note about what's been changed

repeat as needed

Making and using branches:

git branch newbranch — creates a new branch named newbranch

git checkout newbranch — moves you (HEAD) to newbranch

make your changes

git add — for adding new or modified files or directories

git status — makes sure we've got everything staged that we want to pack

git diff — for when we want to see the exact changes

git commit — we start to zip up the suitcase

iesc:wq — command for leaving a note about what's inside the suitcase (what's been changed)

git checkout master — moves you (HEAD) to master

git merge newbranch — merges the changes you made in newbranch into the master branch

if there are merge conflicts do the following

git diff – this will show you the files with merge conflicts. Let's say there is a conflict in file.rb

vim file.rb

you will see lines that show which file the diffs came from - delete the lines you don't want to keep, then save the file.

git commit -a -m 'merged newbranch'

Creating a repo in GitHub and pushing your existing repo to it:

Log in to GitHub - on the top right, hit the plus sign and select Create New Repository

Give your repository a name – do not initialize with a README – remember, you already did git init locally

copy the HTTPS or SSH link for the repo

In Terminal,

git remote add origin https://YourUserName@github.com/User/Project.git – this sets up your github repo as "origin" on your machine

git push origin master -- if you are on the branch newbranch, you would type `git push origin newbranch`, which would create and push newbranch to your github repository
Now, after each commit, you will want to run `git push origin master` to push your changes to your repo.
After you merge someone's pull request or make any other changes in GitHub, run `git fetch origin`

Contributing to someone else's repository:

set up:

GitHub fork button the repo you would like to work on in GitHub

copy the HHTPS or SSH link that you get of the forked repository (the project name will be the same, but you will see your GitHub name in front, instead of the project owners

On your machine, run

git clone https://YourUserName@github.com/User/Project.git --to pull a copy to

your machine and add the repo as "origin"

Go to the original project in GitHub and copy the HHTPS or SSH link

on your machine, run `git remote add upstream https://YourUserName@github.com/OriginalProjectUserName/Project.git– this adds the original project under the nickname "upstream"

contribute:

git fetch upstream – this will pull and merge any changes that have been made to the original project. Doing this regularly ensures that you catch conflicts early and don't code yourself into a corner or something

make your changes, branches, and push to origin as you normally would.

when your feature/bug fix is complete, go to the upstream repository in GitHub and click the create pull request icon

select the repos and the branches you want to merge from and to-- BASE = upstream repo (you want to get to home base) HEAD = your forked repo (on your own shoulders)

After reviewing and commenting on your pull request, the repo owner will (hopefully) merge your commit

INVESTIGATION & OTHER COMMANDS:

git help [command name]

git [command name] --help

man git-[command name]

git rm [file]

git tag [tag name]

git branch (current branch indicated by *)

git checkout [branch-name]

git checkout -b branch-name

git branch -d branch-name