**Git Intro, Basic Commands**

1. **Terms:**

* Directory 🡪 Folder
* Terminal or Command Line 🡪 Application that runs on your computer, which is just an interface for type in Text Commands.
* CLI 🡪 Command Line Interface (typing text instead of pressing)
* cd 🡪 Change Directory (choosing a folder to work with, instead of pressing on one)
* Code Editor 🡪 A place to write code, Word Processor for Writing Code
* Repository 🡪 Your project, or the folder place where your project is kept.
* Github 🡪 A website to host your repositories online

1. **Git Commands:**

* clone 🡪 Bring a repository that is hosted somewhere like Github into a folder on your local machine (bringing it down)
* add 🡪 Track your files and changes in Git (when you created, updated, or deleted files, we want Git to track these changes)
* commit 🡪 We previously know that Git saves the changes we made to our code. We can do so by “committing our changes”, meaning Saving your files in Git. This applies to all changes across various files made.
* push 🡪 Upload Git commits to a remote repo, like GitHub.
* pull 🡪 When there are changes to the code on GitHub, and we want to bring those to the local machine, we use this command. It download changes from remote repo to your local machine, the opposite of push.
* pull request 🡪 changes made in the local machine, need to be requested to reflect the changes on GitHub.

Diagram, box and whisker chart

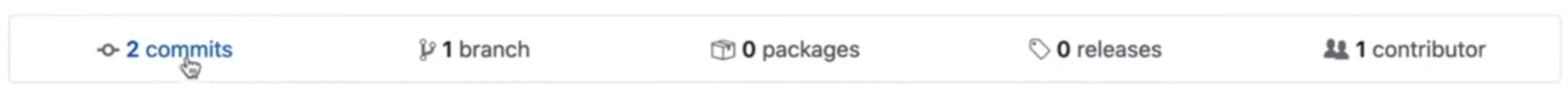
Description automatically generated

1. **Repo started from GitHub**

One can start a project by adding a New Repository.

A file called README.md is usually used for project descriptions. GitHub will usually show this file in the repo main screen.

To see every commitment made: Press on commits



**On GitHub, we don’t need to "add" because it automatically "add" and "commit" as we "commit".**

**\*\*\*Clone working repositories from GitHub**

- To “clone” the folder (repository) down to your local device.

**git clone <SSH> or git clone <HTTPS>**

- Move between directories: use the command

**cd <folderName>**

- Show all files, including hidden files in the directory: (remove -a if we don't want to show hidden files"

**ls -a**

- To show the status of all the files being updated/changed/added:

**git status**

- Note that before committing, we need to tell Git to track all the files and changes made. And we do this using the add command

**git add <file/foldername>** OR **git add .** *(which adds all changes made)*

- To commit files, we MUST add messages. This is similar to the messages on the web interface:

**git commit -m “<title>” -m “<description>”**

**We only need to do both once after we newly created the file. If the file has been tracked (added) before, next time when we have a change, we can simply do**

**git commit -am “<title>”**

- Now that we’ve saved (committed) our code locally, it’s time to reflect these changes on GitHub

**git push origin <branch name>**

where origin is the location of our Git repo; main is the branch that we want to push to

1. **Start a repository from local**

First, we create a different folder for the new repository (or use an untracked folder).

Move into the new folder using cd. Create a new file in the new folder

- Make the folder a Git repository (this will **create a local repo**)

**git init**

- Then we track the file using add, then commit to save the file

**git add .**

**git commit -m “<message>”** *usually suggested as "First commit"*

- Create an empty repository on GitHub.

- Copy the SSH of the new repository, then use:

**git remote add origin <SSH of empty repo>**

This will **link the repo local to the online repo**. Now we can push normally, using

**git push origin <branch name>**

\*\* We can also shorten the work of having to repeat <origin> <branch name> by putting “-u” after push.

**git push -u origin <branch name>**

This will create a default upstream, so that we can later only use

**git push**

to push the file to GitHub.

Graphical user interface, text, application, email

Description automatically generated

1. **Remove version control of a file**

We need to delete the .git folder through terminal (Git Bash)

cd to the git repo, then use the command line

**rm -rf .git**

1. **Switch folder to a new online repository**

If you’d like to keep the version history of your folder, you’ll want remove and replace the existing remote. Git uses “remotes” to configure where it pulls updates from.

Note that removing a remote does not affect your repository—it simply disconnects it from Github, or wherever it’s hosted. You can then add a new remote, pointing to a new Github repository, and push your .git folder to your own repo.

You can list all the remotes alongside the URL they’re pointing to with remote -v:

**git remote -v**

You probably want to remove “origin”, the default remote. You can do that with remote -rm:

**git remote rm origin**

Then, you’ll want to fetch the new remote:

**git fetch origin**

And push the whole folder:

**git push -u origin <branchName>**

If you made changes though, you will need to add and commit those changes before pushing.

Diagram

Description automatically generated