**Git and GitHub**

Purpose: To explain the process and steps for committing changes using GitHub.

What’s Git - A tool used for Version control, which is the practice of tracking and managing changes to software code.

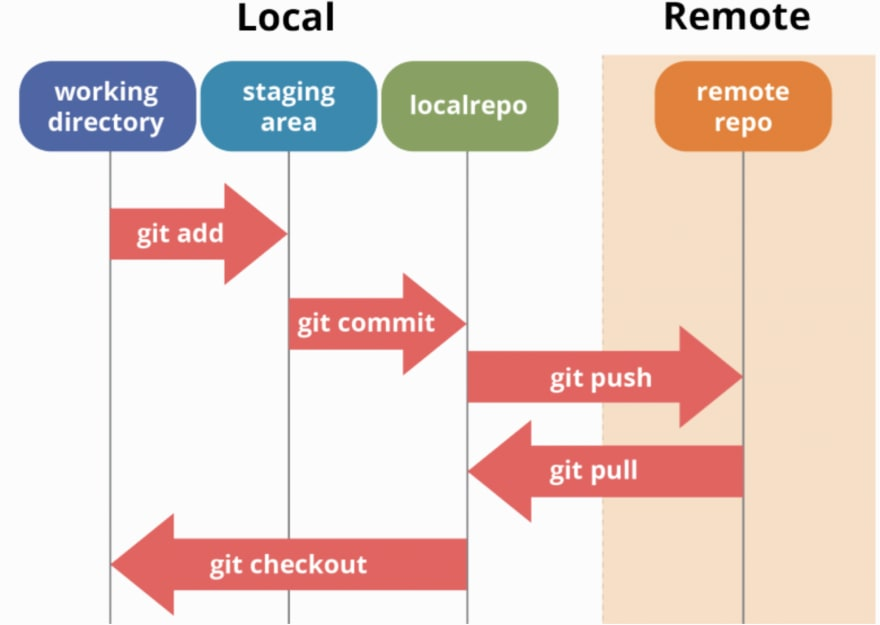
What’s GitHub - Version control and collaboration platform

Git commands

All git commands are run from the terminal. This can either be accessed directly from code editor (eg. Pycharm) or through command prompt in Windows or terminal in Mac. Start by navigating to the folder which you’ll like to track with git.

* In Pycharm use the Python Console
* In windows to access the Command Prompt go to the start menu and type CD to open the window

Overall git flow



* Local Working directory - Directory(folder) with your code/files in your local machine
* Local repo - Repository on your local machine(pc/mac)
* Staging area - Changes that you want to commit eventually
* Remote repository - Where code is stored on cloud
* **git init** - Create a repository (folder with code, aka “repo”) to be tracked by git
* **git clone** – Clone (copy) a repo from a local or a remote repo (eg. located in GitHub) to your local machine(desktop/pc/mac/etc)

Example this command will clone repo from github to local machine

* **git status** - Get status of your changes i.e. what files have been modified, which are in the staging area, which are committed, etc
* **git add** . - Add a change to the staging area

Example: Here we add the test.txt file to the staging area to commit eventually. Alternatively “git add.” will add all the files to the staging area(used with .gitignore)

* **git commit** - Commit your change, usually done with a message describing the change

Example: Here we commit the test.txt file added previously

* **git pull** - Pull changes from remote repo to local repo. This is used to pull changes pushed by a teammate to the remote repo to your local repo
* **git push** - Push changes from local repo to remote repo

Full command : git push origin HEAD:refs/for/<branch-name>

* **git diff** - Shows unstagged changes
* **git log** - Get a history of all commits
* **git branch** - Done to list out all branches in your repo
* **git checkout** - Switch to another branch or create a branch in git, usually done to add a feature to the existing repo and when you don’t want your changes to disrupt the existing state of the repo

Example:

1) Create new branch (notice -b flag)

2) To switch a branch

* **git merge** - Merge changes from one branch to another, this done when your changes have been tested and ready to merge to the main branch

Example: The test file was changed on the fb-test branch and now we want to add those changes to the master branch. First we switch to the master branch(branch we want to merge the changes to) and execute the command

* git revert - Undo changes made by a commit

Example: Here we revert our previous commit. To reference the commit we use the commit hash by checking the logs

* **git reset** - Removes file from staging area without affecting working directory

Example: Here we reset changes made to test file and remove it from staging area

* .**gitignore** - This is a text file which you can place in your directory with file names that you want git to ignore. This is done for logs or local test files which you never want to push to the remote repo

Example: Adding the ipynb\_checkpoints/ files in .gitignore removes them from the status command

Helpful resources: <https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet>

Steps:

git status

git add

git commit -m "<message>"

git push origin main