GIT

What is Git?

- It’s a control version system to keep track of different versions of the files in a project folder.

- When we type our code in a text editor and we save it, **WE OVERWRITE THE PREVIOUS CODE VERSION!!!**

What happens if we want to keep track of previous versions?

That’s where the Git comes to help.

- Git allows you to register the changes in your files

GIT INSTALLATION AND CONFIGURATION

Configuring Git

- In order to configure Git , you need to open your:
    - Terminal in Linux/MacOS
    - GitBash in Windows
- And then you can type the following commands:

```
git # displays a list of basic commands with details

# Configuring git
git config --global user.name "<your name>"
git config --global user.email “<your_email_address>”
git config --list
```

Installing Sublime

- Sublime is a text editor that supports many programming, and Markup languages like Markdown and Python.
- It’s a nice tool also to write plain text.
- Download and install Sublime for your system from here

Configuring Git

- Changing the default text editor to be Sublime (recommended):

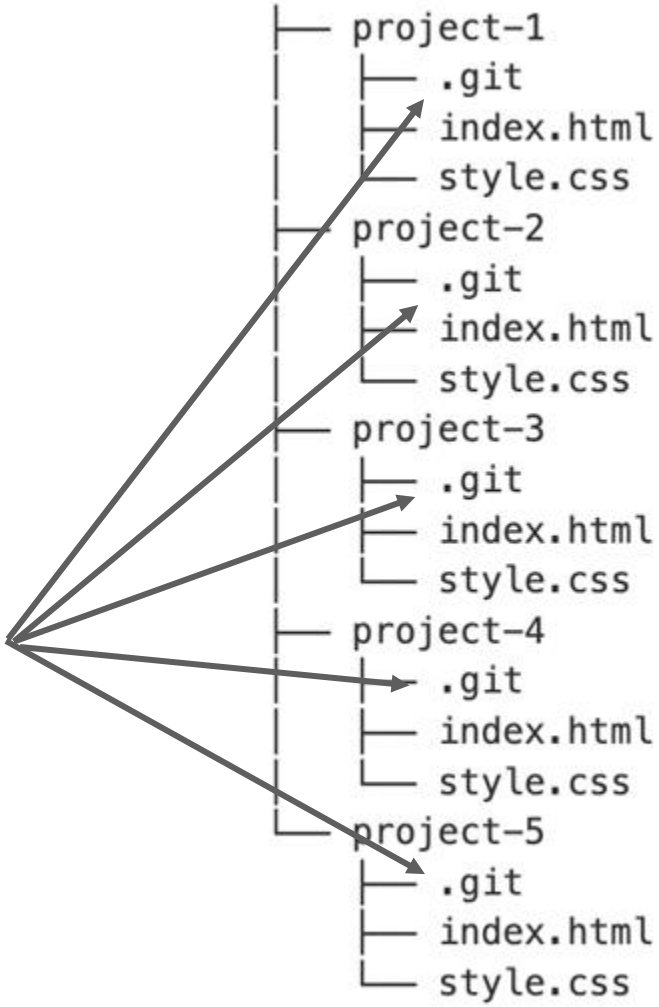
```
# MacOS users:
git config --global core.editor "/Applications/Sublime\
Text.app/Contents/SharedSupport/bin/subl --new-window --wait"

# Windows users type:
git config --global core.editor "'C:\Program Files\Sublime Text
3\sublime_text.exe' -w"
```

- This will popup the Sublime Text editor in some helpful cases.

What is a Git repository?

- Git repository: It's a placeholder to store all the changes in any file of a project folder.
- Each project must have it's own project folder.
- It's a good practice to have every project folder tracked with git independently.
- Each git repo is **independent of each other.**



Creating a local Git repository

- In order to start tracking files with Git locally, you need first to start a “repository” (repo) in a folder of your choice in your computer.
    - It’s very important to start git **ONLY IN THE PROJECT FOLDER, NOT THE MAIN FOLDER. DON’T DO IT RIGHT AFTER OPENING THE TERMINAL!!!!!**
- Open a (Linux Terminal, macOS Terminal, GitBash) and **move to the project folder** in which you want to create your first project.

terminal commands

- Once you are the **desired folder,** we need to instruct Git to start a “repo” (**only do it once for every project**):
    - *git init*

Important notes

- Once a repo is started in a folder:
    - all the files and subfolders within the project folder will be tracked by git (regardless of their type).
    - **we will not have to initialize another repo inside the project subfolder**
- If we want to remove a git repo, we only need to remove the hidden folder: .git located in the parent project folder.

Steps to register file changes with git.

- To check out the what changes we have made, we type:
  - git status

```
~/Desktop/git-practice(master*) » git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        data.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Steps to register files with git.

- We add the files to be registered to the “staging area” using:
    - git add data.txt

```
~/Desktop/git-practice(master*) » git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   data.txt
```

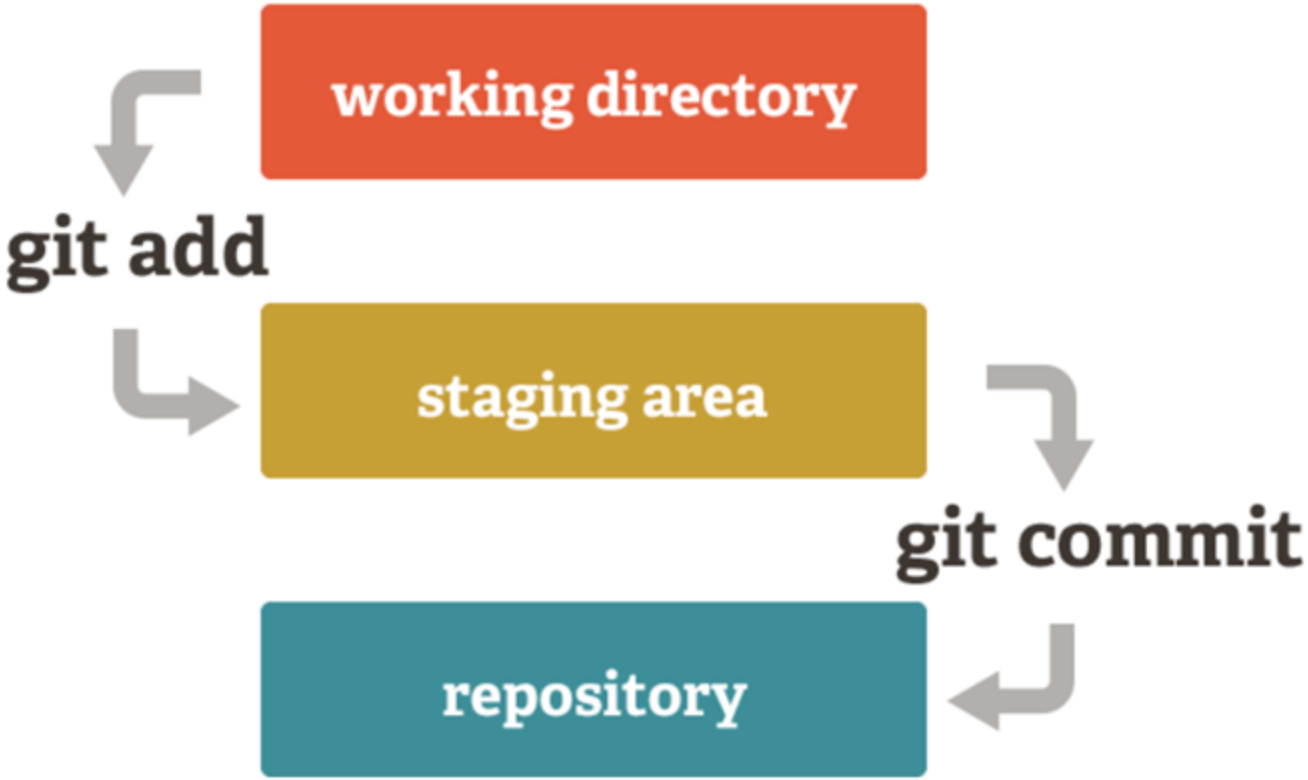
Steps to register files with git.

- If we incidentally added a file into the “staging area”, we can remove it typing:
    - git reset data.txt

Process to create a “commit”

- Once we have files in the “staging area”, we can create an snapshot of the current version of the file in the “staging area” by typing:
    - git commit -m “commit_message_here”
- The commit message should describe briefly the changes made in the file since the last commit.

Summary


working directory
git add
staging area
git commit
repository

History of commits:

- Once we created a “commit”, a history of commits will be created and updated every time a new commit is made.

- We can check this history of commits by typing in the terminal:

    - git log

- All the commits will appear in a chronological order from the most recent to the oldest one.

- Each commit will have:
    - a unique alphanumeric identifier called “hash”,
    - the author:
    - the datetime
    - commit message
    - branch