**Version Control**

Software :- Git

Service Provider :- GitHub,, bitbucket

Commands

Checking Version :- git –version

Creating folder :- mkdir foldername

Listing all the files :- ls

Status of the folder :- git status

Opeinng folder/file :- cd filename/

Initialising the git :- git init

* and after that use git status to chech whether the git repo started or not
* one time per project
* initialises .git -> a hidden folder to keep history of all files & sub-folders

Getting back to previous folder :- cd ..

Showing all the folders :- ls -la(in linux) / Get-ChildItem -Force(in windows)

Opening VSCode :- code .

Creating a text file from terminal

* New-Item -Path "C:\path\to\directory" -Name "filename.txt" -ItemType "File" (in windows) / touch testone.txt (int unix based)
* New-Item -Name "textthree.txt" -ItemType "File"
* HTML file :- New-Item -Path "index.html" -ItemType "File"

Writing content in the file :-

* Set-Content -Path "C:\path\to\directory\filename.txt" -Value "Your text content here"
* "Your text content here" | Out-File -FilePath "C:\path\to\directory\filename.txt" (initialises with the content)
* Add-Content -Path "textone.txt" -Value "This is a new line of text" (adds content)
* Writing code in the file :- code index.html

Moving the file to staging area :- git add filename.ext / git add . (for all file

* Now use git status
* Output maybe:- [

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: textone.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

texttwo.txt]

Committing the file :- git commit -m “comment/message”

Log of the changes :- git log (quiting the page press q)

Log in one line :- git log –oneline

Ignoring files :- git ignore

Setting a git username/useremail :-

* git config –global user.name “firstname secondname”
* git config –global user.email “email”
* git config –global core.editor “code –wait” (vs-code)

gitignore

* Creating :- echo "node\_modules/" > .gitignore
* Appending files
  + Append log files and environment files:- echo "\*.log" >> .gitignore echo ".env" >> .gitignore
  + Append OS-specific files echo ".DS\_Store" >> .gitignore
  + Append IDE-specific files echo ".idea/" >> .gitignore echo "\*.swp" >> .gitignore

Home directory :- cd

Path of home directory :- pwd

Opening gitignore file:- cat .gitignore

Branch in the git :-

* Checking the branch :- git branch (by default “main”)
* Creating new branch :- git branch new\_branch\_name
* Switching branches :- git checkout branch\_name / git switch branch\_name
* Creating a branch and moving there :- git switch -c branch\_name / git checkout -b branch-name
* Deleting the branch :- git branch -d branch\_name
* Detaching head(new branch) :- git checkout <Hash>
* Re-attaching head :- git switch main
* Look at 2 commit prior :- git checkout HEAD~2
* Get back to last commit version :- git restore filename

Knowing the changes :- git diff filename

Stashing

* Opening stash :- git stash
* Poping out of stash :- git stash pop
* Applying stash :- git stash @stash[number]

Rebasing

* Rebasing :- git rebase branch\_name
* Aborting rebasing :- git rebase –abort
* Continuing rebasing :- git rebase –continue

GITHUB

Knowing the remote :- git remote -v

Pushing :- git push origin main

Upstream :- git push –set-upstream origin main / git push -u origin main

Git clone :- git clone <url>

Fetching :- git fetch (get intobut don’t putin my work)

Git pull :- get info and add it to in my work (git fetch + git merge)

Key Points to know

Cd .git

* type ls
* never make changes to these folders

Atomic Commits :- One commit at one time (while writing message write present tense, imperative senternces)

.gitignore

* don’t want to track some files
* node modules, api key, secret
* get template online, patterns, can be tricky



