**HOW GIT AND GITHUB WORKS:**

https://github.com/git-guides/

Diagram

Description automatically generated

Graphical user interface

Description automatically generated

Diagram

Description automatically generated

Install git into your local machine using **git –version**. If it returns the version of git meaning it was successfully installed into your machine.

**Important git commands to remember:**

* git –help – this will return git commands that could help you working with git.
* Ctrl + l or clear – will clear the bash/terminal
* cd .. (go back to root folder)

**CONFIGURE GIT (This is a must if you’re working with new projects)**

* git config --global user.name “name”
* git config --global user.email “email”
* git config –global color.ui auto
* git config –list (To see or check current configuration that you set)

**What are REPOSITORIES in git?** It is a collection of files of various versions of a project.

**Important git commands to remember:**

* cd “path” (to go in that specific directory or path)
* mkdir “foldername” (to create a folder)

**INITIALIZING GIT REPOSITORY**

Why you need to do this? To tell that the working directory is a git repository.

**Important git commands to remember:**

* git init . (after doing this **you must see a “master” or “main” in your bash terminal name)**
* ls -a (this is to list all files or folders inside that repository)
* rm -rf .git (this is to delete git repository)

**GIT ADD –** command to add repository to staging area.

**Important git commands to remember:**

* git add . (this will add all files in the staging area)
* git add -A (this will add all files backward outside of a git repository)

**HOW TO CREATE FILE INSIDE A REPOSITORY AND UNSTAGE FILES**

**Important git commands to remember:**

* touch “filename.extension”
* git status (to see the status of each file inside that git repository)
* git rm –cached “filename.extension” (to unstage specific file)
* git rm -r –cached ““filename.extension” (to unstage all files)

**GIT COMMIT**

**Important git commands to remember:**

* git commit -m =”message”
* git log (to see logs of all commits happened from the repository)
* git log –online (to see logs in a beautiful manner)
* git show “hash” (to see specific has log)
* vi “filename” (to edit a file)

1. Press “I” to insert modifications in that file
2. Press escape to escape insert mode
3. Press colon and wq to close and save file
4. Press enter

* git diff (to see difference of what you have change between current and committed files).
* git restore (helps to unstage or even discard uncommitted local changes)

**AMMEND COMMIT MESSAGES (**Always insert a meaningful commit messages)

**Important git commands to remember:**

* git commit -amend -m “message to replace”

**GITHUB –** Remote repository

* Create GitHub account
* Push and existing repository from command line using the code generated after creating a repository (see below for example)

1. Git remote add origin <https://github.com/AranteAdrian/learning_git.git>
2. Git branch -M main
3. git push -u origin main (this will expect to fail- please follow SSH key setup first)

* git branch – (to check if you are already in the master or main branch)

**SSH KEY SETUP** (You will need to setup this for you to be able to push in the remote repository)

* Generate private and public key. (Follow GitHub instructions)

**GIT PUSH** (Push repository in GitHub)

**Important git commands to remember:**

* git push – origin main (or)
* git push

**GIT PULL** (Pull repository from GitHub to local)

**Important git commands to remember:**

* git pull

**UNDERSTANDING BRANCHES**

**Timeline

Description automatically generated**

**A picture containing bubble chart

Description automatically generated**

**A picture containing bubble chart

Description automatically generated**

**WORKING WITH BRANCHES**

**What is BRANCH?** It represents independent line of development.

**Is Main Branch = Master Branch?** Yes

* git branch (for local – to check which branch you are)
* git branch -r -a (remote – to check which branch you are)
* git branch “new\_feature” (to create a new branch) or git branch -b “new\_branch”
* git checkout – (go to main/previous branch)
* git branch -d “branch\_name” (to delete specific branch)

**PUSH BRANCH TO REMOTE/GITHUB**

* git push origin “branch\_name”

**ADDITIONAL IMPORTANT REMINDERS**

* if you are working in a company, never push changes into the master or main branch directly. Ideal way is to create your own branch, do the changes, push those changes in GitHub (so you don’t lose it) then raise a pull request so that someone can review your changes if it is sensible to merge and will not cause any issue.

**MERGING REQUEST**

Don’t forget to add reviewers. They will check if your merge request does not cause any issue. Apply labels like bug, enhancement, etc. After a successful merge request delete your branch.

**PULL REQUEST GENERAL WORKFLOW**

(This will allow you to pull current changes from remote to your local machine). Pull the latest changes from master, create a new branch, create the new feature and commit it then do merge request and delete your branch. It is advisable to rebase your master branch.

**ANOTHER GIT TOPICS NEED TO LEARN MORE ABOUT**

* Conflicts (Happen when multiple developers are working with specific files)
* The Idea of Merging Conflicts
* Git rebase
* Some git clients (like GITPOD – Cloud version of visual studio cloud)