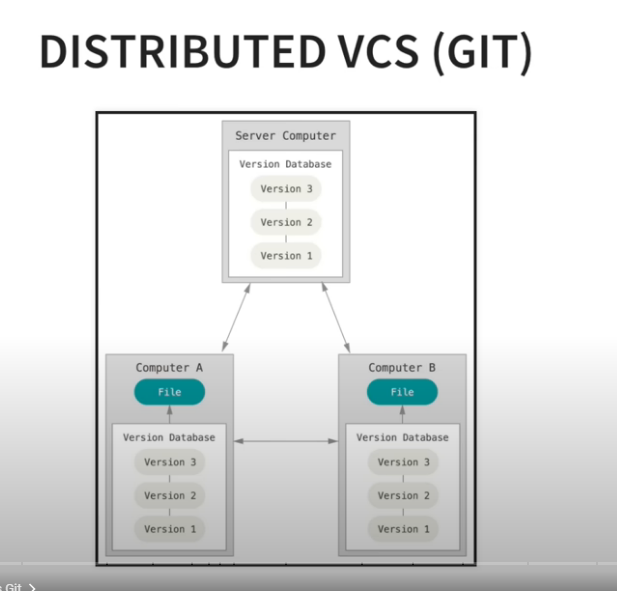
GIT ;

Is a distributed version control system, also has local repository   
  


Install git bash

open git bash and enter   
git –version to confirm the installation

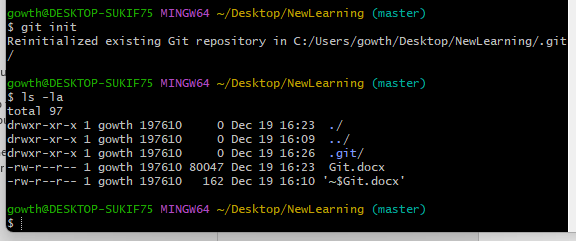
two scenarios..  
1 , will clone the existing repo and do changes

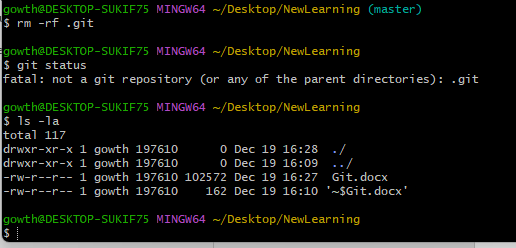
2, create a repo from local and push it to git

We will do second one , we create a local repo first   
  
open git bash from the folder which you want to maintain in git

-git init – will create .git file and it will start recording the changes in the folder

- ls -la – which will list all the file and we can see the .git file also



* To remove tracking , we can remove the .git file with the remove command
* Rm -rf .git(filename) – rm stands for remove and r for recursive delete with the sub folder and directory f for force deleted even when warning pop’s up , even in write mode
* 

Next to ignore the files , which we don’t want to track by repository

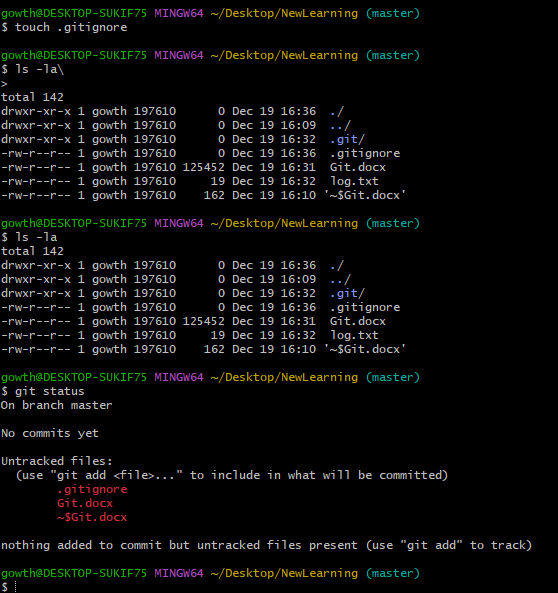
Example : pycache and log files

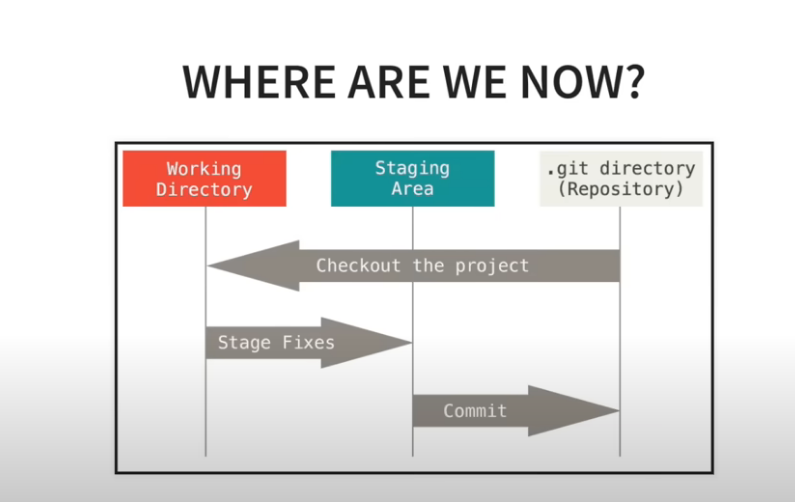
Touch .gitignore

This will create a .ignore file and we can add the file name which needs to be ignored

We can use the wild cards also in the file

\*.log – which will ignore all the log files



Imagine where we are now:

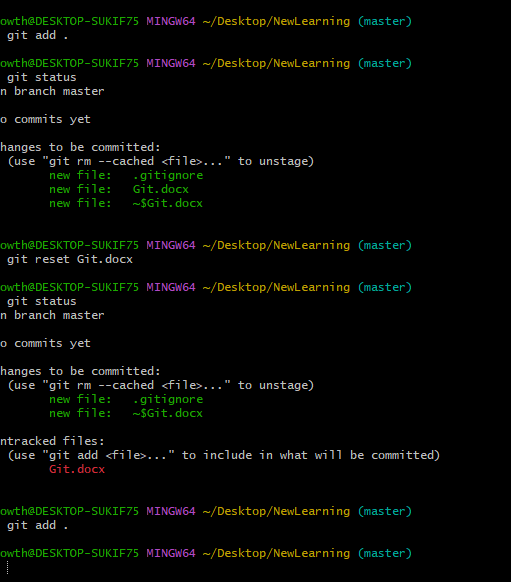
Untracked files and changes will be in the working directory

Git status will return the status of the working directory of untracked files

Why staging area, so we can choose what needs to be committed, make sure u commit the clean code because the report is the history of the final work

Staging the code :

Git add . will add all the untracked files to thestaging area



Git reset. Will unstage all the changes and make it as untracked file again

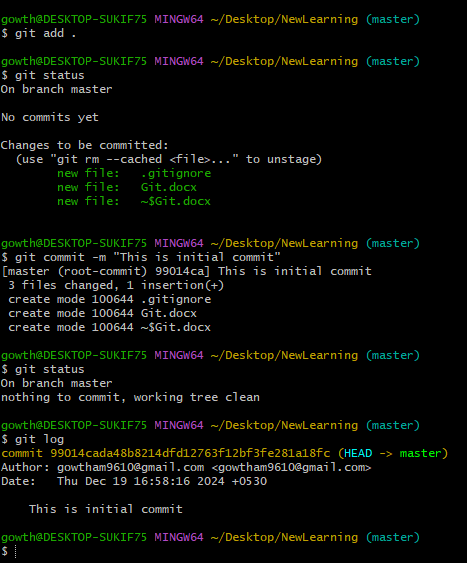
Now .. we commit the code to repo

Git commit -m “this is message ”

This will commit all the code from staging area to repository

We can see the commit log by

Git log



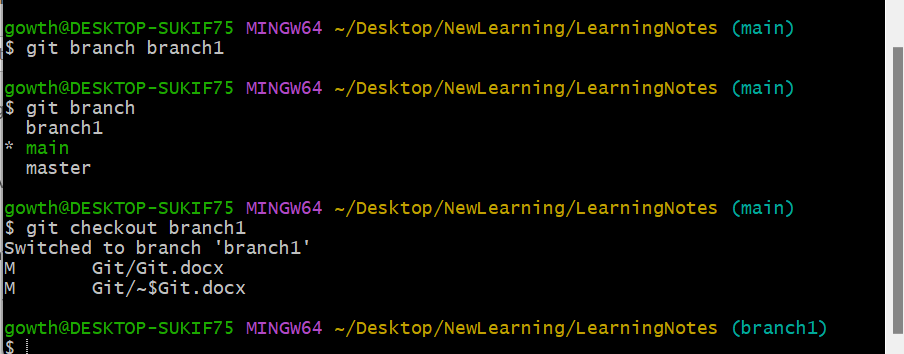
Git diff will show the changes made in the working directory

Git pull origin main - always do it before you push the code , and make sure the code is not modified and get all the latest changes pushed by other developers

And then do git push origin main command to push the code to git

Git Branch :

When we work on new module , we will create a new branch and push the code from that branch , we don’t use the main/master branch



Git branch : will return list of available branch

Git branch branchname : will create a new branch with the given name

Git checkout branch1 : will checkout

**Why Do We Need Branches in Git?**

Branches in Git are a way to work on separate versions of a project without affecting the main codebase. They enable parallel development, safe experimentation, and efficient collaboration.

**Why Do We Need Branches in Git?**

Branches in Git allow you to work on different versions of your project simultaneously. They are a powerful way to isolate changes, experiment with new ideas, collaborate with others, and maintain a clean and organized codebase.

Git push origin branch1

Raise mr request

Make sure branch is deleted if the work is completed

When changes in local then u pull latest code, we will get an error because pull will prevent from overriting the existing code, it will ask you to commit the code first :  
  
3 options are there to resolve this issue :  
  
**Which Option Should You Choose?**

* Use **Option 1** if you want to keep your local changes. – commit and then pull the code
* Use **Option 2** if you’re not ready to commit your changes but want to save them temporarily.- stash the code and then pull the code
* Use **Option 3** if your local changes are unnecessary, and you want to reset to the remote state. – reset the code and pull the code

Using git stash command:

git stash is a Git command used to **temporarily save your changes** in the working directory and staging area without committing them. It allows you to switch to another task (like pulling the latest changes or switching branches) and then return to your stashed work later

**Why Use git stash?**

* **Switching Branches**: You have changes that are not yet ready to commit, but you need to switch to another branch.
* **Pulling Updates**: You want to pull changes from the remote repository, but your local changes might cause conflicts.
* **Interruptions**: You need to temporarily pause your current work to focus on a different task

**Example Workflow**

**Scenario:**

You’re working on a feature in the main branch, and you need to switch to a new branch to fix a critical bug.

1. **Current State**:
   * You’ve modified file1.py and file2.py but haven’t committed the changes.
2. **Stash Your Changes**:

bash

git stash

Output:

plaintext

Saved working directory and index state WIP on main: abc1234 Update feature

1. **Switch to the New Branch**:

bash

git checkout -b bugfix

1. **Work and Commit Changes**:
   * Fix the bug and commit your changes on the bugfix branch.
2. **Return to the main Branch**:

bash

git checkout main

1. **Reapply Your Stashed Changes**:

bash

git stash apply