What is a Git Repository?

A **repository**  is nothing but a collection of source code.

There are four fundamental elements in the Git Workflow.

Diagram

Description automatically generated

**If you consider a file in your Working Directory, it can be in three possible states.**

1. **It can be staged.** This means the files with the updated changes are marked to be committed to the local repository but not yet committed.
2. **It can be modified**. This means the files with the updated changes are not yet stored in the local repository.
3. **It can be committed**. This means that the changes you made to your file are safely stored in the local repository.

Git Terminology

* git add is a command used to add a file that is in the working directory to the staging area.
* git commit is a command used to add all files that are staged to the local repository.
* git push is a command used to add all committed files in the local repository to the remote repository. So in the remote repository, all files and changes will be visible to anyone with access to the remote repository.
* git fetch is a command used to get files from the remote repository to the local repository but not into the working directory.
* git merge is a command used to get the files from the local repository into the working directory.
* git pull is a command used to get files from the remote repository directly into the working directory. It is equivalent to a git fetch and a git merge .

Create a new repository on GitHub.  
  
Now, locate the folder you want to place under git in your terminal.

$ **cd Desktop/DevOps**

Initialize Git:

And to place it under git, enter:

$ **touch README.md** # To create a README file for the repository  
$ **git init**  # Initiates an empty git repository

Now go edit the README.md file to provide information about the repository.

## Add files to the Staging Area for commit:

Now to add the files to the git repository for commit:

$ **git add .**   
# Adds all the files in the local repository and stages them for commit**OR** if you want to add a specific file$ **git add README.md**   
# To add a specific file

## Before we commit let’s see what files are staged:

$ **git status** # Lists all new or modified files to be committed

## Commit Changes you made to your Git Repo:

Now to commit files you added to your git repo:

$ **git commit -m "First commit"**# The message in the " " is given so that the other users can read the message and see what changes you made

## Uncommit Changes you just made to your Git Repo:

Now suppose you just made some error in your code or placed an unwanted file inside the repository, you can unstage the files you just added using:

$ **git reset HEAD~1**# Remove the most recent commit  
# Commit again!

## Add a remote origin and Push:

Now each time you make changes in your files and save it, it won’t be automatically updated on GitHub. All the changes we made in the file are updated in the local repository. Now to update the changes to the master:

$ **git remote add origin remote\_repository\_URL**# sets the new remote

The **git remote** command lets you create, view, and delete connections to other repositories.

$ **git remote -v**# List the remote connections you have to other repositories.

The **git remote -v** command lists the URLs of the remote connections you have to other repositories.

$ **git push -u origin master** # pushes changes to origin

Now the **git push**command pushes the changes in your local repository up to the remote repository you specified as the origin.

## See the Changes you made to your file:

Once you start making changes on your files and you save them, the file won’t match the last version that was committed to git. To see the changes you just made:

$ **git diff** # To show the files changes not yet staged

## Revert back to the last committed version to the Git Repo:

Now you can choose to revert back to the last committed version by entering:

$ **git checkout .OR** for a specific file$ **git checkout -- <filename>**

## View Commit History:

You can use the **git log** command to see the history of commits you made to your files:

$ **git log**

Each time you make changes that you want to be reflected on GitHub, the following are the most common flow of commands:

$ **git add .**  
$ **git status** # Lists all new or modified files to be committed$ **git commit -m "Second commit"**$ **git push -u origin master**

# **gitignore:**

.gitignore tells git which files (or patterns) it should ignore. It's usually used to avoid committing transient files from your working directory that aren't useful to other collaborators, such as compilation products, temporary files IDEs create, etc.

files like \_\_pycache\_\_, .DS\_Store are used by the system to store information for faster access. This is not useful for other collaborators. So we can tell git to ignore them by adding a .gitignore file.

Use the touch command to create the .gitignore file:

$ touch .gitignore

And you can add the following patterns to tell git to ignore such files.

/\*.cmake  
/\*.DS\_Store  
/.user  
/build  
etc. depending upon the files you want git to untrack