**Task 1**

* Demonstrate minimum 15 basic Git command with explanation and screenshot.

**1.git init**

The git init command **creates a new Git repository**. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialized repository, so this is usually the first command you'll run in a new project.

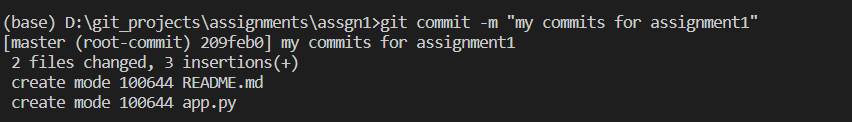
C:\Users\user\OneDrive\Desktop\Screenshot 2022-10-19 104700.png

**2.git add**

 The git add command **adds a change in the working directory to the staging area**. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit .

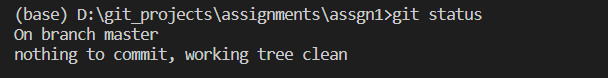
**3.git commit**

The git commit command **captures a snapshot of the project's currently staged changes**. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to.



**4.git status**

The git status command **displays the state of the working directory and the staging area**. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show you any information regarding the committed project history.



**5 git branch**

List all of the branches in your repository. This is synonymous with git branch --list.



**6 git branch <BRANCH NAME>**

Create a new branch called ＜branch＞. This does not check out the new branch.

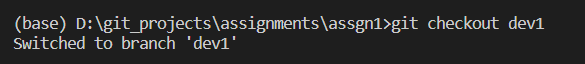
**7 git branch –d <BRANCH NAME>**

Delete the specified branch. This is a “safe” operation in that Git prevents you from deleting the branch if it has unmerged changes.



**8. git** **checkout <BRANCH NAME>**

The git checkout command **lets you navigate between the branches created by git branch** . Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells Git to record all new commits on that branch.



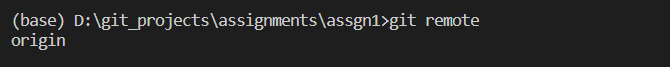
**9. git remote add origin <SSH KEY>**

**Obtain the git remote add URL for the remote repository and add credentials if needed.** Run the git remote add origin command from your local repository with the --set-upstream and the name of the active branch to push.



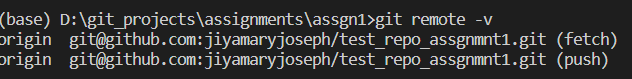
**10. git remote**

List the remote connections you have to other repositories.



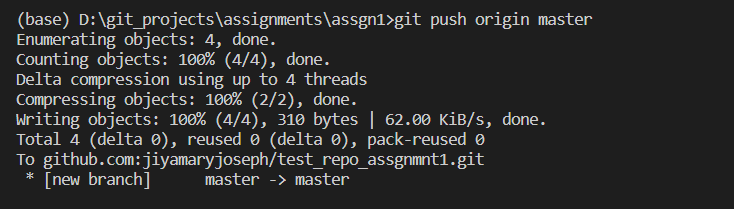
**11. git remote –v**

Same as the above command, but include the URL of each connection.

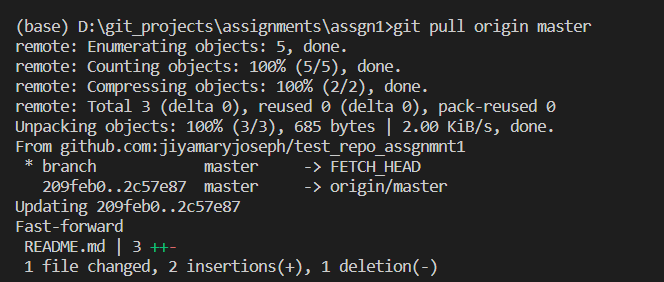


**12.git push <REMOTE NAME> <BRANCH NAME>**

to push your local changes to your online repository.



**13. git push <REMOTE NAME> <BRANCH NAME>**

The git remote command is used to specify what remote endpoints the syncing commands will operate on.

**14. git merge <BRANCH NAME>**

To do a merge (locally), **git checkout the branch you want to merge INTO.** **Then type git merge <branch> where <branch> is the branch you want to merge FROM**.

**15. git fetch <REMOTE NAME>**

This fetch command will fetch all remote branches and also store all references/objects. Once all branches are loaded successfully, you can checkout to the branch you are interested in, giving you a local working copy. Now you can inspect and play with code.