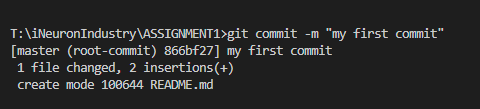
**ASSIGNMENT: 1**

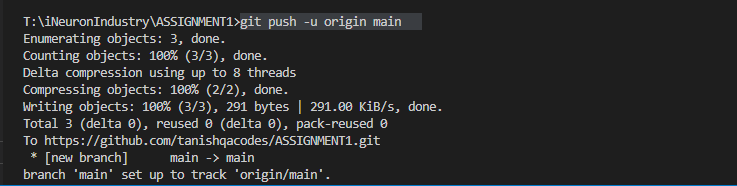
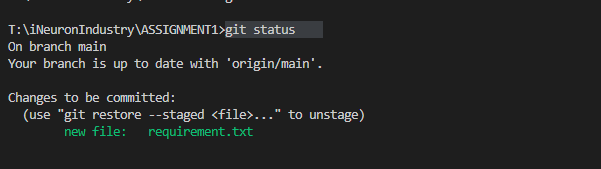
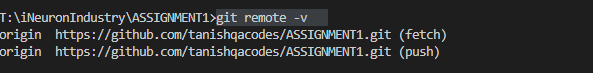
**TASK: 1**

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. 
2. **git config --global user.email:** The global git username and email address are associated with commits on all repositories on your system that don't have repository-specific values. 
3. **git config --global user.name: **
4. **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.
5. **git commit -m "commit message":** An effective Git commit message is a concise explanation of the changes done in the source code. A commit message has a structure like an email subject and body.

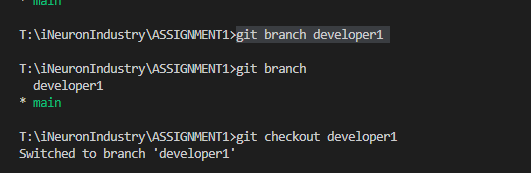
****

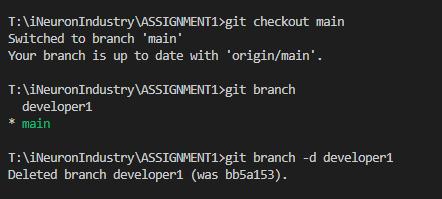
1. **git branch -m main:** Git's branching functionality allows the creation of new branches in a project. These new branches can then be used to test changes to code without affecting the main project code.

****

1. **git remote add origin https://github.com/tanishqacodes/ASSIGNMENT1.git:** git remote manages the set of remotes that you are tracking with your local repository. 
2. **git push -u origin main:** The git push origin master command says "push the commits in the local branch named master to the remote named origin". Once this is executed, all the stuff that we last synchronized with origin will be sent to the remote repository and other people will be able to see them there. 
3. **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. 
4. **git remote -v:** Remote branches are references (pointers) to the state of branches in your remote repositories. They’re local branches that you can’t move; they’re moved automatically for you whenever you do any network communication. Remote branches act as bookmarks to remind you where the branches on your remote repositories were the last time you connected to them. 
5. **git branch:** In Git, branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.



1. **git branch <new branch>:** To create a new branch from the master branch.
2. **git checkout <branch name** >: So, this will create a new branch but still you are not switched to the created new branch. So, to switch to the new branch. ****
3. **git branch -d <branch name>:** To delete a branch, argument ‘-d’ means deleting the branch only if the branch is pushed and merged with the remote branch. So, this is the safe delete option.



1. **git branch -r:** List all remote branch,

****