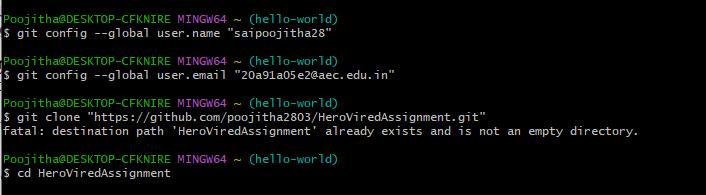
**DevOps Assignment**

**Q1. Describe the usage of the git stash command by using an example and also state the process by giving the screenshot of all the commands written in git bash.**

**Git Stash command**: The git stash command enables you to switch branches without committing the current branch.

Git stash command helps in saving the previously written code and then goes back to the last commit for a fresh start. Now we can add the new feature without disturbing the old one as it is saved locally saved.

After committing the new feature you can go on working with the old one which was incomplete and not committed.



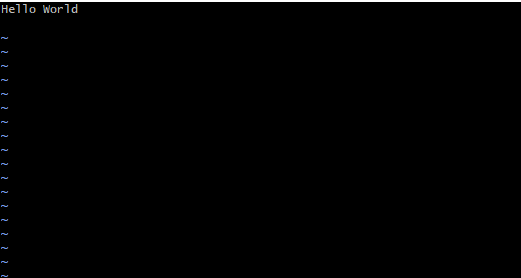
A new directory is created with the name HeroViredAssignment.

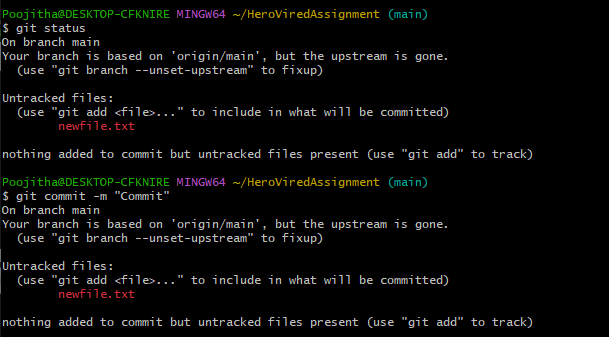
The present working directory is changed to the new directory.



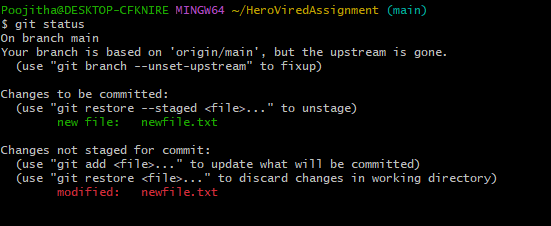
New File is created with the name newfile.txt.

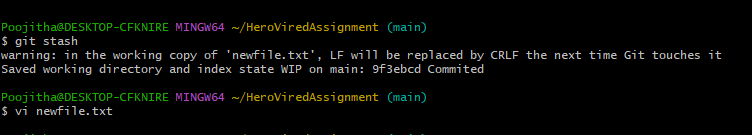
Adding the content into the file makes the changes to the file and gets saved.

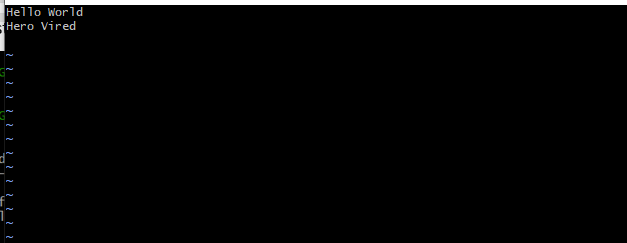




The file is now modified,and it is not committed,now if you want to pull the code on the other branch,then you have to remove these uncommitted changes,so use git stash command.

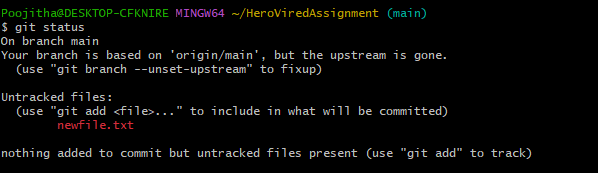






The file is now stashed and it is under untracked state.

By default,running git stash will stash the changes that have been added to your index(staged changes)and unstages changes.To stash your untracked files,use git stash -u.



**Listing stashes:** You can create multiple slashes and view them using git stash list command.

****

**Providing additional message:**

To provide more context to the stash we create the stash using the following command.

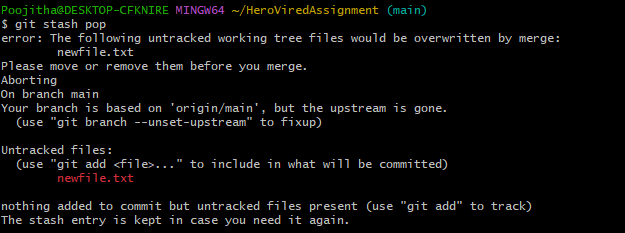
git stash save “message”

**Getting back stashed changes:**

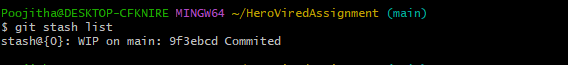
You can reapply the previously stashed changes with the ‘git stash pop’ or ‘git stash apply’ command.

1.‘git stash pop’ removes the changes from stash and reapplies the changes in working copy,

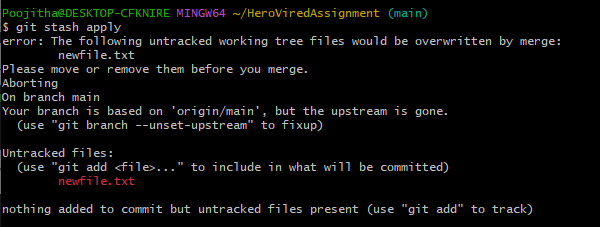
2.‘git stash apply’ do not remove changes .but reapplies the changes in working copy.

****

Now check whether stash is removed or not.

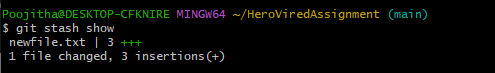


By using “git stash apply” We got the previous uncommitted changes.



**To view the stash summary:**

Git stash show is used to view the summary



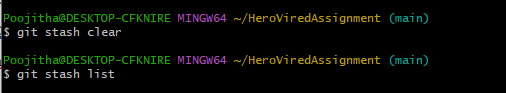
**Deleting stashes:**

To delete a particular stash:

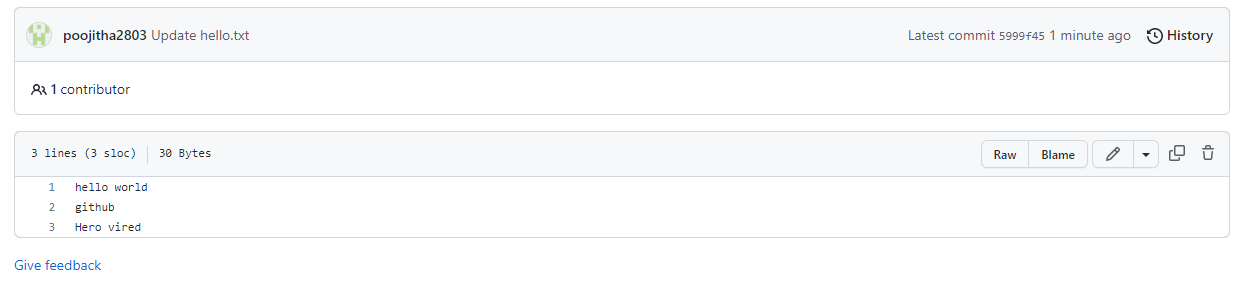
git stash drop stash

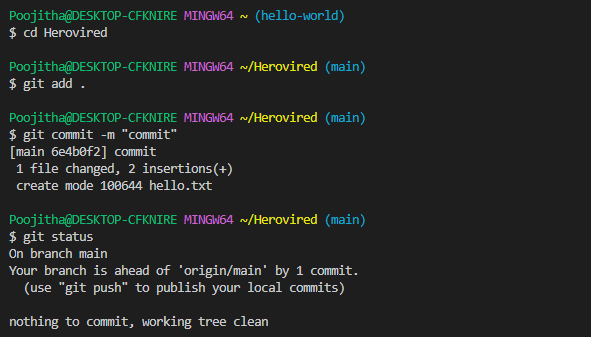
To delete all stashes at once,use the below comman

git stash clear

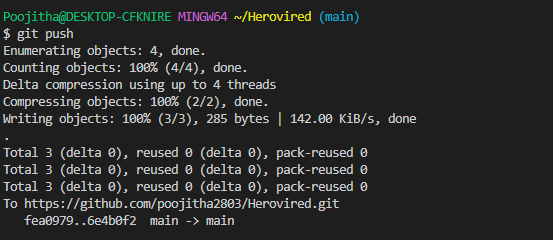


**Q2. By using a sample example of your choice, use the git fetch command and also use the git merge command and describe the whole process through a screenshot with all the commands and their output in git bash.**

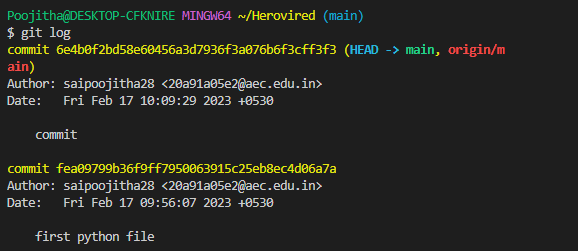
****

****

By using \*git push\* command we push the file hello into the github.



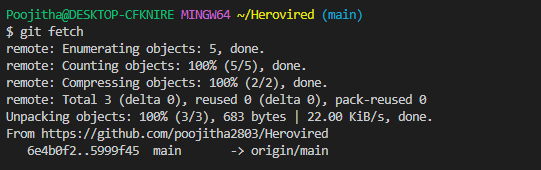
Git log🡪 gives the changes which are made at particular time.

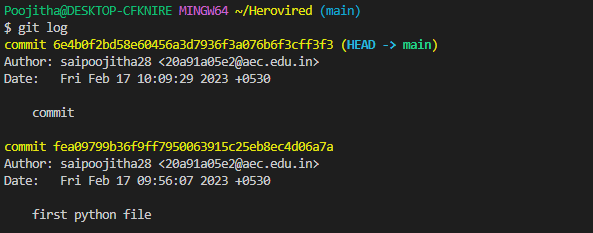


Changes can be made in the file in github directly and then commit.

Now to fetch the changes which are made in the file we use fetch command.

\*Git fetch\*





**Q3. State the difference between git fetch and git pull by doing a practical example in your git bash and attach a screenshot of all the processes.**

**Git fetch:** git fetch is a primary command used to download contents from a remote repository. git fetch is used in conjunction with git remote , git branch , git checkout , and git reset to update a local repository to the state of a remote. The git fetch command is a critical piece of collaborative git work flows.

**Git pull:** The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows.

First, let's use the **git fetch** command. This command will retrieve any changes that have been made in the remote repository since the last time we fetched, but it will not merge those changes with our local repository:

$ git fetch

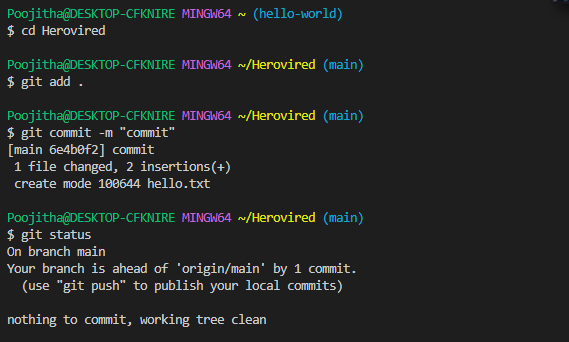
If there are any new changes in the remote repository, this command will download them to our local repository, but it will not update our working directory or merge them with our local changes.

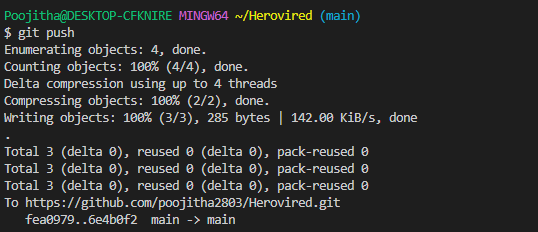
Now, let's use the **git pull** command. This command will retrieve any changes that have been made in the remote repository and merge them with our local repository:

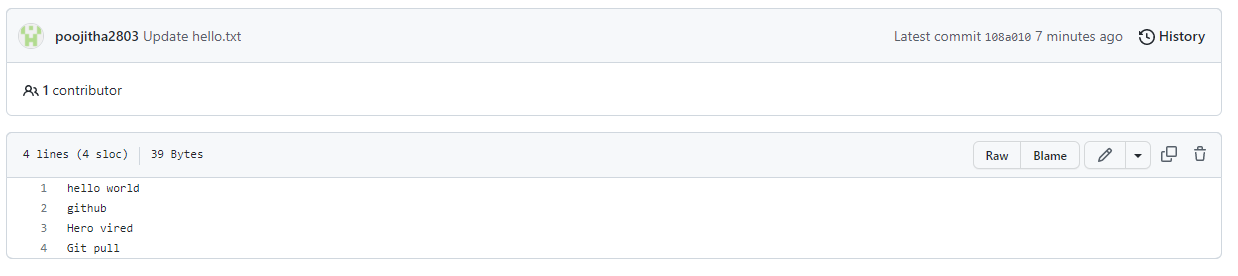
$ git pull

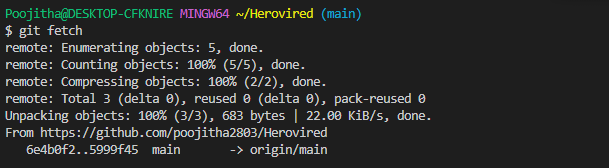
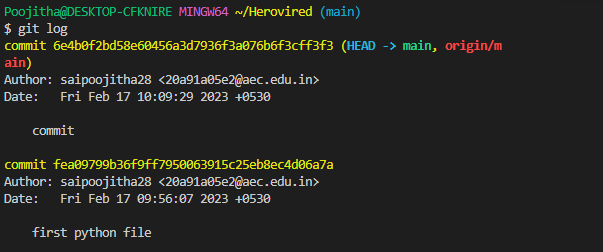
This command will download any changes from the "master" branch of the remote repository and merge them with our local repository. If there are any conflicts between our local changes and the changes from the remote repository, we will be prompted to resolve them.

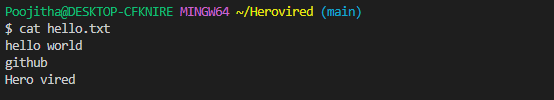
So the main difference between **git fetch** and **git pull** is that git fetch downloads the changes from the remote repository but does not merge them with our local repository, while git pull downloads the changes from the remote repository and merges them with our local repository.

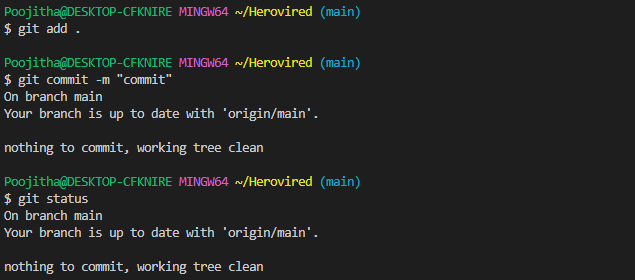


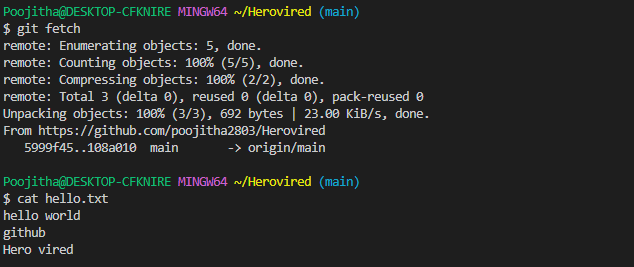


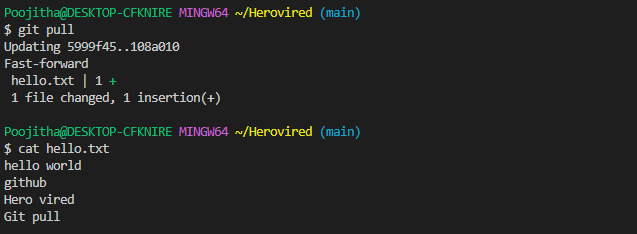








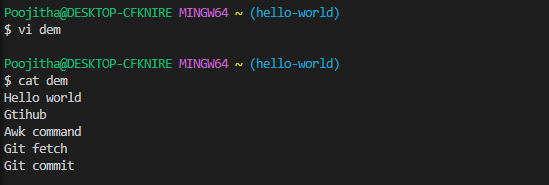




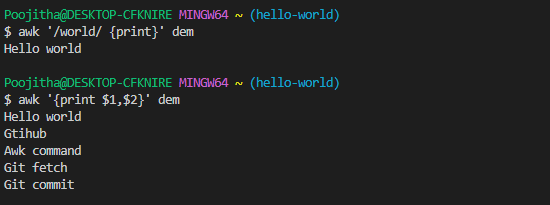
**Q4. Try to find out about the awk command and use it while reading a file created by yourself. Also, make a bash script file and try to find out the prime number from the range 1 to 20.**

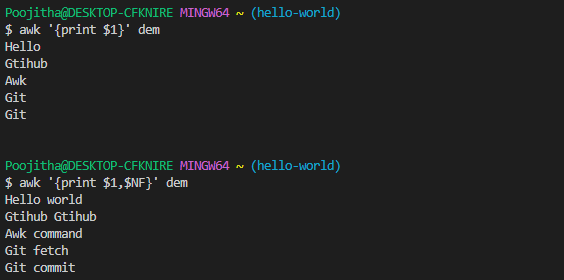
**The whole process should be carried out and by using the history command, give the screenshot of all the processes being carried out.**

**Awk command 🡪** Awk is a powerful tool used in Unix/Linux environments to manipulate and analyze text files. It allows you to easily extract and manipulate data from a file, and also perform more complex operations such as pattern matching and conditional statements.



To print the content in the file we use the following commands.





**Creation of the bash script file.**

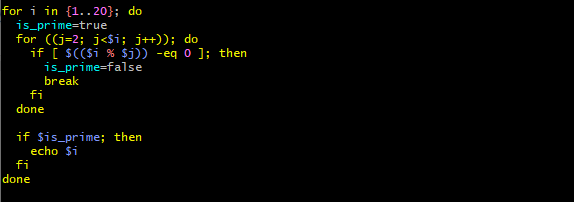
Create a file with the extension .sh

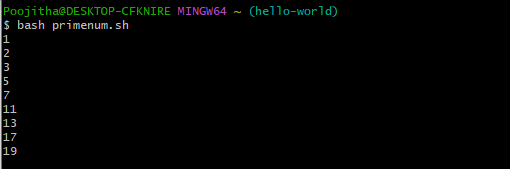
Now type the code and save the file by giving the permissions like read, write and execute.

Now run the file by using the following command

Bash filename.sh

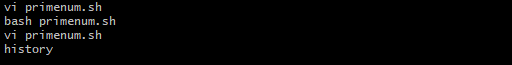






History command🡪 it gives what all the commands are being executed till now.





**Q5. Set up a container and run a Ubuntu operating system. For this purpose, you can make use of the docker hub and run the container in interactive mode.**

**All the processes pertaining to this should be provided in a screenshot for grading.**

Once Docker is installed, open a terminal or command prompt and enter the following command to download the latest Ubuntu image from Docker Hub:

Docker pull ubuntu:latest

This command will download the latest version of the Ubuntu image and save it to your local machine.

Once the download is complete, you can run the container using the following command:

Docker run -it ubuntu:latest

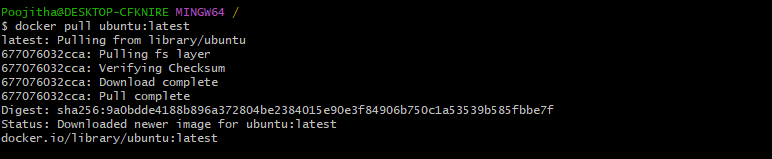
This command will start the container in interactive mode and provide you with a command prompt where you can run Ubuntu commands.

Note that the **-it** flag specifies that the container should run in interactive mode with a TTY attached.

You should now see the command prompt for the Ubuntu container. You can run Ubuntu commands as you would on a regular Ubuntu system.

For example, you can enter **ls** to list the contents of the current directory or **apt-get update** to update the package list.

****

****

