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.

Stash:

In Git , Stash command is used to save the changes made in current working directory and store them in a stash stack and resets the current working directory to its previous commit and the changes are not staged.

These stashed changes can be applied further for future purpose. The Stashed changes can be applied by indexing.

Command:

git stash

Below mentioned are the command line arguments for stash command

apply:

apply command is used to apply the working directory contents with stash changes.

These changes are not commited but staged in the git.

Syntax:

git stash apply stash_index

pop:

pop command is used to pop the latest stashed change from the stash.

These changes will be added to the current directory;

Syntax:

git stash pop

Clear:

Clear is used to empty the stash.The changes will not be reflected to the current working directory.

Syntax:

git stash clear

drop: drop command is used to drop a index from the stash and the index will be deleted from the stash.

By default it removes index 0 from the stash.

Syntax:

git stash drop index

List:

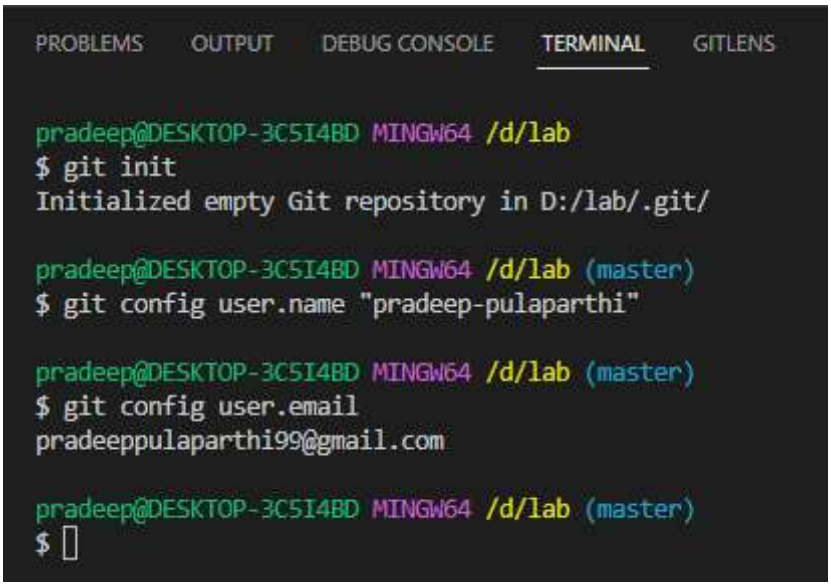
list command is used to list all the contents of the stash.

Syntax:

git stash list

Example:

Open a new folder and initialize the git there.



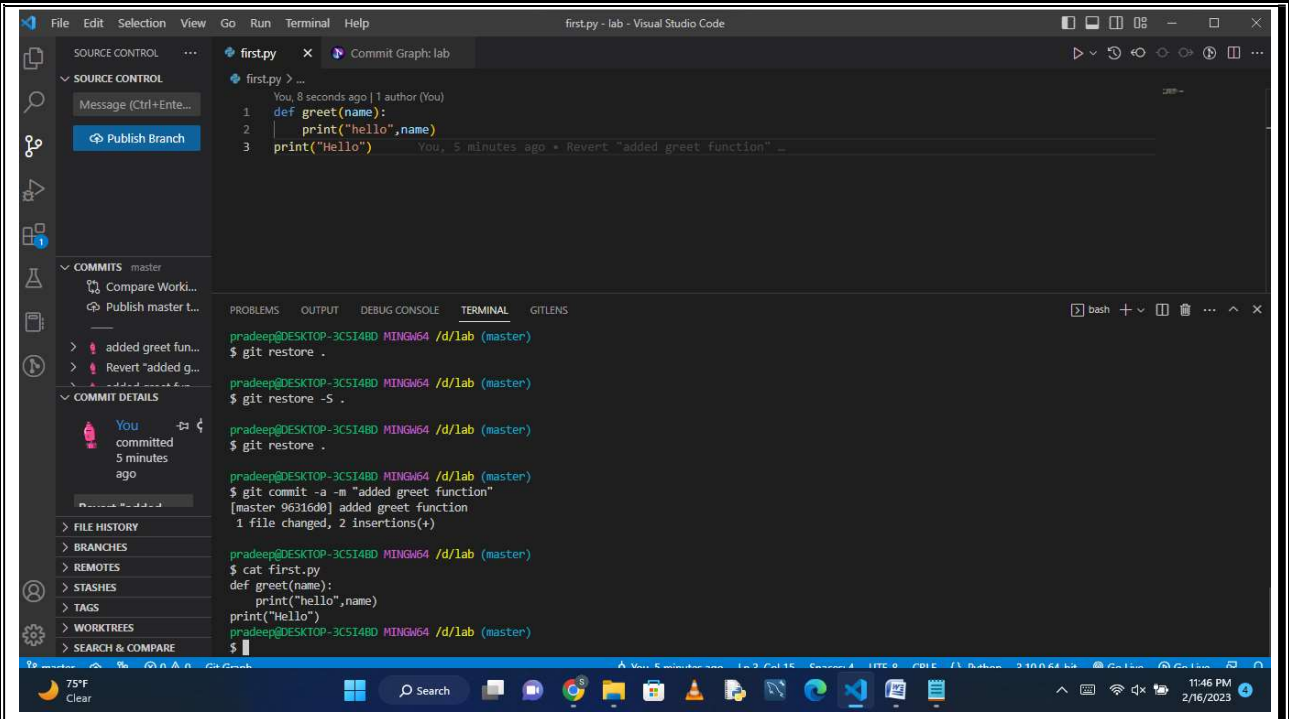
Then create a file named first.py and add some contents.

Now add the gile into staged area.Then commit the changes in the local repository.

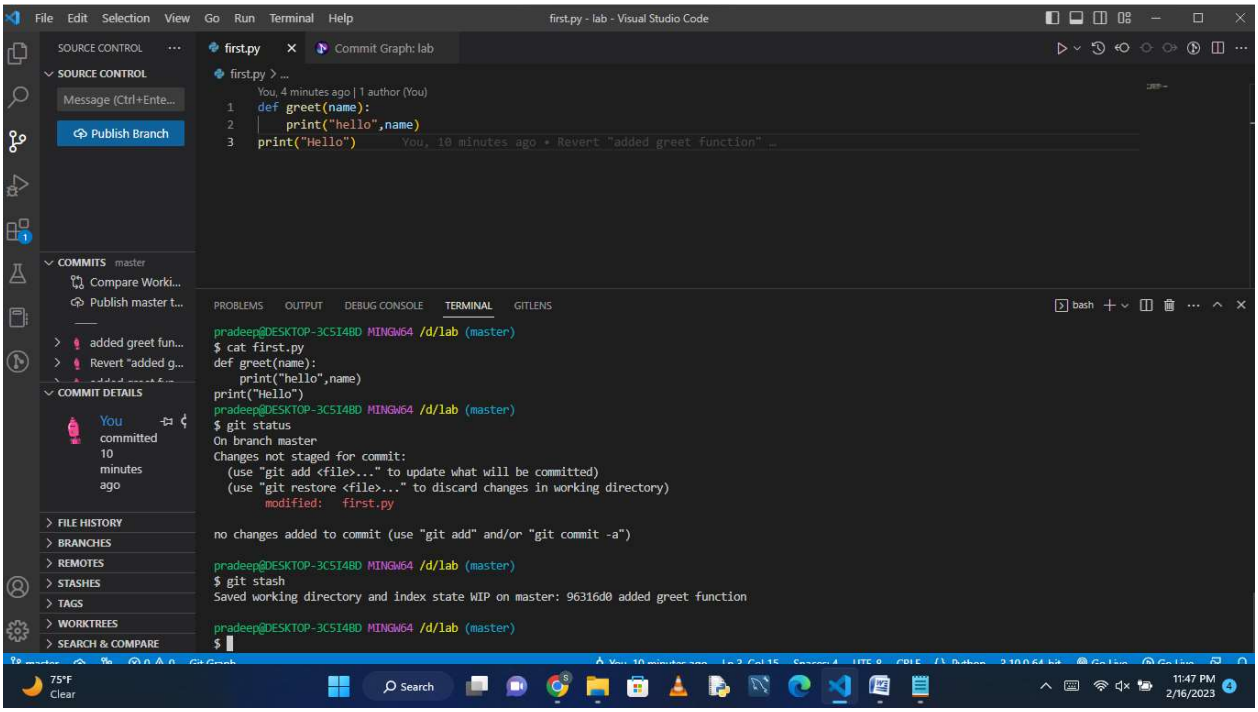
```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ ls
first.py

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ git add *

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   first.py


pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ git commit -m "created first.py file"
[master (root-commit) 24443e7] created first.py file
 1 file changed, 1 insertion(+)
 create mode 100644 first.py
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ cat first.py
print("Hello")
```

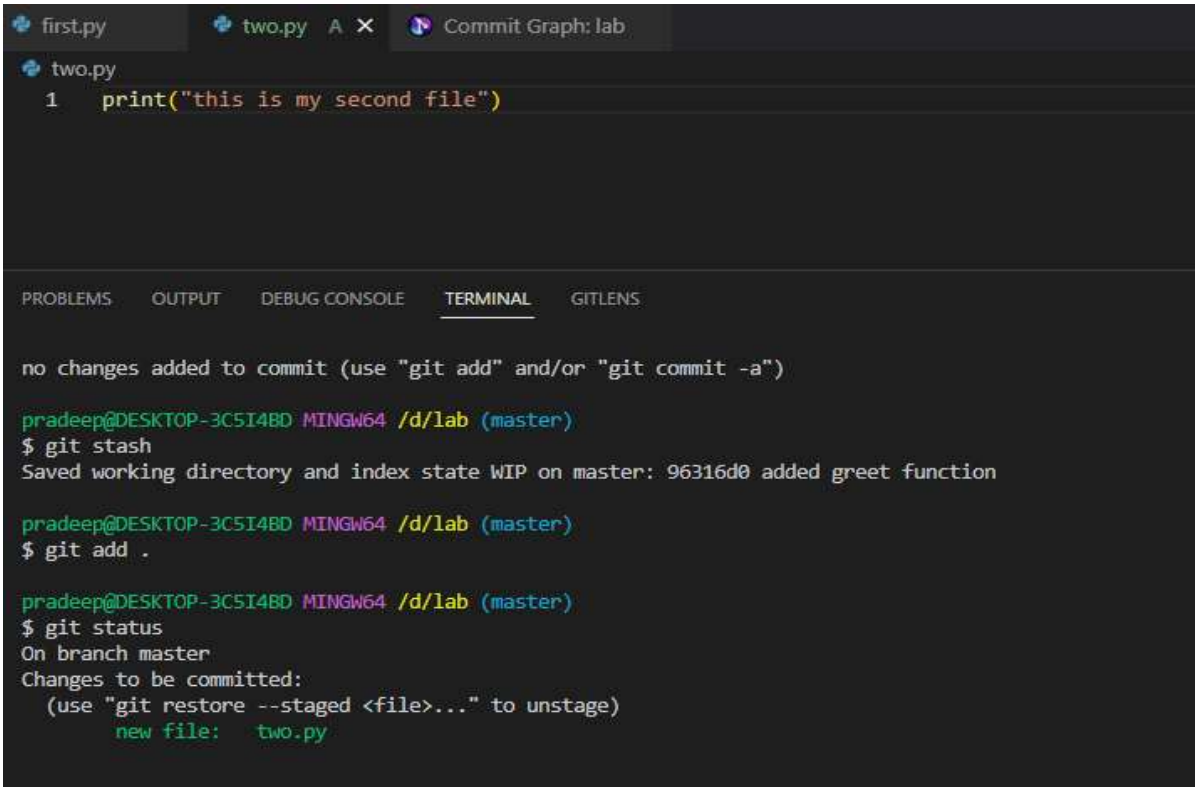
Now do one more commit and contents are:

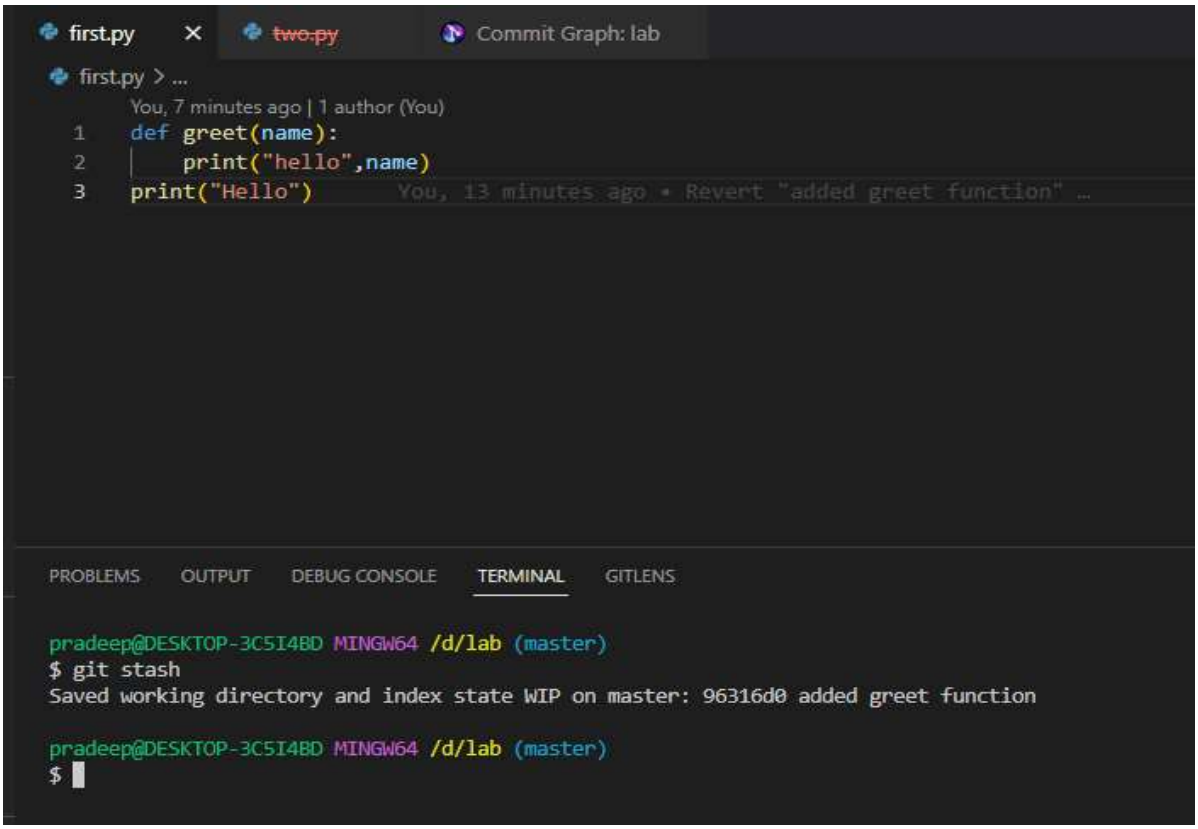


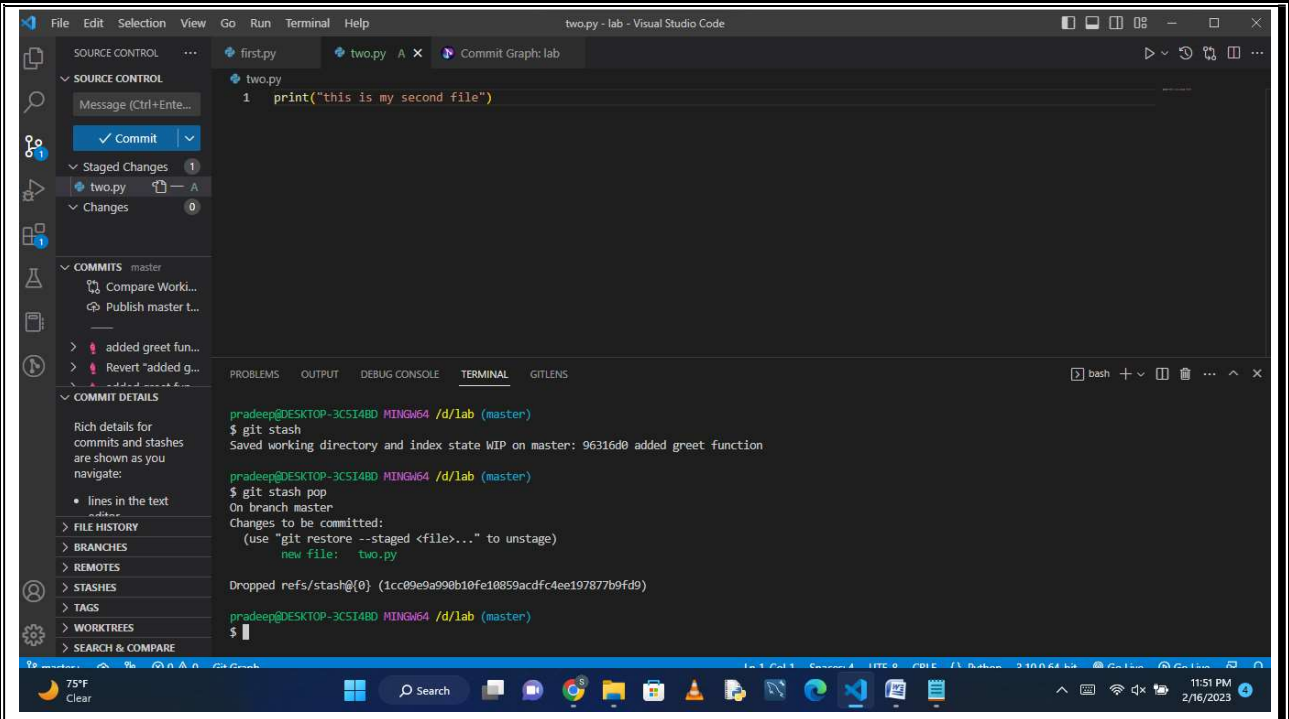
Add a comment and stash the changes.



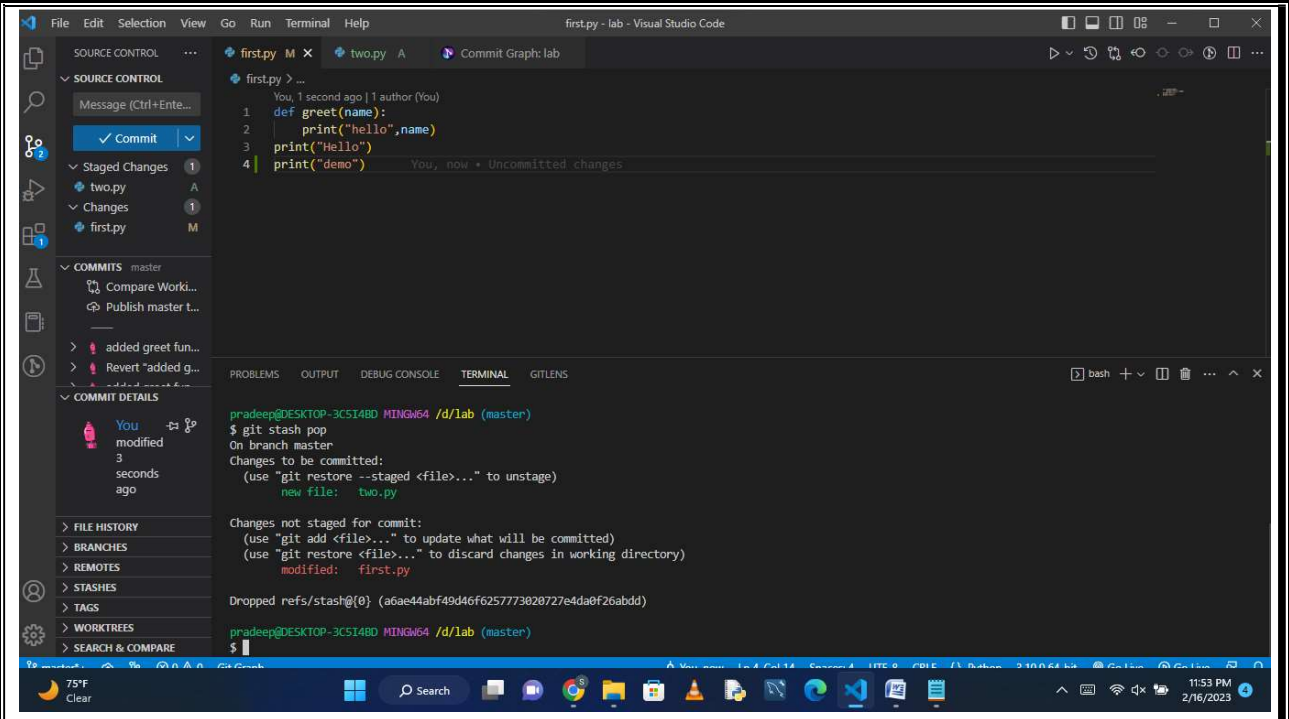
Create one more file and stash the changes.



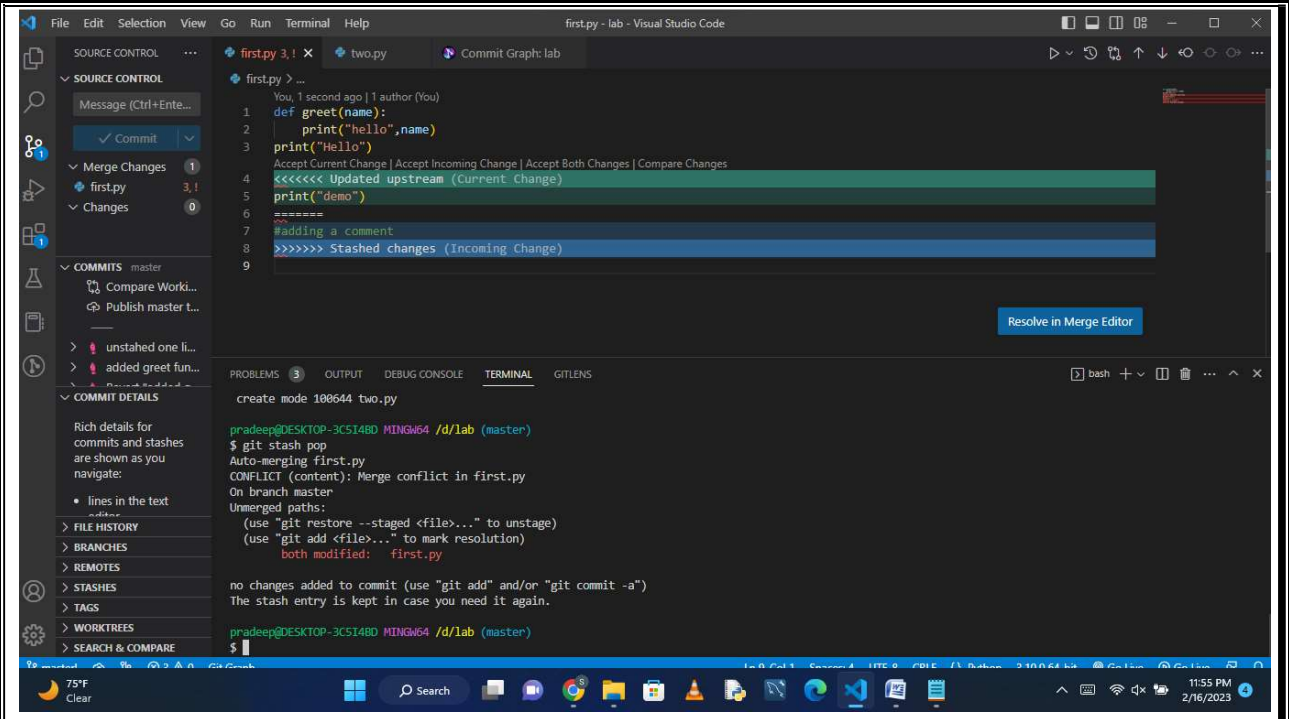




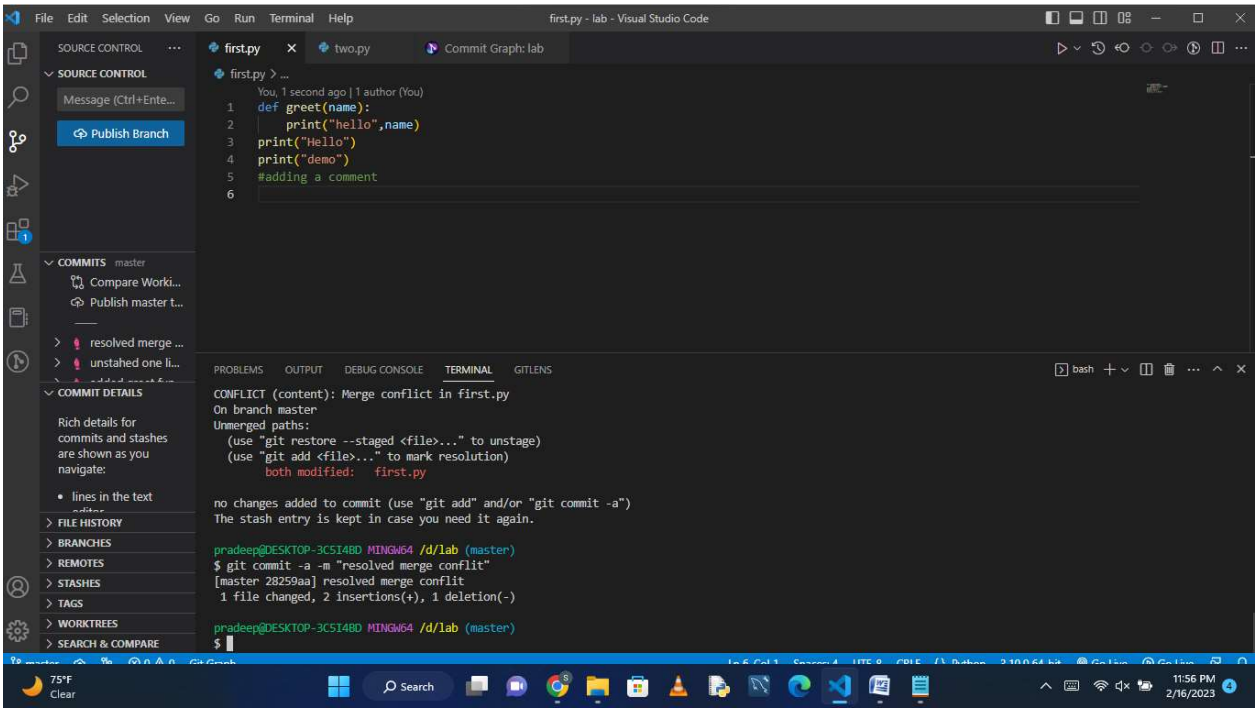
Add some changes to the first.py to get some merge conflits.




```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ git commit -a -m "unstahed one line in first.py"
[master 1419f5d] unstahed one line in first.py
 2 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 two.py
```



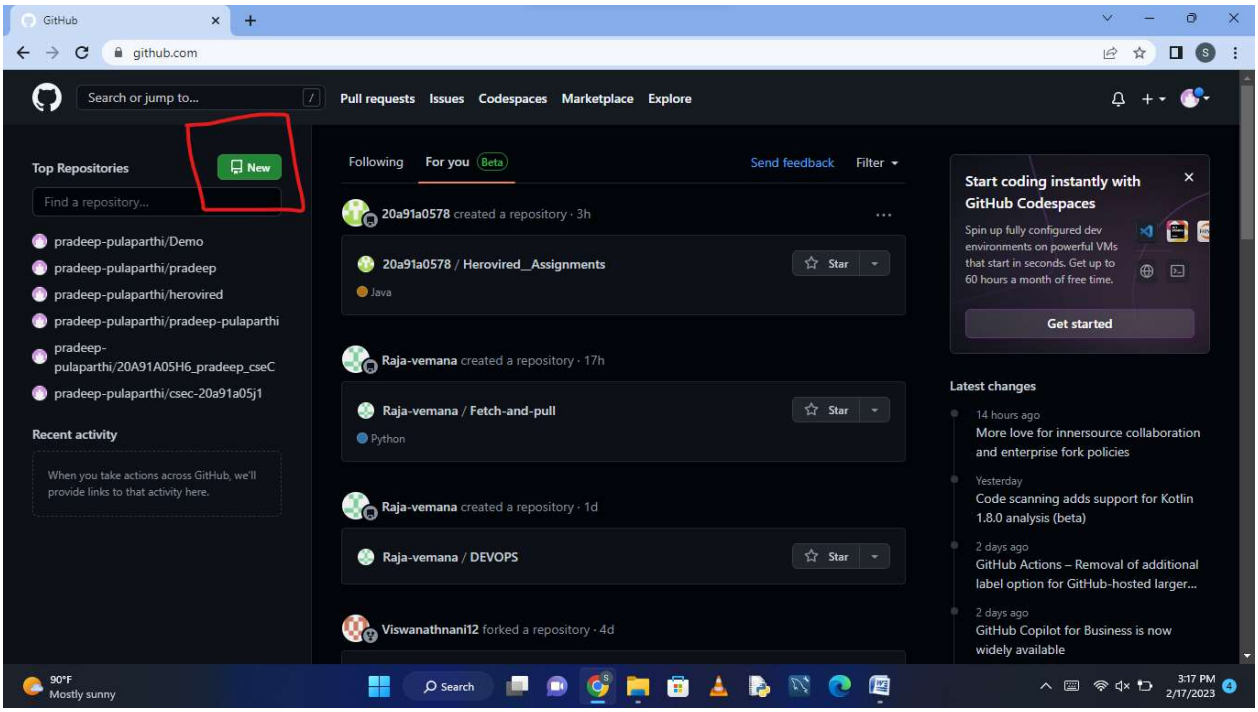
After resolving merge conflit.

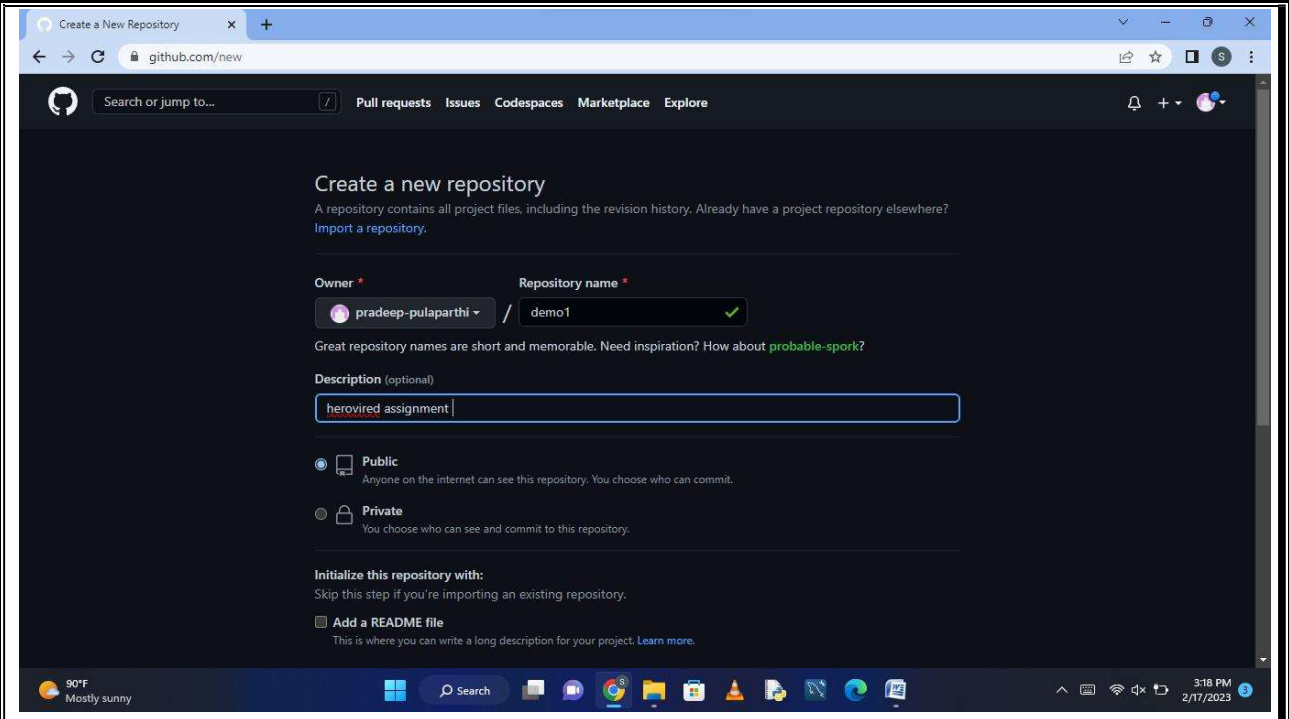


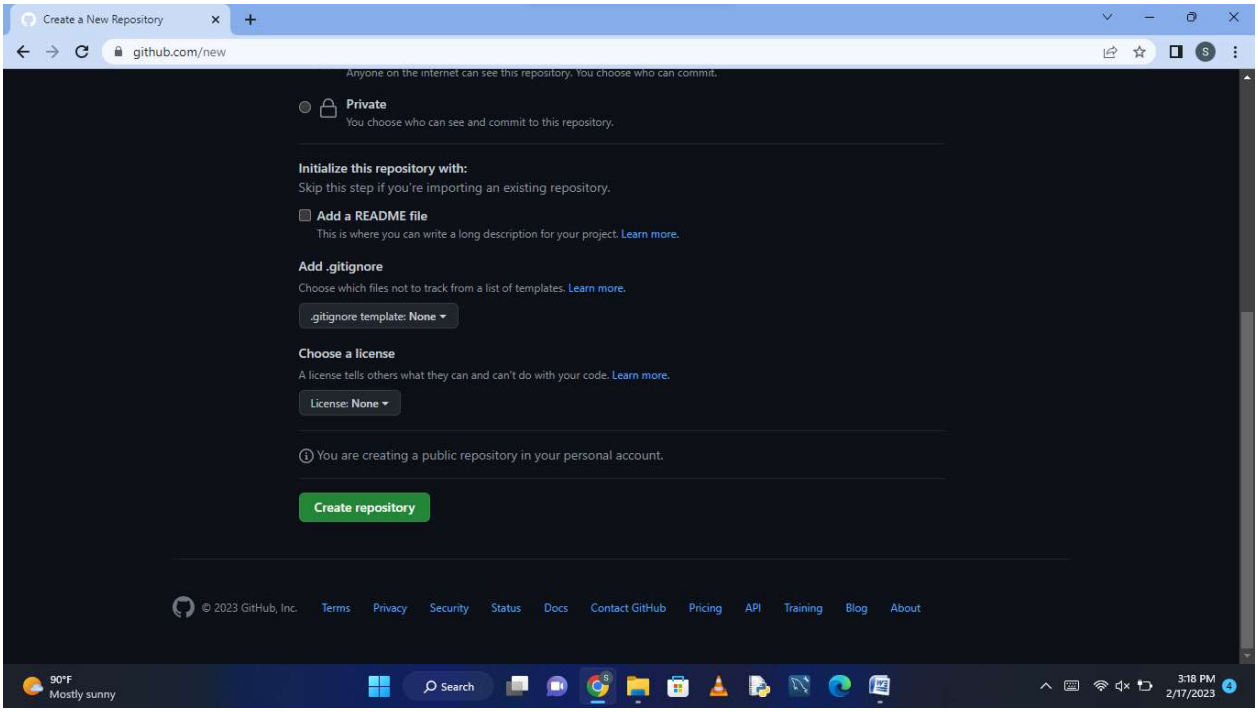
Q2. By using a sample example of your choice, use the git fetch command and also use the git merge command and describe the whole process through a screenshot with all the commands and their output in git bash.

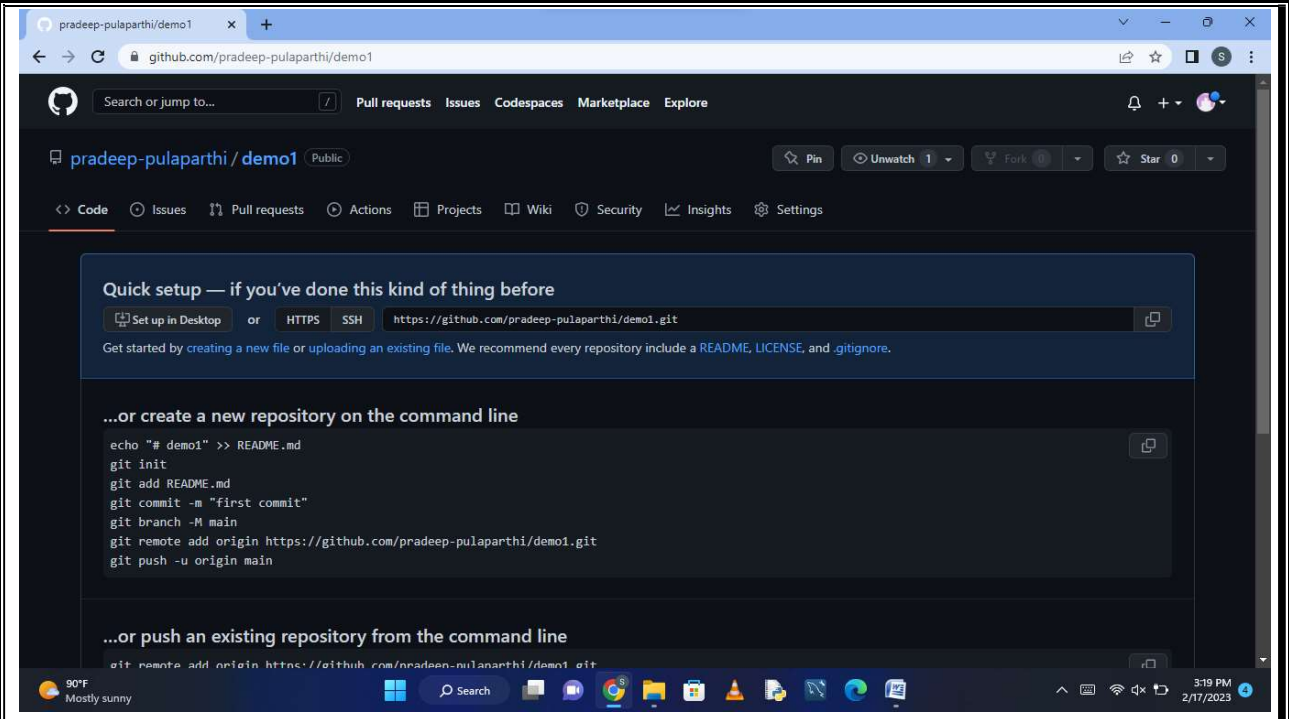
Git fetch: Git fetch is a command that allows you to download objects from remote repository but it doesn't integrate any of this new data into your working files.

*Create a new repository in github.




Create a New Repository
github.com/new
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Owner *
Repository name *
pradeep-pulaparthi
demo1
Great repository names are short and memorable. Need inspiration? How about probable-spork?
Description (optional)
herovired assignment
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Skip this step if you're importing an existing repository.
Add a README file
This is where you can write a long description for your project. Learn more.
90°F
Mostly sunny
Search
3:18 PM
2/17/2023


Create a New Repository
github.com/new
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Skip this step if you're importing an existing repository.
Add a README file
This is where you can write a long description for your project. Learn more.
Add .gitignore
Choose which files not to track from a list of templates. Learn more.
.gitignore template: None
Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: None
You are creating a public repository in your personal account.
Create repository
© 2023 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About
90°F
Mostly sunny
Search
3:18 PM
2/17/2023

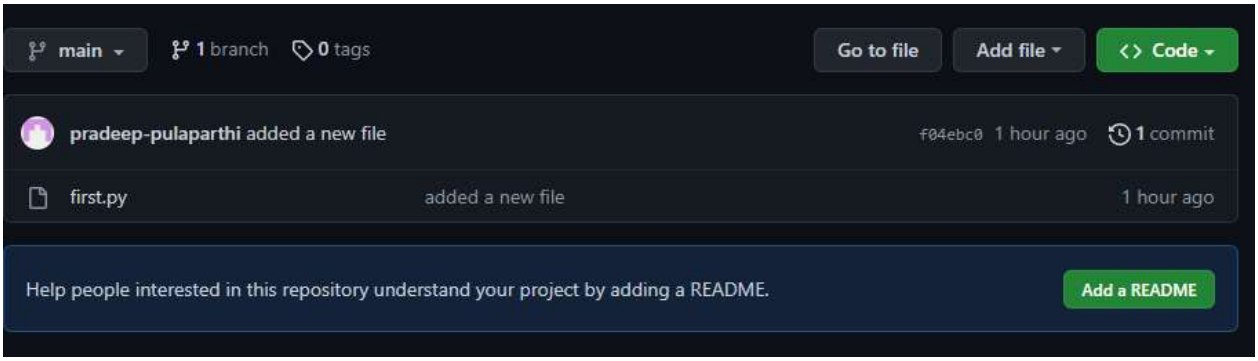


*Clone the empty remote repository into local repository using git bash.

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ git config user.name pradeep-pulaparthi

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ git config user.email pradeeppulaparthi99@gmail.com

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ git clone https://github.com/pradeep-pulaparthi/demo1.git
Cloning into 'demo1'...
warning: You appear to have cloned an empty repository.

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab (master)
$ cd demo1

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (main)
$ git log --oneline
fatal: your current branch 'main' does not have any commits yet

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (main)
$ git log --oneline --all

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (main)
$
```

Create a file in git hub with some content.




```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (main)
$ git log --oneline --all
```

But we cannot find any commits in the local repository.

Git fetch:

Let us fetch the repository. It will download all the changes that are made in the remote repository but doesn't merge our changes with working files.

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (main)
$ git fetch
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 594 bytes | 37.00 KiB/s, done.
From https://github.com/pradeep-pulaparthi/demo1
 * [new branch]      main       -> origin/main
```

Use the git log --oneline --all command to check whether the changes have been downloaded or not.

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (main)
$ git log --oneline --all
f04ebc0 (origin/main) added a new file
```

Here using git fetch the commits are not applied to the main branch.

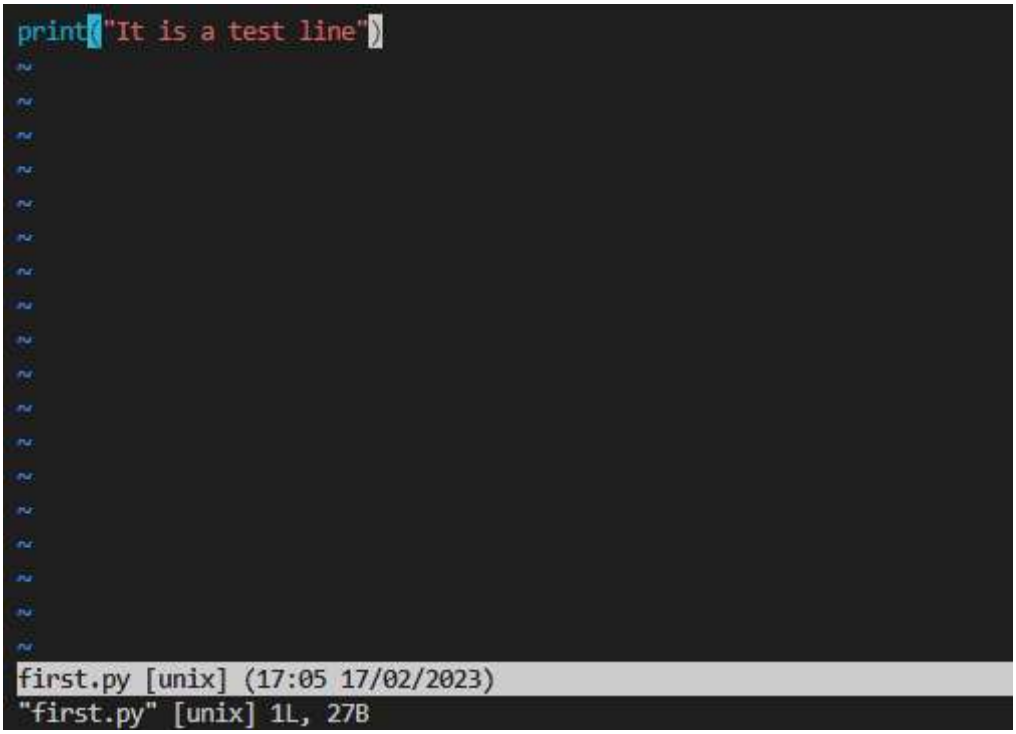
Git merge:

Git merging is basically to merge multiple sequences of commits, stored in multiple branches in a unified history or to be simple you can say in a single branch.

Create a branch with git branch command

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (main)
$ git checkout -b mybranch
Switched to a new branch 'mybranch'
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (branch1)
$ vi first.py
```



```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (branch1)
$ git status
On branch branch1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    ../first.py
        deleted:    ../two.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ./
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (branch1)
$ git commit -a -m "added a file"
[branch1 353d9c8] added a file
 4 files changed, 2 insertions(+), 6 deletions(-)
 create mode 160000 demo1/demo1
 create mode 100644 demo1/first.py
 delete mode 100644 first.py
 delete mode 100644 two.py
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (branch1)
$ git log --oneline
353d9c8 (HEAD -> branch1) added a file
```

Now create one more branch

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (branch1)
$ git branch mybranch2

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (branch1)
$ git switch mybranch2
Switched to branch 'mybranch2'

pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (mybranch2)
$ vi first.py
```

```
print("Hello")
~
~
~
~
~
~
~
~
~
~
~
~
~
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (mybranch2)
$ git status
On branch mybranch2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   first.py

no changes added to commit (use "git add" and/or "git commit -a")
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (mybranch2)
$ git commit -a -m "modified first.py file"
[mybranch2 0c7fbdd] modified first.py file
 1 file changed, 1 insertion(+), 1 deletion(-)
```

There are two different branches whenever we need to combine the work of both the branches we use git merge command.

Syntax: git merge branch_name

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (branch1)
$ git merge mybranch2
Updating 353d9c8..0c7fbdd
Fast-forward
 demo1/first.py | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 /d/lab/demo1 (branch1)
$ git log --oneline
0c7fbdd (HEAD -> branch1, mybranch2) modified first.py file
353d9c8 added a file
```

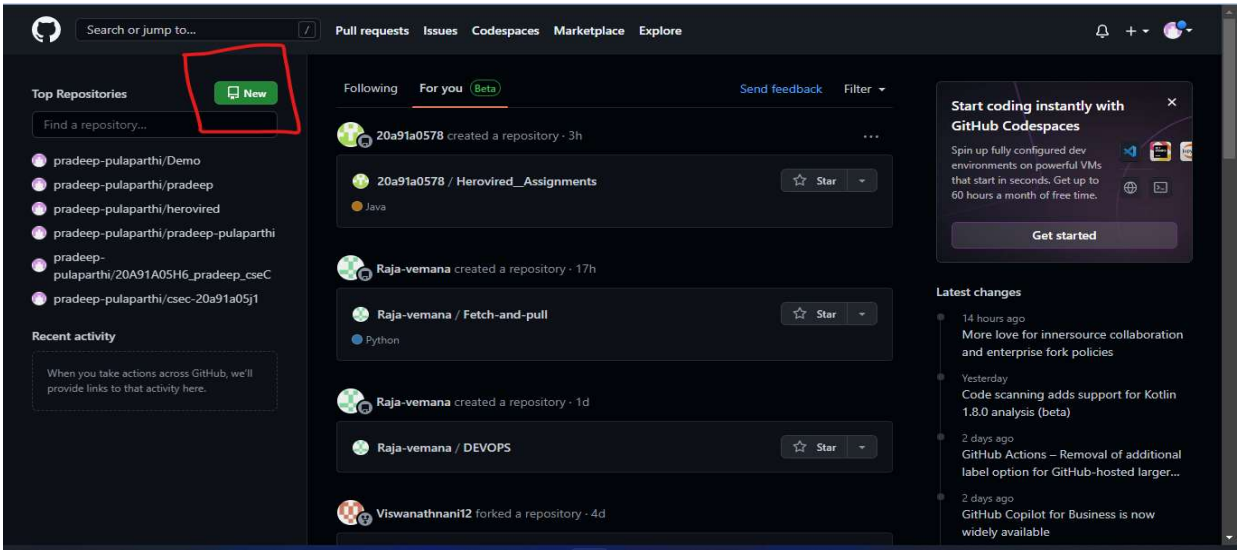
Q3) State the difference between git fetch and git pull by doing a practical example in your git bash and attach a screenshot of all the processes.

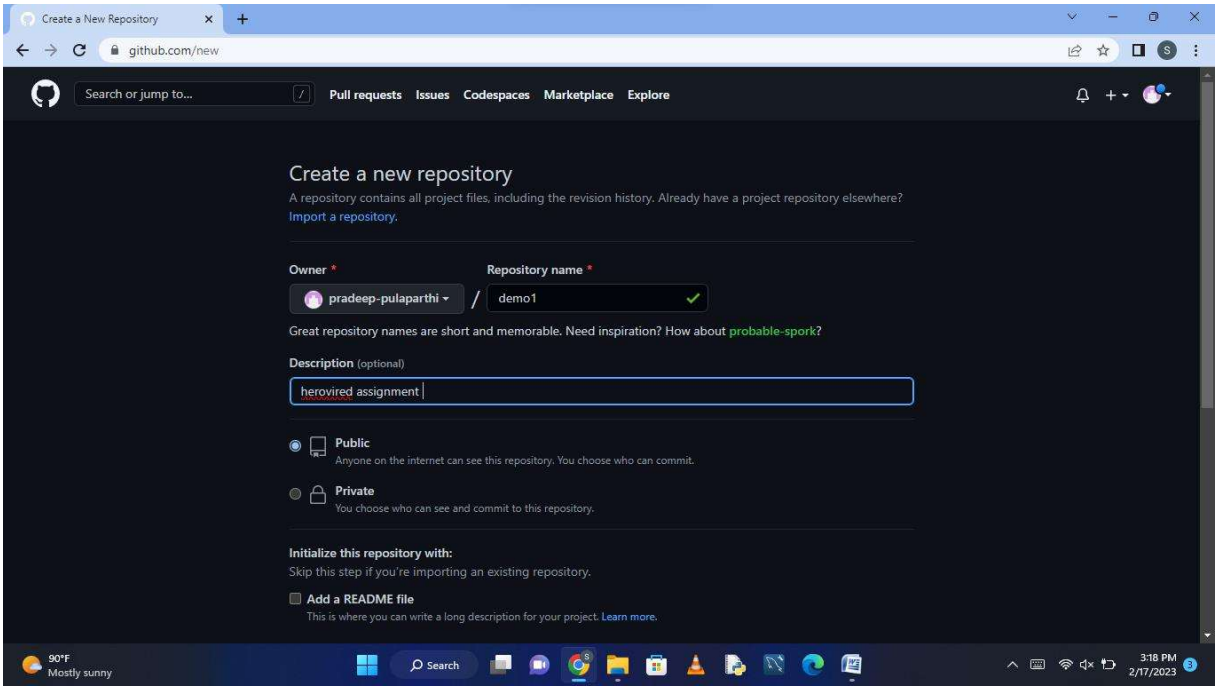
Git fetch: Git fetch is a command that allows you to download objects from remote repository but it doesn't integrate any of this new data into your working files.

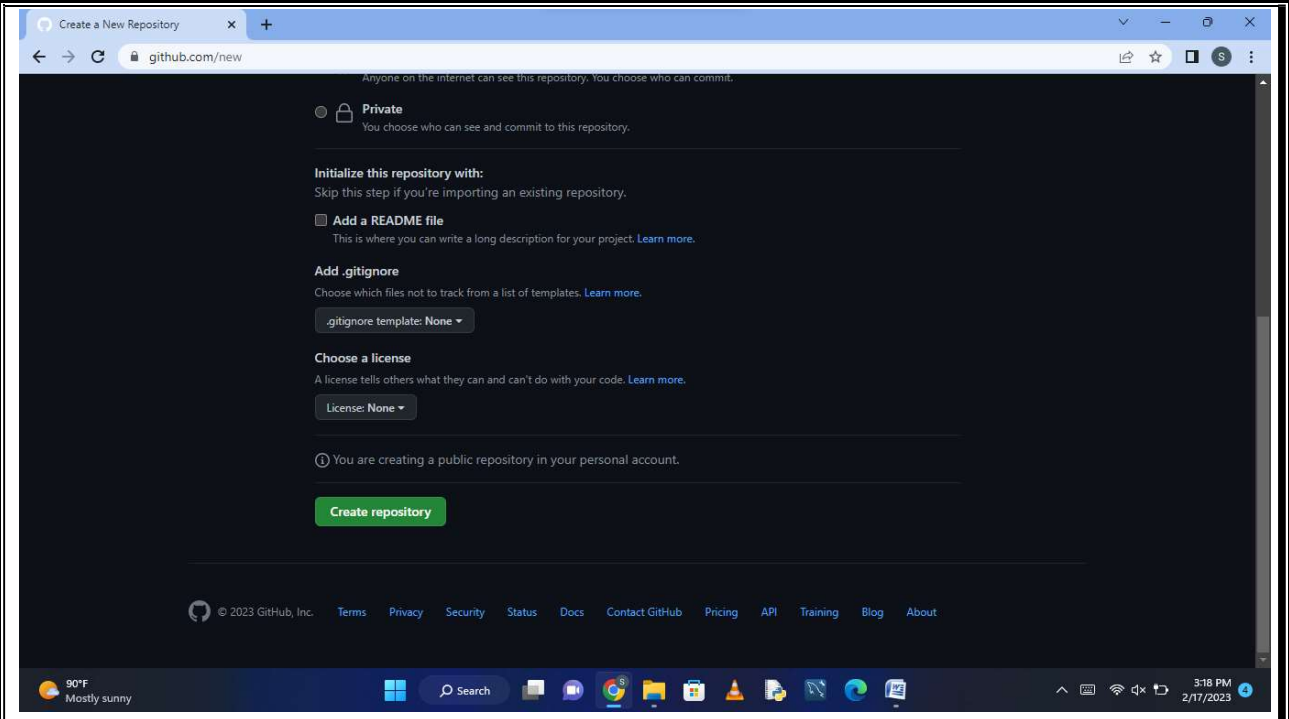
Git pull: Git pull is a command that allows you to fetch from and integrate with another repository or local branch. It update your current HEAD branch with the latest changes from the remote server.

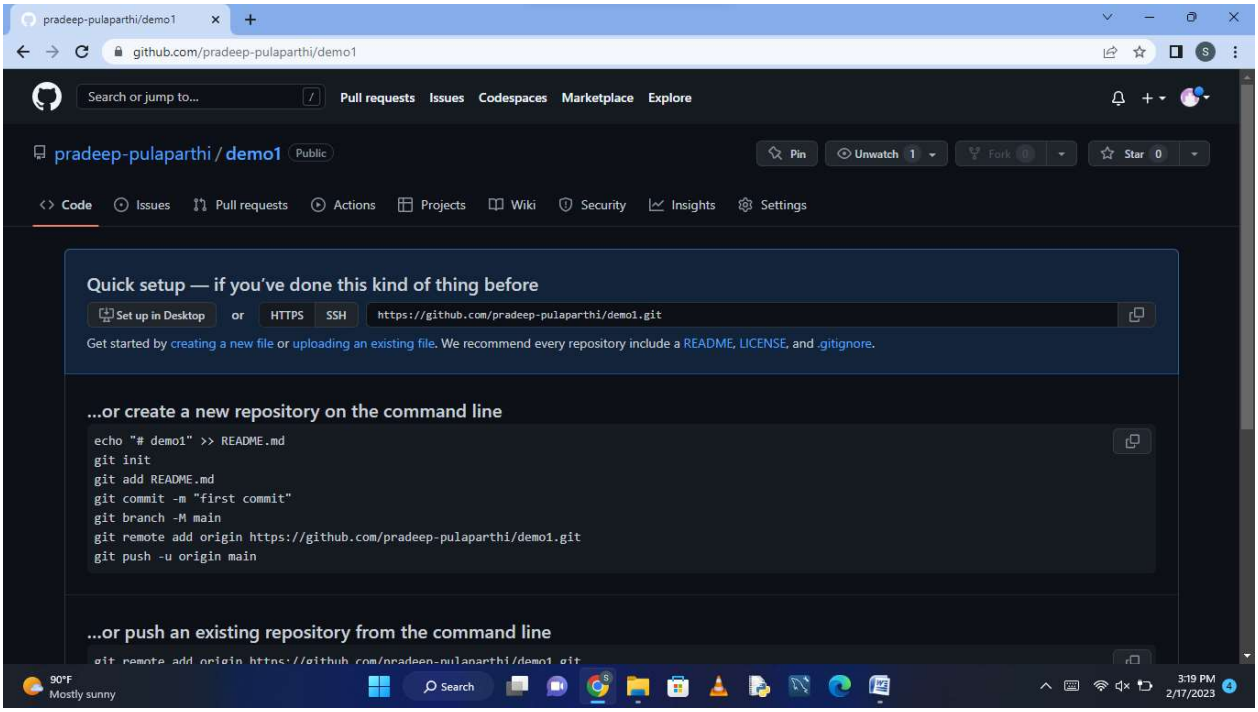
We can say that git pull is a git fetch followed by an additional action say git merge.

Create a new repository in github.





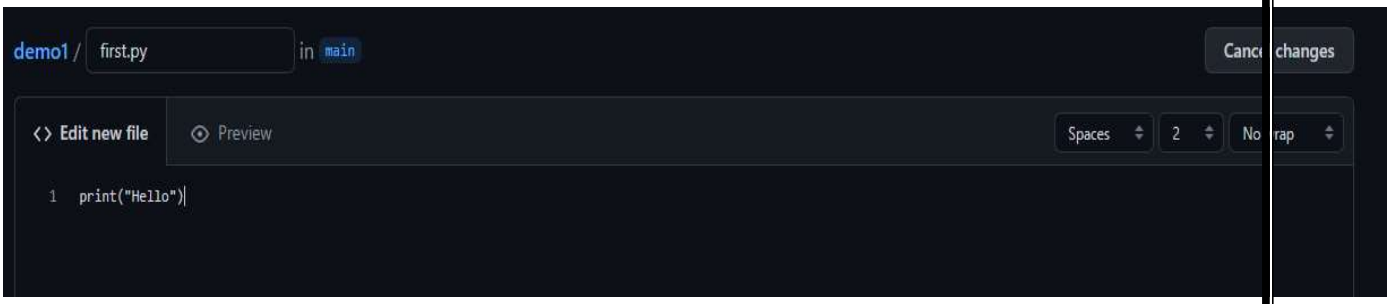




We cannot find any commits in our local repository even if we made a commit in remote repository.

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo (master)
$ git init
Initialized empty Git repository in C:/Users/pradeep/Desktop/demo/.git/

pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo (master)
$ git clone https://github.com/pradeep-pulaparthi/demo1.git
Cloning into 'demo1'...
warning: You appear to have cloned an empty repository.
```

Create a file in the github.



Commit the changes

We cannot find any commits in our local repository even if we made a commit in remote repository

Git fetch:

Let us fetch the repository. It will download all the changes that are made in the remote repository but doesn't merge our changes with working files.

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ git fetch
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 604 bytes | 14.00 KiB/s, done.
From https://github.com/pradeep-pulaparthi/demo1
 * [new branch]      main       -> origin/main
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ git log --oneline --all
7c713ed (origin/main) Create first.py
```

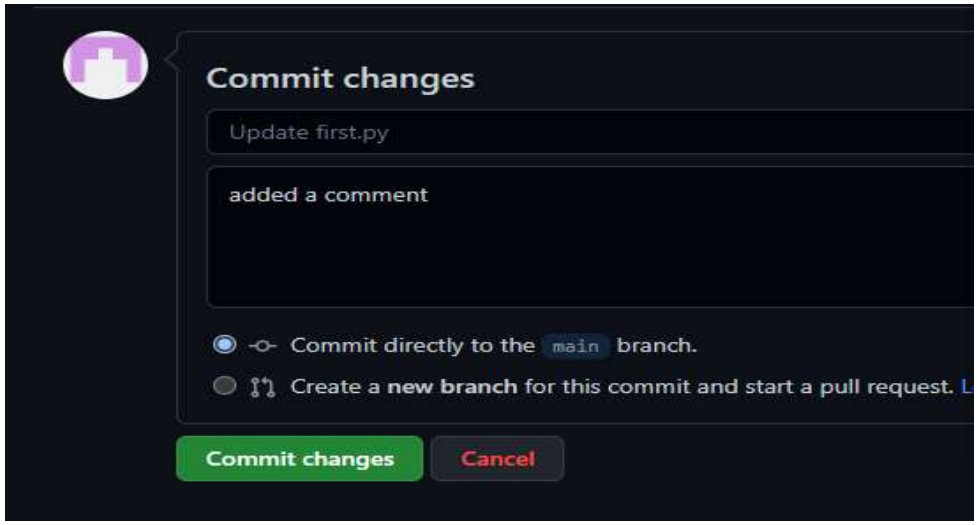
Here using git fetch the commits are not applied to the main branch

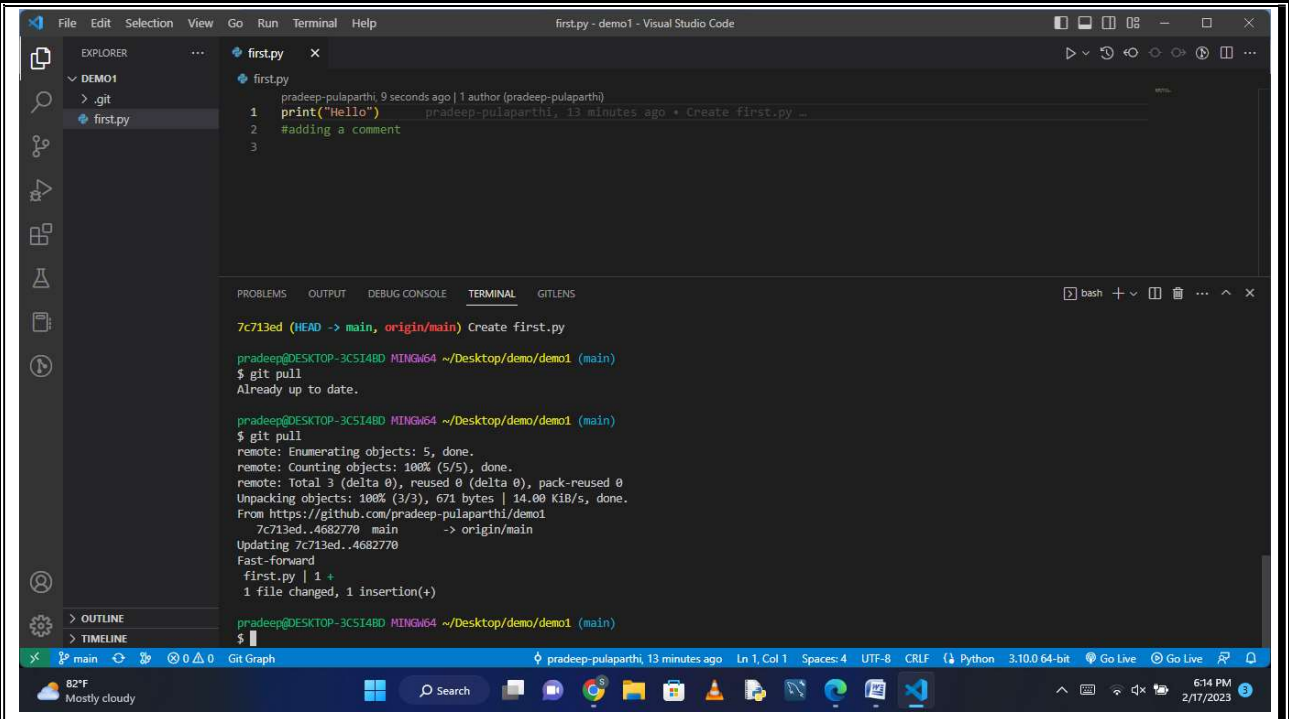
Git pull: It downloads and also applies the changes to the working files.

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ git pull
Already up to date.
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ git log --all --oneline
7c713ed (HEAD -> main, origin/main) Create first.py
```

Now do some more changes in the file in github;






```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 671 bytes | 14.00 KiB/s, done.
From https://github.com/pradeep-pulaparthi/demo1
   7c713ed..4682770  main       -> origin/main
Updating 7c713ed..4682770
Fast-forward
 first.py | 1 +
 1 file changed, 1 insertion(+)
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ git log --oneline
4682770 (HEAD -> main, origin/main) Update first.py
7c713ed Create first.py
```

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:

The Awk is a powerful scripting language used for text scripting. It searches and replaces the texts and sorts, validates, and indexes the database. It performs various actions on a file like searching a specified text and more.

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ awk '{print "this a demo"}'

this a demo
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ vi fruits.txt
```

```
FRUIT  COST
Apple  100
Banana 200
Guava  90
Papaya 100
Mango  200
Kiwi   150
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ awk '{ print }' fruits.txt
FRUIT  COST
Apple  100
Banana 200
Guava  90
Papaya 100
Mango  200
Kiwi   150
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ awk '/Apple/{ print }' fruits.txt
Apple  100
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ awk '/100/{ print }' fruits.txt
Apple  100
Papaya 100
```

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ awk '{ print $2}' fruits.txt
COST
100
200
90
100
200
150
```

The above command will print column1

Steps to write a program to find the primes in a range:

1)open vi editor using vi file.sh

2)press Insert key (or i) and write the program

3)save and quit the file using escape+":wq" command

4)run the file using sh filename.sh

Program:

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ vi prime.sh
```

```
isprime()
{
    n=$1
    if [ $1 == 1 ] ; then
    return 0
    fi
    for (( i=2; i*i<=$1 ; i++ ));
    do
        if [ $(($n%i)) == 0 ] ;
        then
            return 0
        fi
    done
    echo $1
}
for i in {1..20}
do
    isprime $i
done
prime.sh [unix] (18:44 17/02/2023)
"prime.sh" [noeol][unix] 19L, 247B
```

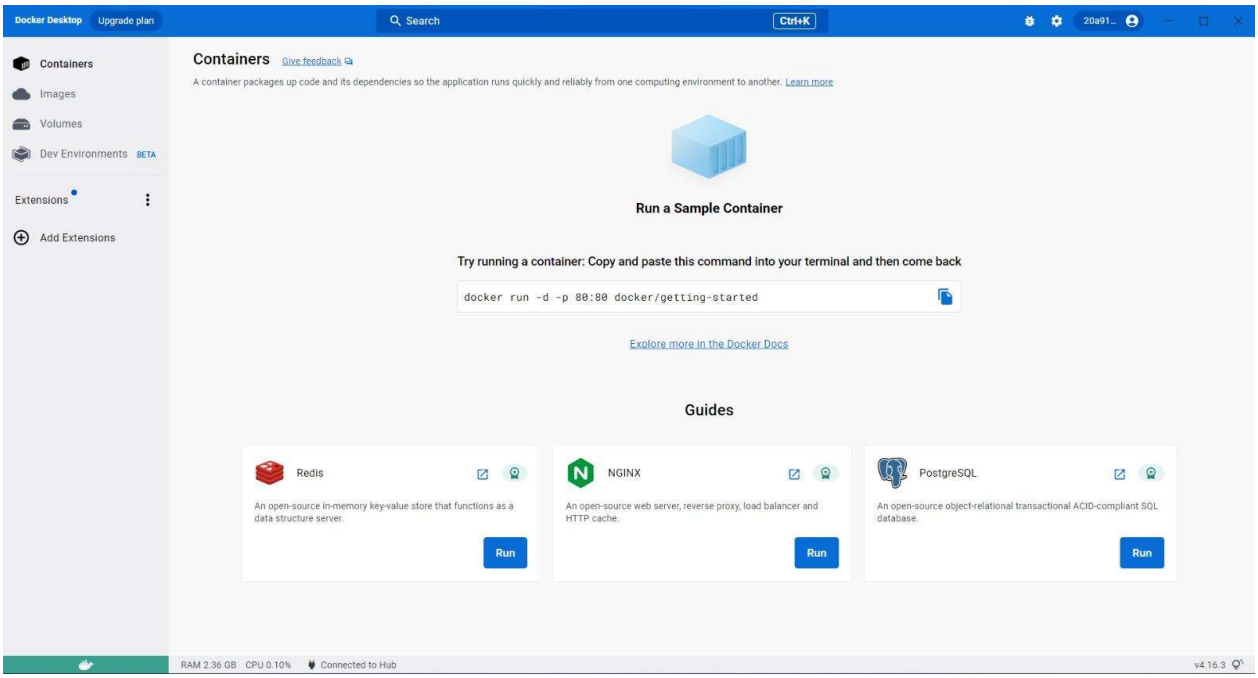
Run the file

```
pradeep@DESKTOP-3C5I4BD MINGW64 ~/Desktop/demo/demo1 (main)
$ sh prime.sh
2
3
5
7
11
13
17
19
```

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.

Image: Images are used to create containers. It uses a private container registry to share container images within the enterprise and also use public container registry to share container images with whole world.

Container: Containers are used to hold the entire package that is needed to run the application. We can say that the image is a template and the container is a copy of the template. *These are the containers present in the docker desktop.



For setting up a container and run the ubuntu os,

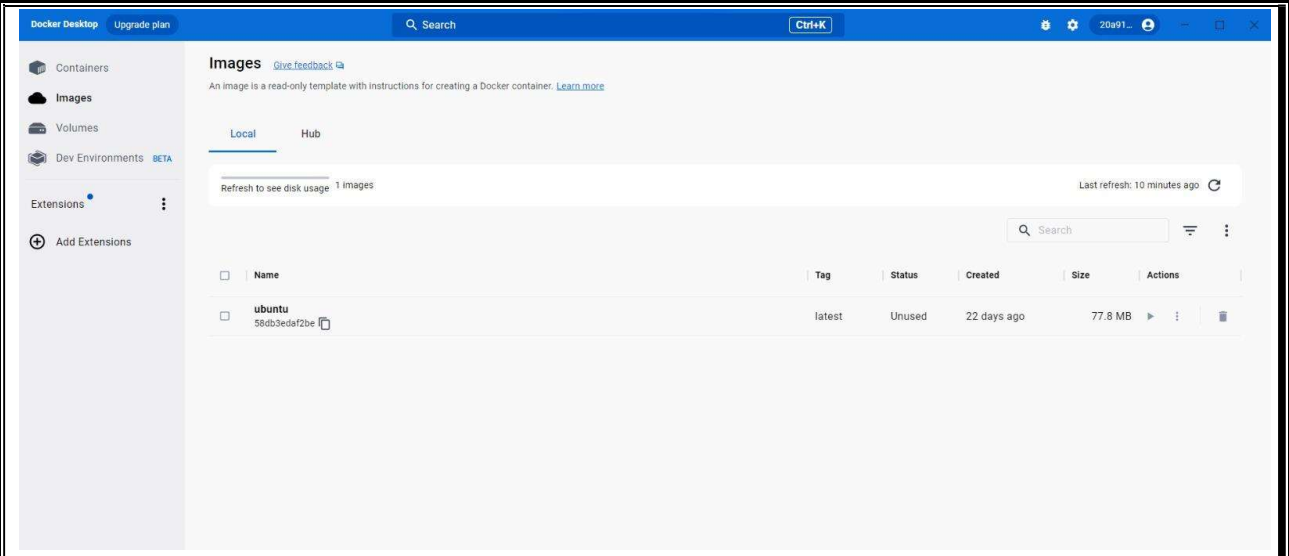
First we need to download the image of Ubuntu from docker hub using the command docker pull ubuntu .

-> To create a container and execute the image use the command docker run -it ubuntu .

-> To get an idea about the available update use apt update command.

Download the ubuntu OS image from the docker hub.

```
C:\WINDOWS\system32>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
677076032cca: Pull complete
Digest: sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbbe7f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```



Now run the ubuntu image which has been downloaded.

```
C:\Users\hp>docker run -it ubuntu
root@c7ada82d2d12:/# apt update
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [807 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [860 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [5557 B]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [752 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [808 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [10.9 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1091 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1136 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [22.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [49.0 kB]
Fetched 25.8 MB in 54s (481 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@c7ada82d2d12:/# apt list --upgradable
Listing... Done
libpam-modules-bin/jammy-updates,jammy-security 1.4.0-11ubuntu2.3 amd64 [upgradable from: 1.4.0-11ubuntu2.1]
libpam-modules/jammy-updates,jammy-security 1.4.0-11ubuntu2.3 amd64 [upgradable from: 1.4.0-11ubuntu2.1]
libpam-runtime/jammy-updates,jammy-security 1.4.0-11ubuntu2.3 all [upgradable from: 1.4.0-11ubuntu2.1]
libpam0g/jammy-updates,jammy-security 1.4.0-11ubuntu2.3 amd64 [upgradable from: 1.4.0-11ubuntu2.1]
libssl3/jammy-updates,jammy-security 3.0.2-0ubuntu1.8 amd64 [upgradable from: 3.0.2-0ubuntu1.7]
root@c7ada82d2d12:/#
```

Now the container of Ubuntu is created!!.