

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: The git stash command takes your uncommitted changes (both staged and unstaged), saves them away for later use, and then reverts them from your working copy.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git branch
* master

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git config --global user.name "20A91A05D2"

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git config --global user.email "20a91a05d2@aec.edu.in"

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git branch devops

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git branch
devops
* master
```

Create a file. For example here we created assign1.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ vi assign1
```

Assignment1 on stash

```

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
assign1 [unix] (09:57 16/02/2023) 2
"assign1" [unix] 2L. 22B

```

*Use **git status** command to observe in which stage the file is in.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        assign1
```

*The file is in untracked state.

*Now we are adding the untracked file to the staging area using **git add command** and then commit the changes using **git commit command**.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git add .
warning: in the working copy of 'assign1', LF will be replaced by CRLF the
time Git touches it

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git commit -m "line1 added"
[master 564d580] line1 added
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 assign1
```

*Let us consider the branch we are working be “br1” if there is a need to switch to another branch say “br2” then if there are any committed changes in br1 then these changes will be transferred/reflected to br2 which will disturb the workflow of br2.

Inorder to avoid this we either need to remove the uncommitted changes. But if the changes are needed by us we need to place them in a separate place .For placing these changes we make use of “Stash”.

***git stash (or) git stash save**: Used to stash the uncommitted changes(staged or unstaged) into a stash stack. It undoes to the latest commit and place the additional changes(uncommitted) into a stack instead of deleting.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ vi assign1
```

[illegible]

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git status
On branch master
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:   assign1

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

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git stash
warning: in the working copy of 'assign1', LF will be replaced by CRLF the
time Git touches it
Saved working directory and index state WIP on master: 564d580 line1 add
```

***git stash list**: Used to list out all the stashes stored in the stash stack.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git stash list
stash@{0}: WIP on master: 564d580 line1 added
```

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ vi assign1
```

[illegible]**assign1 [dos] (10:20 16/02/2023)**

```
"assign1" [dos] 2L, 24B
```

*As the stack follows the LIFO(Last In First Out) the latest stashes will be at the top of the stack i.e, at 0th index.

*In the above example we get the latest commit id along with the commit message.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ vi assign1
```

as

Br
\$
On
ch

no

Bl
\$
S

BL
\$
st
st

*g
(sta

Th

\$1

*git stash show can also be used done along with the index number to show the changes at that particular index.

Syntax: `git stash show -index`

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git stash show 1
assign1 | 1 +
1 file changed, 1 insertion(+)
```

***git stash apply:** Used to apply the latest stash without removing the stash from the stash stack.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git stash apply
On branch master
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:   assign1

no changes added to commit (use "git add" and/or "git commit -a")
Assignment1 on stash
Some more lines
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
assign1 [dos] (10:38 16/02/2023)
"assign1" [dos] 2L, 39B
```

*The changes from the latest stash i.e, the line “Some more lines” got added to the file.

*The stash was also not removed from the stack.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git stash list
stash@{0}: WIP on master: 564d580 line1 added
stash@{1}: WIP on master: 564d580 line1 added
```

*We can also use it along with index number by giving `→git apply index`

***git stash pop**: Used to apply the recorded changes of your latest stash on the current working branch as well as remove that stash from the stash stack.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git stash pop
On branch master
nothing to commit, working tree clean
Dropped refs/stash@{0} (36d3d75abc852923166ab65bbaa400ee4351ea87)
```

```
Assignment1 on stash
Some more lines
```

~~~~~

```
assign1 [dos] (10:43 16/02/2023)
"assign1" [dos] 2L 39B
```

\***git stash clear** : Used to clear all the stashes from stash stack.

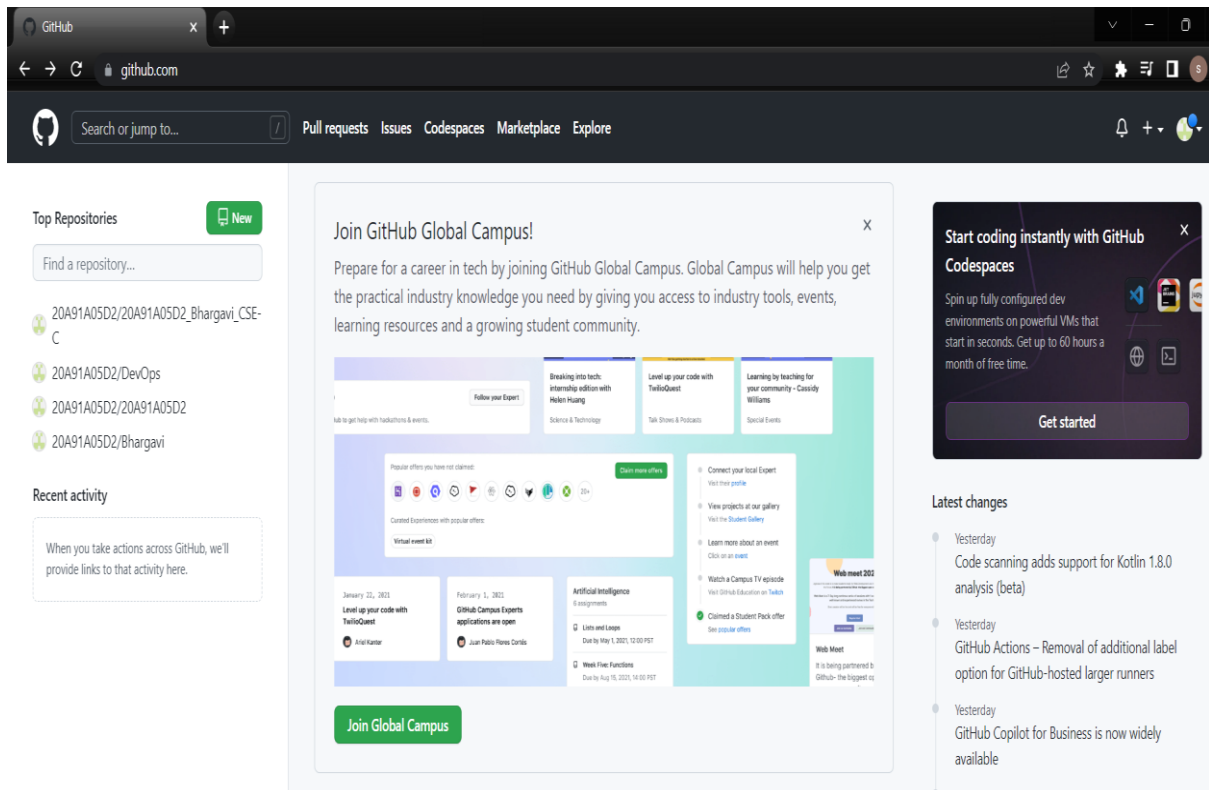
```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git stash clear

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/mygit (master)
$ git stash list
```

**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

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner \*

20A91A05D2 ▾

Repository name \*

/ fetch and pull ✓

Great repository names are short, lowercase, and hyphenated. Your new repository will be created as **fetch-and-pull**. Not **expert-chainsaw**?

Description (optional)

☒ 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

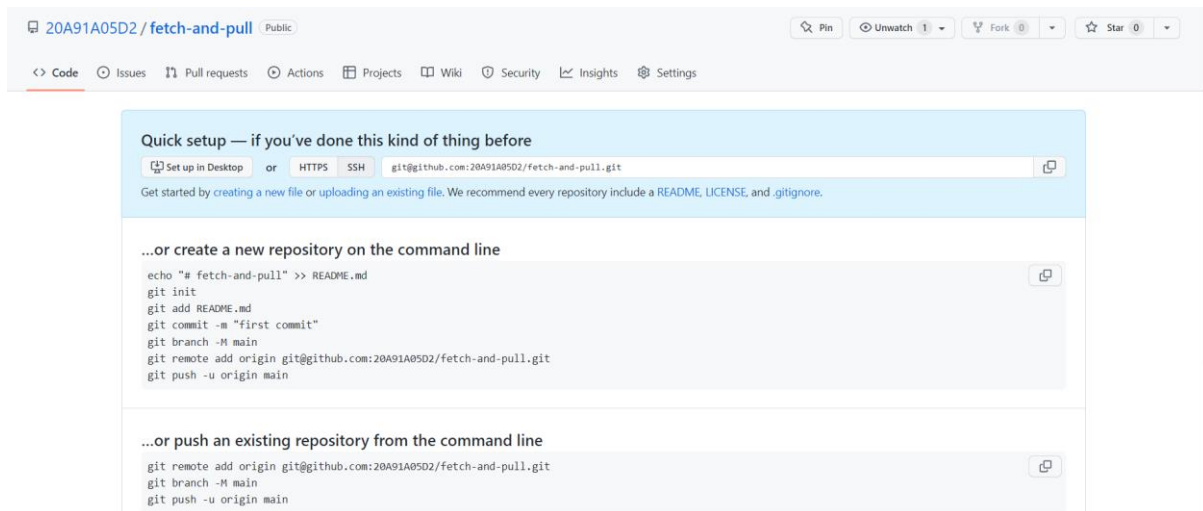
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾





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

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~ (master)
$ cd documents

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ git init
Initialized empty Git repository in C:/Users/hp/Documents/.git/

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ git config --global user.name "20A91A05D2"

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ git config --global user.email "20a91a05d2@aec.edu.in"

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ git clone https://github.com/20A91A05D2/fetch-and-pull.git
Cloning into 'fetch-and-pull'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

\*Check if there are any commits present in the repository.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ cd fetch-and-pull

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
$ git log --oneline --all

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
```

\*Create a file in the github.

20A91A05D2 / fetch-and-pull Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH <https://github.com/20A91A05D2/fetch-and-pull.git>

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# fetch-and-pull" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/20A91A05D2/fetch-and-pull.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/20A91A05D2/fetch-and-pull.git
git branch -M main
```

20A91A05D2 / fetch-and-pull Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

fetch-and-pull / one.py in main Cancel changes

Edit new file Preview Spaces 2 No wrap

```
1 print("hello")
```

Commit new file

line1 added

Add an optional extended description...

Commit new file Cancel

20A91A05D2 / fetch-and-pull Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file <> Code About

20A91A05D2 line1 added d99e1ed now 1 commit

one.py line1 added now

Help people interested in this repository understand your project by adding a README. Add a README

Releases No releases published

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
$ git log --oneline --all

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
```

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.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (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), 596 bytes | 45.00 KiB/s, done.
From https://github.com/20A91A05D2/fetch-and-pull
* [new branch]      main      -> origin/main
```

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

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
$ git log --oneline --all
d99e1ed (origin/main) line1 added
```

\*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.

\*It can be used to merge the changes that are made in the remote repository to the local repository after fetching the changes.

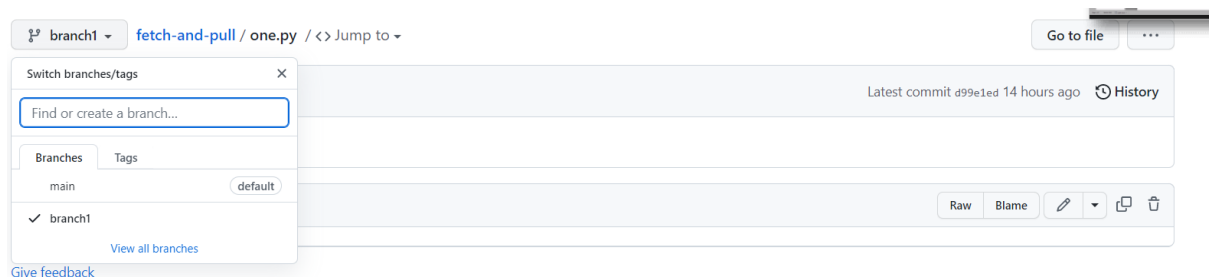
```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~ (main)
$ cd documents

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (main)
$ cd fetch-and-pull
```

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (br1)
$ git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$ git log --oneline
d99e1ed (HEAD -> main, origin/main, origin/HEAD, br1) line1 added
```

\*Now create a new branch in the remote repository and make a new commit.



**branch1** 2 branches 0 tags

[Go to file](#) [Add file](#) [Code](#)

This branch is up to date with main. [Contribute](#)

**20A91A05D2** line1 added d99e1ed 14 hours ago 1 commit

one.py line1 added 14 hours ago

Help people interested in this repository understand your project by adding a README. [Add a README](#)

\*Modify the file one.py and make a commit.

20A91A05D2 / fetch-and-pull Public [Pin](#) [Unwatch](#) 1

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

fetch-and-pull / one.py in branch1

[Edit file](#) [Preview changes](#)

Spaces

2

```
1 print("hello")
2 print("Line2")
```

**Commit changes**

Add an optional extended description...

☒ Commit directly to the **branch1** branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

[Commit changes](#) [Cancel](#)

branch1 had recent pushes less than a minute ago [Compare & pull request](#)

branch1 2 branches 0 tags [Go to file](#) [Add file](#) [Code](#)

This branch is 1 commit ahead of main. [Contribute](#)

20A91A05D2 line2 added d547923 now 2 commits

| File   | Commit      | Time |
|--------|-------------|------|
| one.py | line2 added | now  |

Help people interested in this repository understand your project by adding a README. [Add a README](#)

**About**  
No description, website  
0 stars  
1 watching  
0 forks

**Releases**  
No releases published  
[Create a new release](#)

**Packages**  
No packages published  
[Publish your first package](#)

\*We found that the branch1 is 1 commit ahead of the main

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (br1)
$ git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$ git log --oneline
d99e1ed (HEAD -> main, origin/main, origin/HEAD, br1) line1 added
```

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$ git fetch
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), 637 bytes | 31.00 KiB/s, done.
From https://github.com/20A91A05D2/fetch-and-pull
* [new branch]      branch1      -> origin/branch1
```

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$ git branch
  br1
* main
```

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$ git merge origin/branch1
Updating d99e1ed..d547923
Fast-forward
 one.py | 1 +
 1 file changed, 1 insertion(+)

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$ git log --oneline
d547923 (HEAD -> main, origin/branch1) line2 added
d99e1ed (origin/main, origin/HEAD, br1) line1 added
```

We can see that the branch1 is merged with main branch and the commit made in the branch1 is also merged with the main branch.

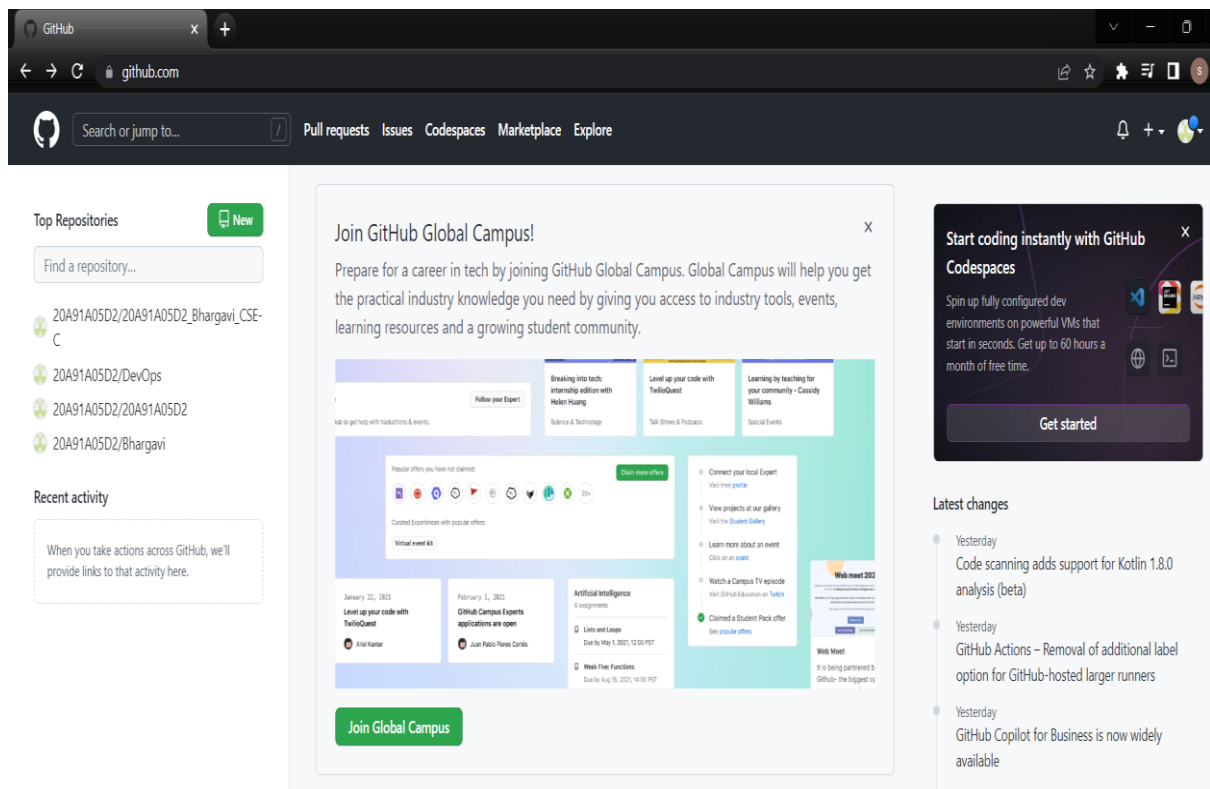
**Q3. State the difference between git fetch and git pull by doing a practical example in your git bash and attach a screenshot of all the processes.**

**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.



# Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*

20A91A05D2 ▾

Repository name \*

fetch and pull ✓

Great repository names are short and lowercase. Your new repository will be created as **fetch-and-pull**. [But expert-chainsaw?](#)

Description (optional)

☒ **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.](#)

**Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: **None** ▾

20A91A05D2 / fetch-and-pull Public

Pin Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:20A91A05D2/fetch-and-pull.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# fetch-and-pull" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:20A91A05D2/fetch-and-pull.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:20A91A05D2/fetch-and-pull.git
git branch -M main
git push -u origin main
```

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

```

Bhargavi@LAPTOP-29HRU449 MINGW64 ~ (master)
$ cd documents

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ git init
Initialized empty Git repository in C:/Users/hp/Documents/.git/

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ git config --global user.name "20A91A05D2"

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ git config --global user.email "20a91a05d2@aec.edu.in"

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ git clone https://github.com/20A91A05D2/fetch-and-pull.git
Cloning into 'fetch-and-pull'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

```

\*Check if there are any commits present in the repository.

```

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents (master)
$ cd fetch-and-pull

```

```

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
$ git log --oneline --all

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)

```

\*Create a file in the github.

20A91A05D2 / fetch-and-pull Public

Pin Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```

echo "# fetch-and-pull" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/20A91A05D2/fetch-and-pull.git
git push -u origin main

```

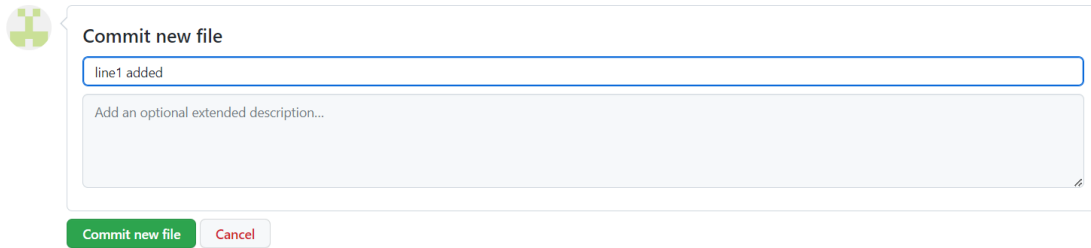
...or push an existing repository from the command line

```

git remote add origin https://github.com/20A91A05D2/fetch-and-pull.git
git branch -M main

```



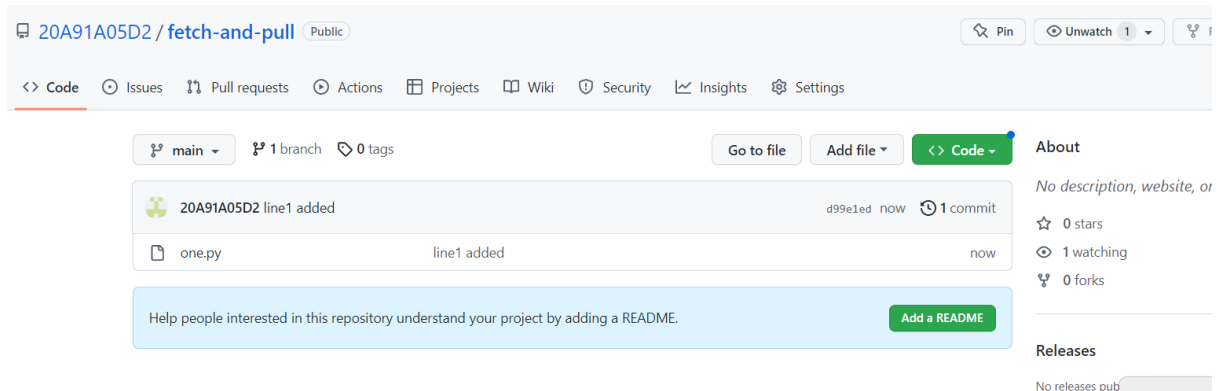
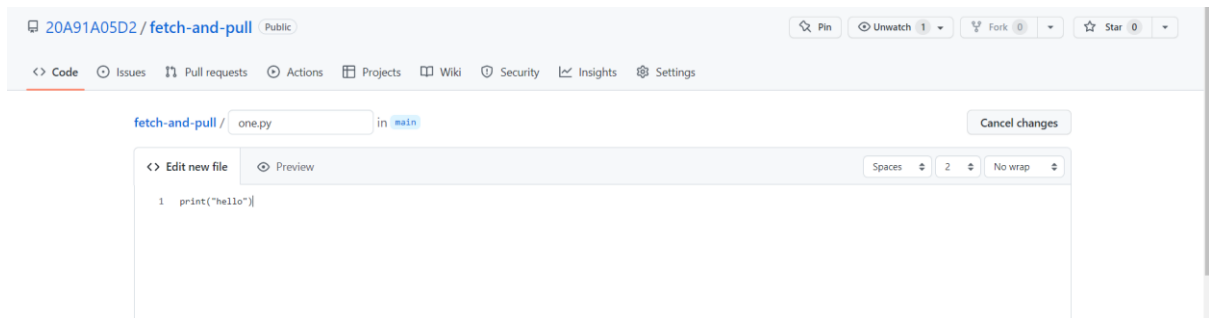


Commit new file

line1 added

Add an optional extended description...

Commit new file Cancel



```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
$ git log --oneline --all

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
```

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.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (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), 596 bytes | 45.00 KiB/s, done.
From https://github.com/20A91A05D2/fetch-and-pull
* [new branch]      main      -> origin/main
```

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

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/fetch-and-pull (main)
$ git log --oneline --all
d99e1ed (origin/main) line1 added
```

\*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.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$ git pull
Already up to date.

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$ git log --oneline --all
d99e1ed (HEAD -> main, origin/main, origin/HEAD) line1 added

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/documents/fetch-and-pull (main)
$
```

Here the HEAD->main is present which indicates that the changes have been applied to the present branch.

**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.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~ (master)
$ awk '{ print "printing using AWK command"}'

printing using AWK command
printing using AWK command
printing using AWK command
```

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~ (master)
$ cd desktop

Bhargavi@LAPTOP-29HRU449 MINGW64 ~/desktop (master)
$ vi student|
```

```
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
student [unix] (10:06 17/02/2023) 7,9  
"student" [unix] 7L, 66B
```

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/desktop (master)
$ awk '{print $NF}' student
Dept
CSE
ECE
EEE
IT
CSE
IT
```

Steps to follow bash scripting:

Step1) create the file with extension .sh.

Step 2)open the shell and write the script.

Step 3)save the code and run the code .

To run the run a code

Syntax: bash filename.sh

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/desktop (master)
$ vi prime.sh
```

```
for((i=2;i<=20;))
do
    for((j=i-1;j>=2;))
    do
        if [ `expr $i % $j` -ne 0 ] ; then
            prime=1
        else
            prime=0
            break
        fi
        j=`expr $j - 1`
    done
    if [ $prime -eq 1 ] ; then
        echo $i
    fi
    i=`expr $i + 1`
done
~
~
~
~
prime.sh[+] [unix] (10:20 17/02/2023) 11,16-30 All
-- INSERT --
```

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~/desktop (master)
$ bash prime.sh
prime.sh: line 13: [: -eq: unary operator expected
3
5
7
11
13
17
19
```

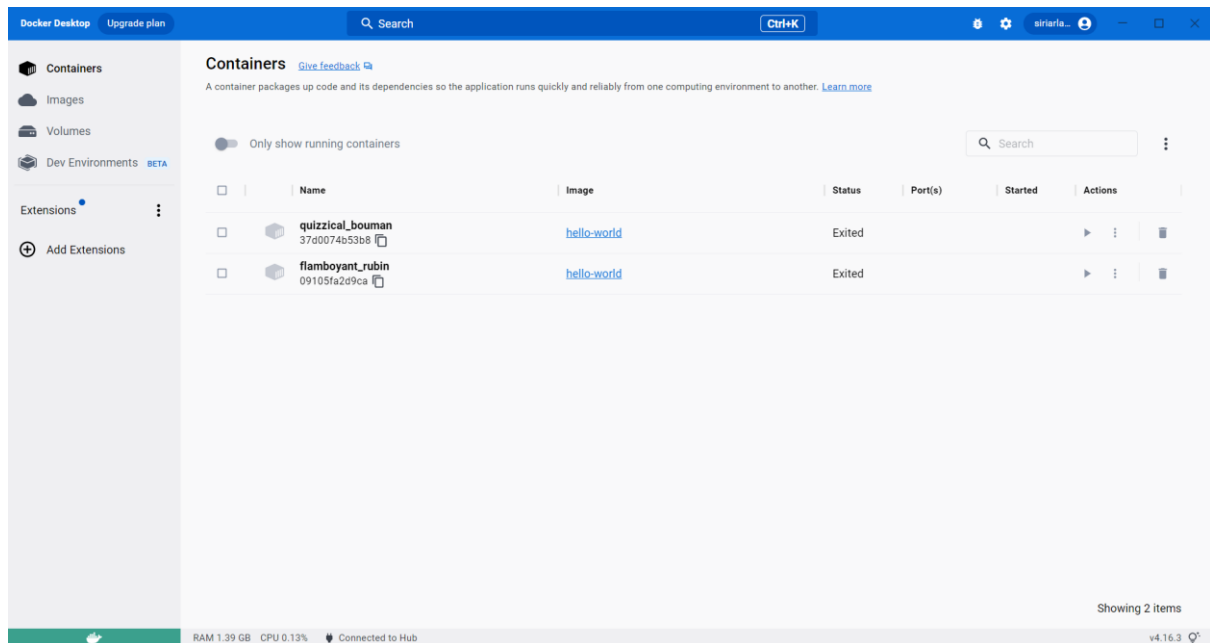
**Q5. Set up a container and run a Ubuntu operating system. For this purpose, you can make use of the docker hub and run the container in interactive mode.**

**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,

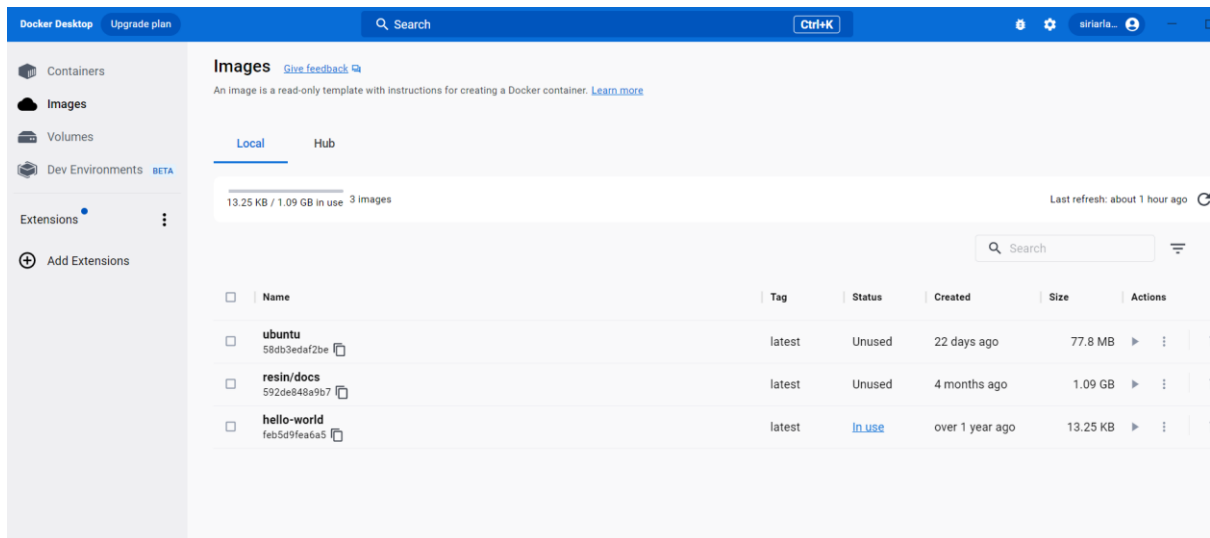
→First we need to download the image of Ubuntu from docker hub using the command **docker pull ubuntu** .

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

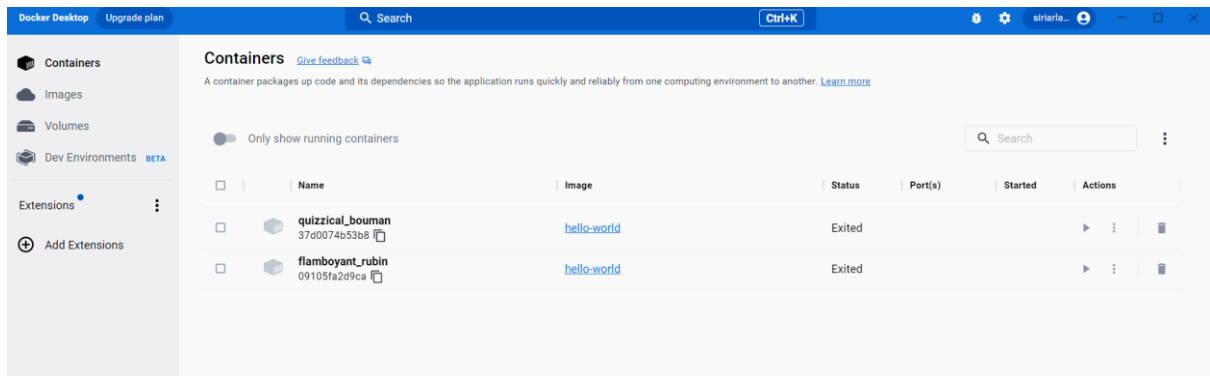
→To get an idea about the available update use **apt update** command.

\*\*Download the ubuntu OS image from the docker hub.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~ (master)
$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
677076032cca: Pulling fs layer
677076032cca: Verifying Checksum
677076032cca: Download complete
677076032cca: Pull complete
Digest: sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbb7f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```



\*An image ubuntu got downloaded but a container is not created.



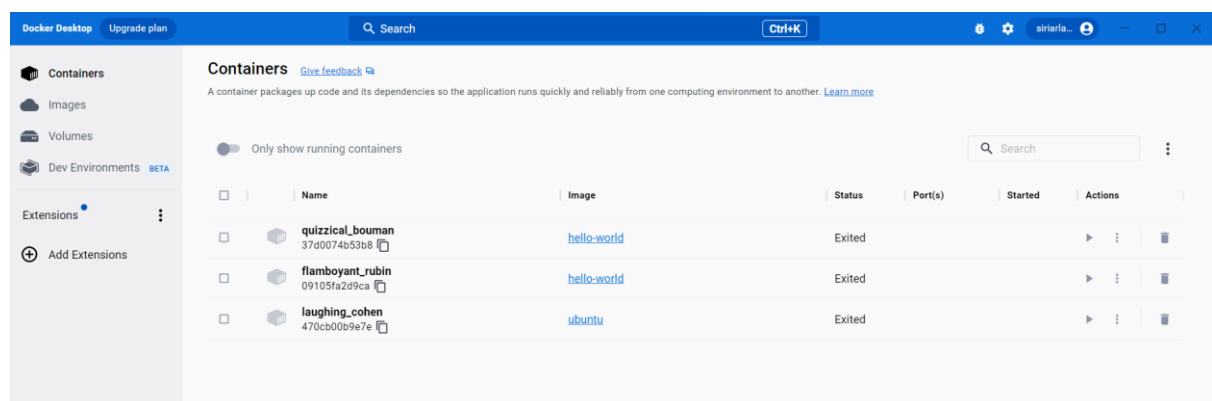
\*Now run the ubuntu image which has been downloaded.

```
Bhargavi@LAPTOP-29HRU449 MINGW64 ~ (master)
$ docker run ubuntu
```

```

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 a container got created for the ubuntu image.