1Q. 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.

1.Git Stash Command: Git stash is a built-in command with the distributed Version control tool in Git that locally stores all the most recent changes in a workspace and resets the state of the workspace to the prior commit state. A user can retrieve all files put into the stash with the git stash pop and git stash apply commands.

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new (master)
$ git clone https://github.com/garapati-vamshidhar/herovired_01.git
Cloning into 'herovired_01'...
warning: You appear to have cloned an empty repository.
```

```
MINGW64:/c/Users/vamsi/desktop/herovired01

vamsi@DESKTOP-433CL41 MINGW64 ~ (master)
$ git config --global user.name "garapati-vamshidhar"

vamsi@DESKTOP-433CL41 MINGW64 ~ (master)
$ git config --global user.email "garapativamshidhar2002@gmail.com"

vamsi@DESKTOP-433CL41 MINGW64 ~ (master)
$ cd desktop

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop (master)
$ cd herovired01

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git init
Reinitialized existing Git repository in C:/Users/vamsi/Desktop/herovired01/.git/
```

```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git add .
warning: in the working copy of 'test1.txt', LF will be replaced by CRLF the next time Git tou
ches it

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git commit -m "commited"
[master (root-commit) f14f753] commited
 1 file changed, 1 insertion(+)
 create mode 100644 test1.txt

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ vi test1.txt

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (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:   test1.txt

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

```
MINGW64:/c/Users/vamsi/desktop/herovired01
hello!!!!
~
~
~
~
~
~
~
```

```
MINGW64:/c/Users/vamsi/desktop/herovired01
hello!!!!
!!!modified!!!
~
~
~
~
~
```

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.

By default,running git stash will stash the changes that have been added to your index and unstages changes.

```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ cat test1.txt
hello!!!!
!!!modified!!!

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git stash show
 test1.txt | 1 +
 1 file changed, 1 insertion(+)

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$
```

2.Listing stashes:

We can create multiple slashes and view them using git stash list command.

```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git stash list
stash@{0}: WIP on master: f14f753 commited
stash@{1}: WIP on master: f14f753 commited
```

3.Providing additional message:

To provide more context to the stash we create the stash using the following command. git stash save “message”

4.Getting back stashed changes:

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.

```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git stash pop
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:   test1.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (40db174a2648a630f3a66d8331283e33d253d984)

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git stash list
stash@{0}: WIP on master: f14f753 commited
```

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

```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git stash apply
error: Your local changes to the following files would be overwritten by merge:
        test1.txt
Please commit your changes or stash them before you merge.
Aborting
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:   test1.txt

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

5.To view the stash summary:

Git stash show is used to view the summary

```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git stash show
 test1.txt | 1 +
 1 file changed, 1 insertion(+)
```

6.Deleting stashes:

To delete a particular stash:

git stash drop stash@{0}

To delete all stashes at once,use the below comman

git stash clear

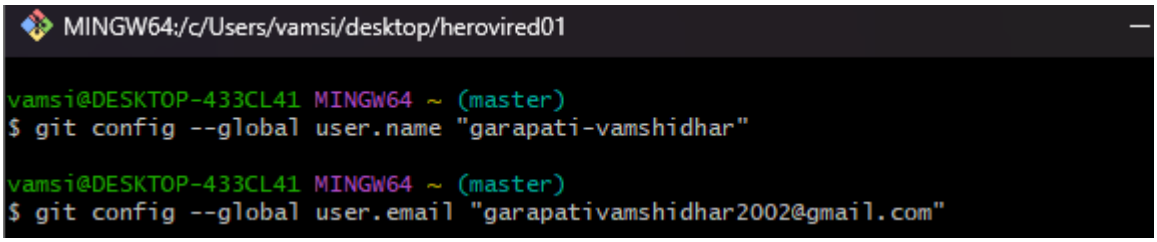
```
vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git stash clear

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
$ git stash list

vamsi@DESKTOP-433CL41 MINGW64 ~/desktop/herovired01 (master)
```

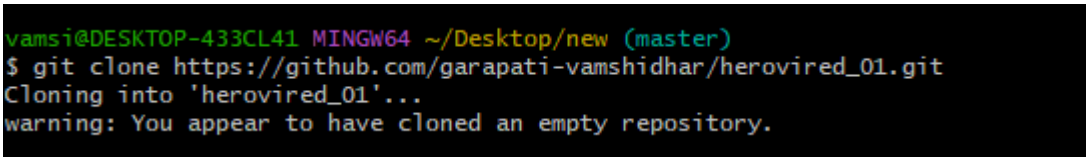
2Q. By using a sample example of your choice, use the git fetch command and also use the git merge command and describe the whole process through a screenshot with all the commands and their output in git bash.

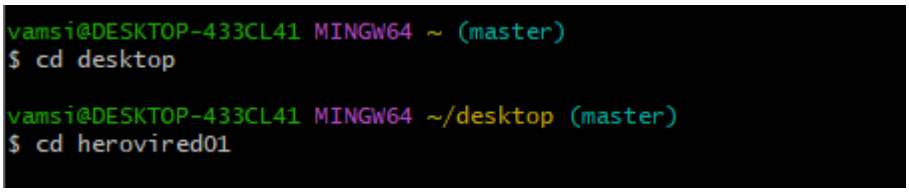
Fetch just downloads the objects and refs from a remote repository and normally updates the remote tracking branches. Pull, however, will not only download the changes, but also merges them - it is the combination of fetch and merge .The configured remote tracking branch is selected automatically.

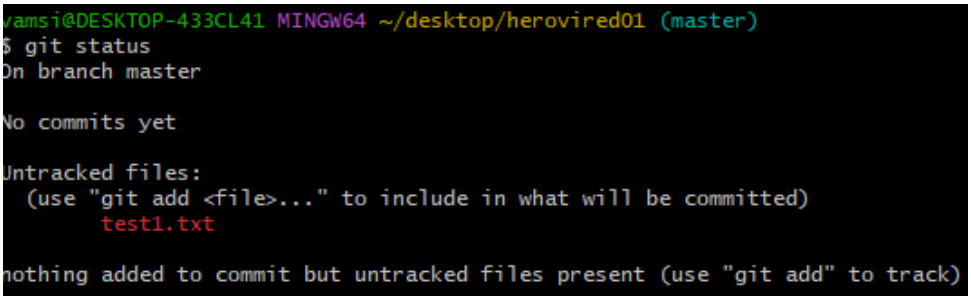
Git pull=git fetch+git merge

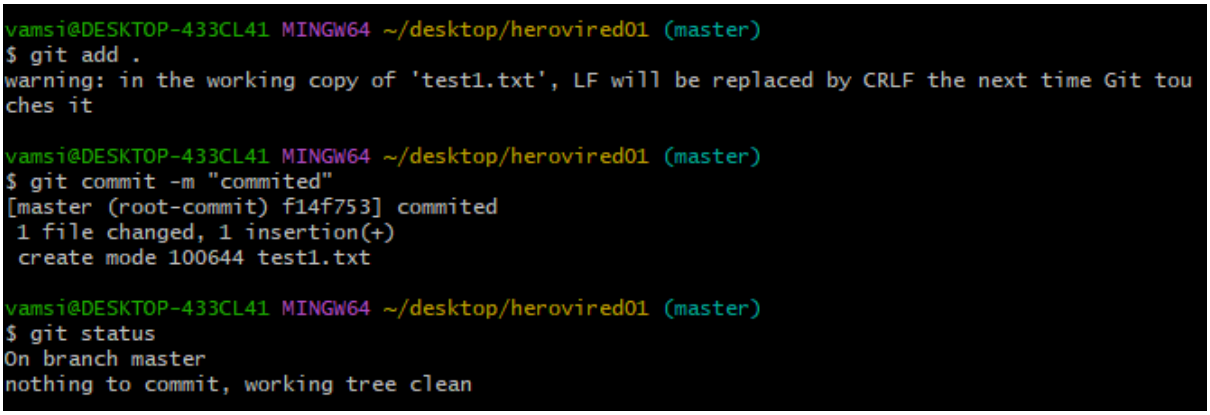
The "git merge" command. The git merge command is used to merge the branches. The **syntax:** $ git merge <query>











Git Fetch

git fetch is the command that tells your local git to retrieve the latest meta-data info from the original .

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (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), 610 bytes | 40.00 KiB/s, done.
From https://github.com/garapati-vamshidhar/herovired_01
 * [new branch]      main       -> origin/main
```

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ git log
commit 5d95d250f434ce89e59ebe4832a9ff929cae2a5b (HEAD -> main, origin/main)
Author: garapati-vamshidhar <garapativamshidhar2002@gmail.com>
Date:   Fri Feb 17 19:49:48 2023 +0530

    fine

commit daa1c8ef960f8fc9b5731e1b2c88b0cfb863076f
Author: Garapati Vamshidhar <84447482+garapati-vamshidhar@users.noreply.github.com>
Date:   Fri Feb 17 19:43:35 2023 +0530

    Create second.py

commit 659f950b9b1855fc42a0fdb1a04681f06a34a802
Author: Garapati Vamshidhar <84447482+garapati-vamshidhar@users.noreply.github.com>
Date:   Fri Feb 17 19:41:32 2023 +0530

    Create hello.py
```

Git Merge

The "git merge" command. The git merge command is used to merge the branches. The **syntax**: $ git merge <query>

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (v3)
$ git switch main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ git merge v3
Already up to date.
```

3Q. 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

```
MINGW64:/c/Users/vamsi/desktop/herovired01

vamsi@DESKTOP-433CL41 MINGW64 ~ (master)
$ git config --global user.name "garapati-vamshidhar"

vamsi@DESKTOP-433CL41 MINGW64 ~ (master)
$ git config --global user.email "garapativamshidhar2002@gmail.com"
```

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new (master)
$ git clone https://github.com/garapati-vamshidhar/herovired_01.git
Cloning into 'herovired_01'...
warning: You appear to have cloned an empty repository.
```

Git fetch:git fetch is the command that tells your local git to retrieve the latest meta-data info from the original.

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (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), 610 bytes | 40.00 KiB/s, done.
From https://github.com/garapati-vamshidhar/herovired_01
 * [new branch]      main       -> origin/main
```

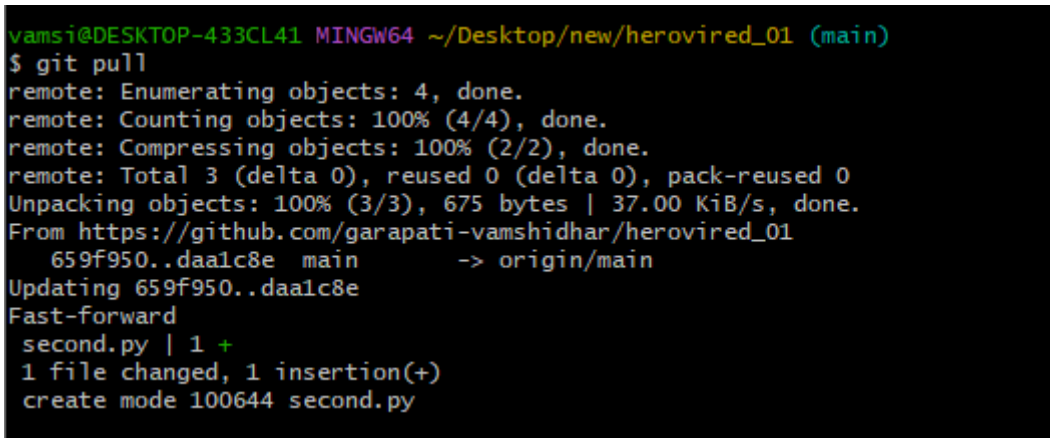
```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ git log
commit 5d95d250f434ce89e59ebe4832a9ff929cae2a5b (HEAD -> main, origin/main)
Author: garapati-vamshidhar <garapativamshidhar2002@gmail.com>
Date:   Fri Feb 17 19:49:48 2023 +0530

    fine

commit daa1c8ef960f8fc9b5731e1b2c88b0cfb863076f
Author: Garapati Vamshidhar <84447482+garapati-vamshidhar@users.noreply.github.com>
Date:   Fri Feb 17 19:43:35 2023 +0530

    Create second.py

commit 659f950b9b1855fc42a0fdb1a04681f06a34a802
Author: Garapati Vamshidhar <84447482+garapati-vamshidhar@users.noreply.github.com>
Date:   Fri Feb 17 19:41:32 2023 +0530

    Create hello.py
```

Git Push

The git push command is **used to upload local repository content to a remote repository**. Pushing is how you transfer commits from your local repository to a remote repo.

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 318 bytes | 318.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/garapati-vamshidhar/herovired_01.git
   daa1c8e..5d95d25  main -> main
```

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. The git pull command is actually a combination of two other commands, git fetch followed by git merge. In the first stage of operation git pull will execute a git fetch scoped to the local branch that HEAD is pointed at. Once the content is downloaded, git pull will enter a merge workflow. A new merge commit will be-created and HEAD updated to point at the new commit.



4Q. 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.

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ awk '{ print "hello"}'

hello

hello

hello
```

Using Awk command

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ awk '{ print "hello"}'

hello

hello

hello


vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ vi test

vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ awk '{print}' test
ROLL        DEPT
321         CSE
322         ECE
323         IT
324         CSE

vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ awk '/IT/ {print}' test
323         IT

vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ awk '{print $1}' test
ROLL
321
322
323
324
```

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

MINGW64:/c/Users/vamsi/Desktop/new/herovired_01

```
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
```

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ vi prime.sh

vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ bash prime.sh
2
3
5
7
11
13
17
19
```

(OR)

```
vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ sh prime.sh> new.txt

vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ cat new.txt
2
3
5
7
11
13
17
19

vamsi@DESKTOP-433CL41 MINGW64 ~/Desktop/new/herovired_01 (main)
$ awk '{print}' new.txt
2
3
5
7
11
13
17
19
```

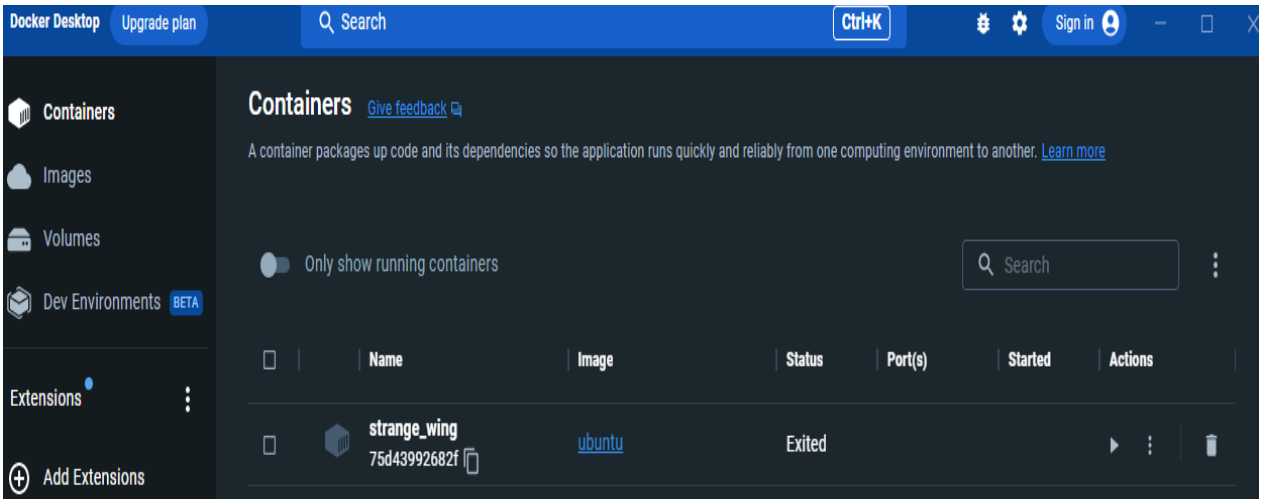
5Q. Set up a container and run a Ubuntu operating system. For this purpose, you can make use of the docker hub and run the container in interactive mode.
All the processes pertaining to this should be provided in a screenshot for grading.

Steps to follow

- 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:\Users\vamsi>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
```



```
C:\Users\vamsi>docker run -it ubuntu
root@195d3143e29c:/# apt update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [752 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [807 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [5557 B]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [860 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [808 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1136 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1091 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [10.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [49.0 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [22.4 kB]
Fetched 25.8 MB in 6min 24s (67.3 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@195d3143e29c:/#
```