

# 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 saves the previously written code and then goes back to the last commit for a fresh start. Now you can add the new feature without disturbing the old one as it is saved locally. After committing the new feature you can go on with working on the old feature which was incomplete and uncommitted.

```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~  
$ git config --global user.name "priyaanka22"  
  
Priyanka@LAPTOP-90M8TU8U MINGW64 ~  
$ git config --global user.email "priyankasatti2002@gmail.com"
```

```
MINGW64/c/Users/User/practice  
  
Priyanka@LAPTOP-90M8TU8U MINGW64 ~  
$ git clone "https://github.com/priyaanka22/practice.git"  
Cloning into 'practice'...  
warning: You appear to have cloned an empty repository.  
  
Priyanka@LAPTOP-90M8TU8U MINGW64 ~  
$ cd practice  
  
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)  
$ vi file1.txt  
  
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)  
$ git status  
On branch main  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    file1.txt  
  
nothing added to commit but untracked files present (use "git add" to track)  
  
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)  
$ git add .  
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it  
  
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)  
$ git status  
On branch main  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   file1.txt  
  
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)  
$ vi file1.txt
```


```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt

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:   file1.txt
```

Modified file

 MINGW64:/c/Users/User/practice

```
Hello this is file 1
This is priyanka
```

```
~
~
~
~
~
~
~
~
```

The file is now modified, and it is committed.

Now we would be modifying the file again as shown below

```
Hello this is file 1
This is priyanka
This is modified file
```

```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git status
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
    (use "git branch --unset-upstream" to fixup)

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:   file1.txt

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

The file is now modified, 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.

```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it
Saved working directory and index state WIP on main: f550c9b changes are committed

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$
```

MINGW64:/c/Users/User/practice

```
Hello this is file 1
This is priyanka
```



The changes are removed now.

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 the changes. To stash your untracked files, use `git stash -u`.

## 1. Listing stashes:

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

```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash list
stash@{0}: WIP on main: f550c9b changes are committed

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ |
```

## 2. Providing additional message:

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

git stash save "message"

### 3. Getting back stashed changes:

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

'git stash pop' removes the changes from stash and reapplies the changes in working copy,

'git stash apply' do not remove changes, but reapplies the changes in working copy.

```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash list
stash@{0}: WIP on main: f550c9b changes are committed

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash pop
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

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:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (eef234b6357170ef63bec8656e2fa026f97f03a0)

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ |
```

The stash is dropped.

```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash list

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ cat file1
cat: file1: No such file or directory

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ cat file1.txt
Hello this is file 1
This is priyanka
This is modified file

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ |
```

We got the previous uncommitted changes.

### 4. To view the stash summary:

Git stash show

```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash list
stash@{0}: WIP on main: f550c9b changes are committed

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash apply
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)

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:   file1.txt

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

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash show
file1.txt | 1 +
1 file changed, 1 insertion(+)

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ |
```

## 5. Deleting stashes:

To delete a particular stash:

```
git stash drop stash@{1}
```

To delete all stashes at once, use the below command

```
git stash clear
```

```
Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash clear

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ git stash list

Priyanka@LAPTOP-90M8TU8U MINGW64 ~/practice (main)
$ |
```

**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.

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 (cf. the section called “Merging”). The configured remote tracking branch is selected automatically.

### Git pull=git fetch + git merge

MINGW64:/c/Users/User/git\_priya

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~
$ git config --global user.name "priyaanka22"

priyanka@LAPTOP-90M8TU8U MINGW64 ~
$ git config --global user.email "priyankasatti2002@gmail.com"

priyanka@LAPTOP-90M8TU8U MINGW64 ~
$ git clone https://github.com/priyaanka22/git_priya.git
fatal: destination path 'git_priya' already exists and is not an empty directory
.

priyanka@LAPTOP-90M8TU8U MINGW64 ~
$ cd git_priya

priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)
$ vim first

priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)
$ |
```

MINGW64:/c/Users/User/git\_priya

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)
$ git add .
warning: in the working copy of 'first', LF will be replaced by CRLF the next time Git touches it

priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   first

priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)
$ git commit -m "created another file first"
[main 3b31093] created another file first
 1 file changed, 1 insertion(+)
 create mode 100644 first

priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)
$ git branch
* main
```

### GIT FETCH

Git fetch is the command that tells your local git to retrieve the latest meta-data info from the original (yet doesn't do any file transferring. It's more like just checking to see if there are any changes available).

```
priyanka@LAPTOP-9OM8TU8U MINGW64 ~/git_priya (main)
$ git fetch

priyanka@LAPTOP-9OM8TU8U MINGW64 ~/git_priya (main)
$ git log
commit f60a65a0e3ff51cb06ed08779acc76f70ab4bc2b (HEAD -> main, b2)
Author: priyaanka22 <priyankasatti2002@gmail.com>
Date:   Fri Feb 17 21:23:26 2023 +0530

    files are committed

commit 3b31093b7d43d27332bc4dd7c3ed1a4e93db2761 (b1)
Author: priyaanka22 <priyankasatti2002@gmail.com>
Date:   Fri Feb 17 21:11:34 2023 +0530

    created another file first

commit bca15937e9d511cd0774ad16b11badb2800df581 (origin/main)
Author: priyaanka22 <80589016+priyaanka22@users.noreply.github.com>
Date:   Fri Feb 17 12:37:36 2023 +0530

    Create first.py

priyanka@LAPTOP-9OM8TU8U MINGW64 ~/git_priya (main)
$ |
```

## GIT MERGE

The git merge command is used to merge the branches. The syntax for the git merge command is as: \$ git merge <query>



```

priyanka@LAPTOP-9QM8TUSU MINGW64 ~/git_priya (b2)
$ git status
On branch b2
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

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

priyanka@LAPTOP-9QM8TUSU MINGW64 ~/git_priya (b2)
$ git add .
warning: in the working copy of 'first', LF will be replaced by CRLF the next time Git touches it

priyanka@LAPTOP-9QM8TUSU MINGW64 ~/git_priya (b2)
$ git commit -m "files are committed"
[b2 f60a65a] files are committed
1 file changed, 1 insertion(+), 1 deletion(-)

priyanka@LAPTOP-9QM8TUSU MINGW64 ~/git_priya (b2)
$ 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)

priyanka@LAPTOP-9QM8TUSU MINGW64 ~/git_priya (main)
$ git merge b2
Updating 3b31093..f60a65a
Fast-forward
  first | 2 +-
  1 file changed, 1 insertion(+), 1 deletion(-)

```

**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:

The git fetch command is used to get the updates that have been pushed to our remote branches to local machines.

#### GIT PULL:

Git pull=Git fetch +Git merge

Git Pull brings the copy of the remote directory changes into the local repository

#### **Step 1) Git clone an empty repository**

First give the username and email using git config command

Then create a new repository and clone using git clone command

MINGW64:/c/Users/User

```
priyanka@LAPTOP-9OM8TU8U MINGW64 ~
$ git config --global user.name "priyaanka22"

priyanka@LAPTOP-9OM8TU8U MINGW64 ~
$ git config --global user.email "priyankasatti2002@gmail.com"

priyanka@LAPTOP-9OM8TU8U MINGW64 ~
$ git clone https://github.com/priyaanka22/git_priya.git
Cloning into 'git_priya'...
warning: You appear to have cloned an empty repository.

priyanka@LAPTOP-9OM8TU8U MINGW64 ~
$ |
```

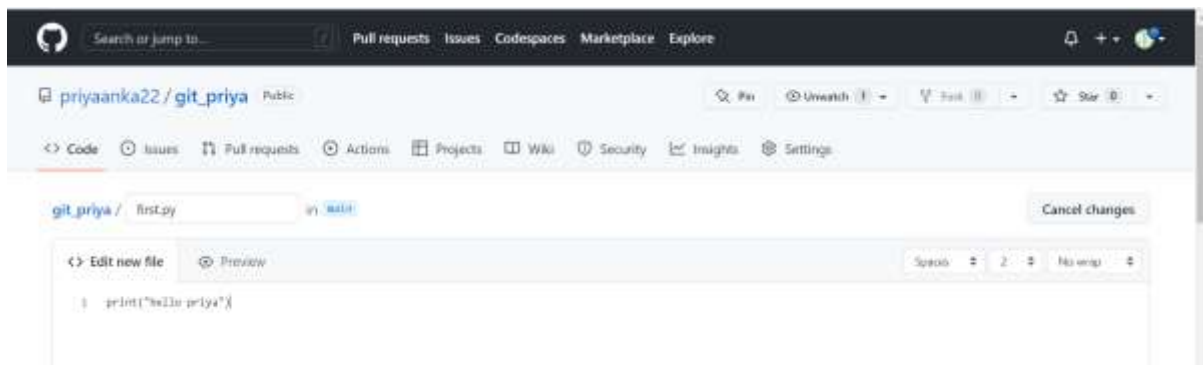
## Step 2) Git log --oneline --all

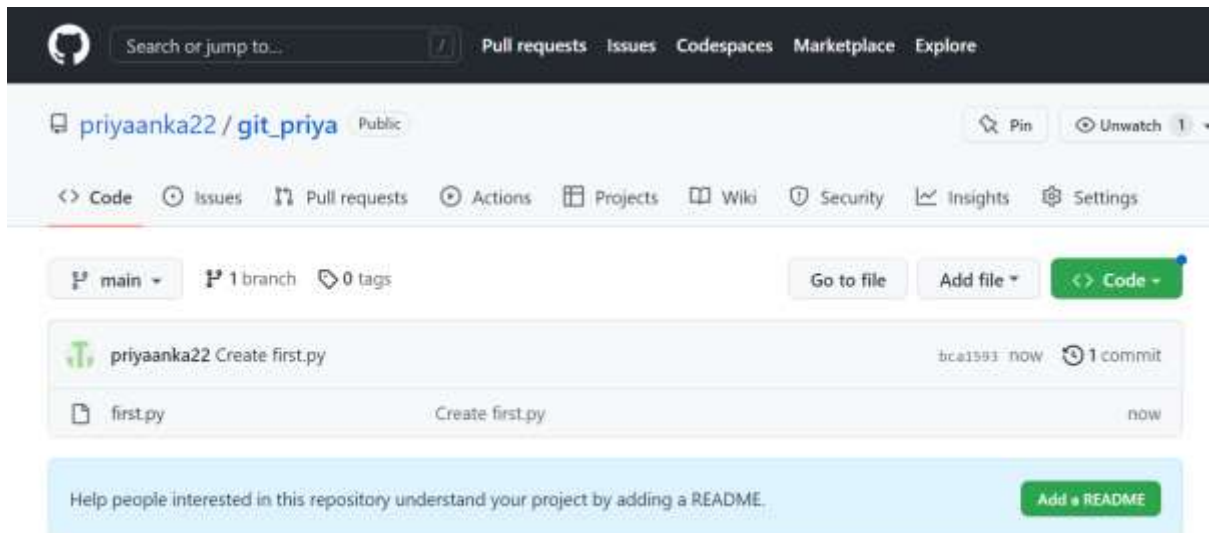
```
priyanka@LAPTOP-9OM8TU8U MINGW64 ~
$ git log --oneline --all
fatal: not a git repository (or any of the parent directories): .git

priyanka@LAPTOP-9OM8TU8U MINGW64 ~
$ |
```

This command gives no output since no commit has been done yet.

Now go to github and go to the repository which we cloned and create a file and then update it.





Step 3) In the git bash, go to the cloned repository using cd command

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~  
$ cd git_priya  
  
priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)  
$ git log --online --all  
fatal: unrecognized argument: --online  
  
priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)  
$ git log --oneline --all  
  
priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)  
$ |
```

We have moved to the cloned repository and use the git log command to view the commit history, but here we cannot see any commits ,even though we made one commit in the remote repository.

Step 4) Git fetch

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (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), 605 bytes | 21.00 KiB/s, done.  
From https://github.com/priyaanka22/git_priya  
* [new branch]      main      -> origin/main  
  
priyanka@LAPTOP-90M8TU8U MINGW64 ~/git_priya (main)  
$ |
```

Git fetch will download the changes in remote repository, fetch commands just downloads the changes made in the remote repository.

Once the git fetch command is used, it downloads the changes made in the remote repository.

```
priyanka@LAPTOP-9OM8TU8U MINGW64 ~/git_priya (main)
$ git log --oneline --all
bca1593 (origin/main) Create first.py

priyanka@LAPTOP-9OM8TU8U MINGW64 ~/git_priya (main)
$ git pull

priyanka@LAPTOP-9OM8TU8U MINGW64 ~/git_priya (main)
$ git log --oneline
bca1593 (HEAD -> main, origin/main) Create first.py
```

Here when git log command is used then, it shows one commit made in the remote repository.


Here one commit that is made in the remote repository is being shown, but it is not applied. To apply and download we use the git pull command.

**Git pull == It downloads and merges**

Now the commits are applied to the main branch by using the git pull command.

**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.

 MINGW64:/c/Users/User/documents

```
priyanka@LAPTOP-9OM8TU8U MINGW64 ~
$ awk '{print "This is the awk command"}'

This is the awk command
This is the awk command
This is the awk command
This is the awk command
```

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~  
$ cd documents  
  
priyanka@LAPTOP-90M8TU8U MINGW64 ~/documents  
$ vi persons
```

 MINGW64:/c/Users/User/documents

Name	Dept
Riya	CSE
Shubh	MECH
Raj	EEE
Sai	ECE
Indu	IT

~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~/documents  
$ awk '{print}' persons  
Name      Dept  
Riya      CSE  
Shubh     MECH  
Raj       EEE  
Sai       ECE  
Indu      IT
```

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~/documents  
$ awk '/MECH/ {print}' persons  
Shubh     MECH
```

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~/documents  
$ awk '{print $1}' persons  
Name  
Riya  
Shubh  
Raj  
Sai  
Indu
```

above command will print column1

### Steps to follow bash scripting:


Step 1) create the file with the 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

```
priyanka@LAPTOP-9OM8TU8U MINGW64 ~/documents
$ vi prime.sh
```

 MINGW64:/c/Users/User/documents

```
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
~
~
~
~
```

```
priyanka@LAPTOP-90M8TU8U MINGW64 ~/documents
$ bash 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.

Steps for setting up a container and run an Ubuntu OS.

Step 1) Download the Ubuntu image.

Syntax: docker pull ubuntu

Step 2) To bring the Ubuntu image.

Syntax: docker run -it ubuntu

NOTE: “-it” option runs the container in an interactive mode and opens up a shell within the ubuntu OS.

Step 3) To get an idea about the available update.

Syntax: apt update.

```
C:\Users\User>docker pull ubuntu
Using default tag; latest
latest: Pulling from library/ubuntu
677076032cca: Pull complete
Digest: sha256:9a0bddde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbbe7f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

C:\Users\User>docker run -it ubuntu
root@0f1745f47496:/#
```

```
C:\Users\User>docker run -it ubuntu
root@3d468b4235e0:/# apt update
Ign:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Ign:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Ign:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Ign:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Ign:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Ign:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Ign:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Ign:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Ign:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Err:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
  Could not connect to security.ubuntu.com:80 (185.125.190.39). - connect (113:
13: No route to host) Could not connect to security.ubuntu.com:80 (91.189.91.39)
8). - connect (113: No route to host)
Err:1 http://archive.ubuntu.com/ubuntu jammy InRelease
  Could not connect to archive.ubuntu.com:80 (185.125.190.36). - connect (113: N
No route to host) Could not connect to archive.ubuntu.com:80 (91.189.91.38). - c
- connect (113: No route to host)
Err:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
  Unable to connect to archive.ubuntu.com:80:
Err:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
  Unable to connect to archive.ubuntu.com:80:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```



