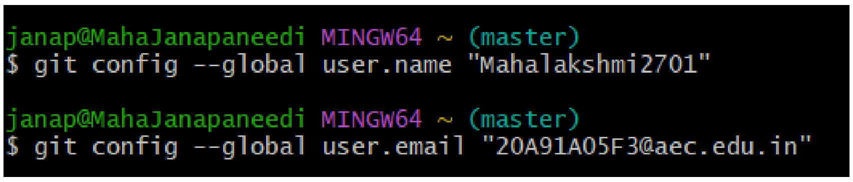
DevOps Assignment

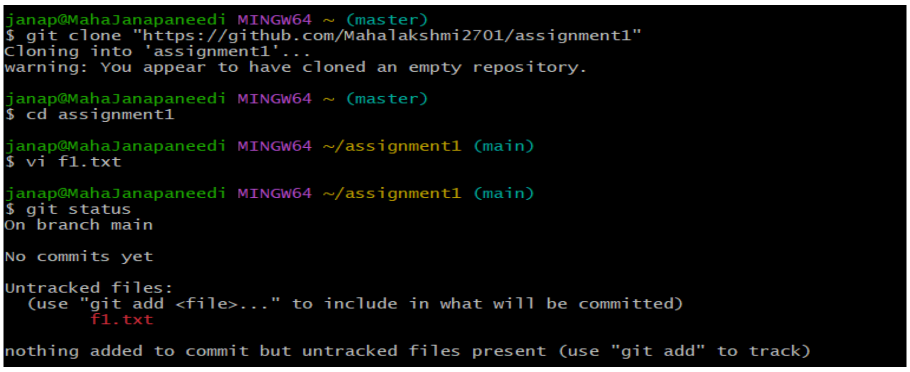
Q1. Describe the usage of the git stash command by using an example and also state the process by giving the screenshot of all the commands written in git bash.

1. Git Stash Command:

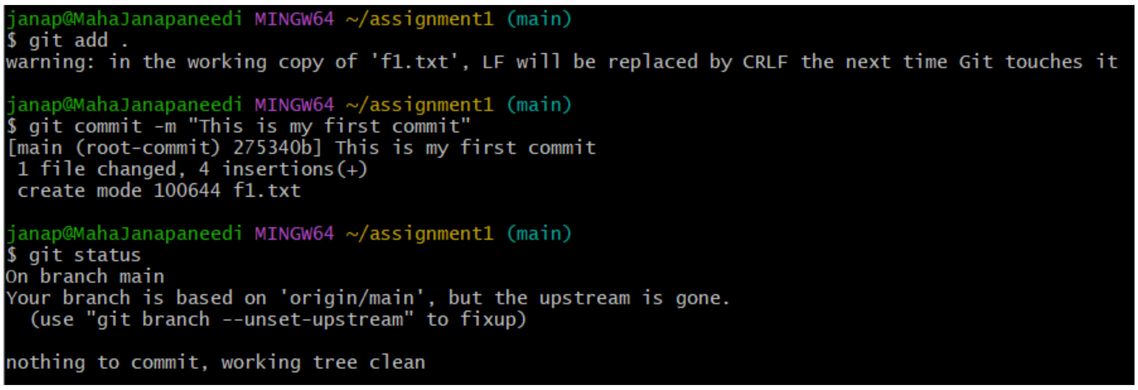
Git stash is a built-in command with the distributed Version control tool in Git that **locally stores all the most recent changes in a workspace and resets the state of the workspace to the prior commit state**.
The Git stash command saves the previously written code and then goes back to the last commit 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 working with the old one which was incomplete and not committed.



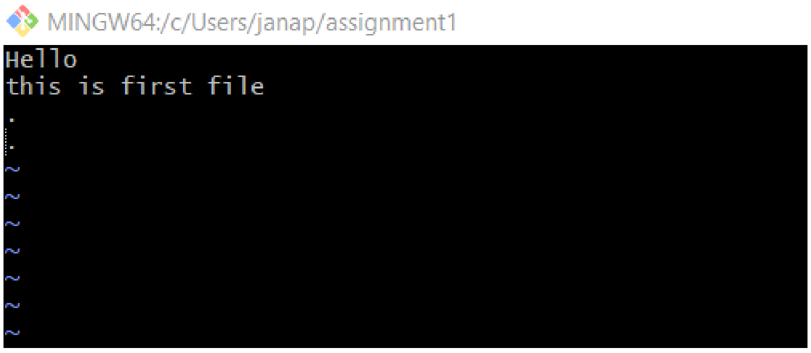
```
janap@MahaJanapaneedi MINGW64 ~ (master)
$ git config --global user.name "Mahalakshmi2701"

janap@MahaJanapaneedi MINGW64 ~ (master)
$ git config --global user.email "20A91A05F3@aec.edu.in"
```



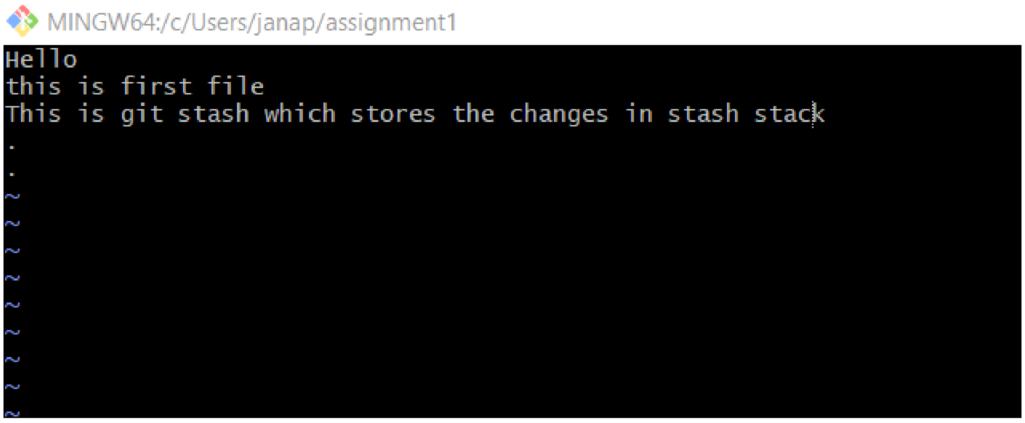
```
janap@MahaJanapaneedi MINGW64 ~ (master)
$ git clone "https://github.com/Mahalakshmi2701/assignment1"
Cloning into 'assignment1'...
warning: You appear to have cloned an empty repository.

janap@MahaJanapaneedi MINGW64 ~ (master)
$ cd assignment1

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ vi f1.txt

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        f1.txt

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



```
janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git add .
warning: in the working copy of 'f1.txt', LF will be replaced by CRLF the next time Git touches it

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git commit -m "This is my first commit"
[main (root-commit) 275340b] This is my first commit
 1 file changed, 4 insertions(+)
 create mode 100644 f1.txt

janap@MahaJanapaneedi MINGW64 ~/assignment1 (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)

nothing to commit, working tree clean
```

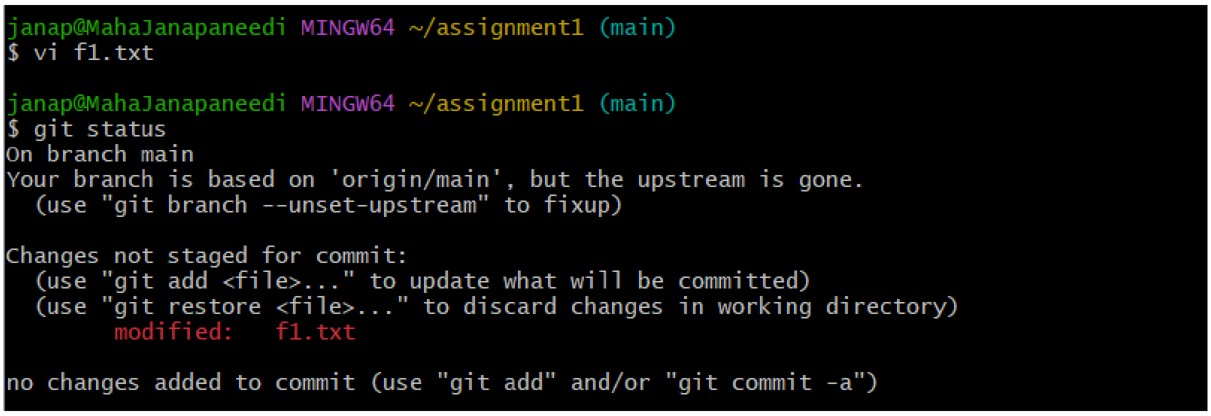


```
MINGW64:/c/Users/janap/assignment1
Hello
this is first file
.
.
~
~
~
~
~
~
~
```



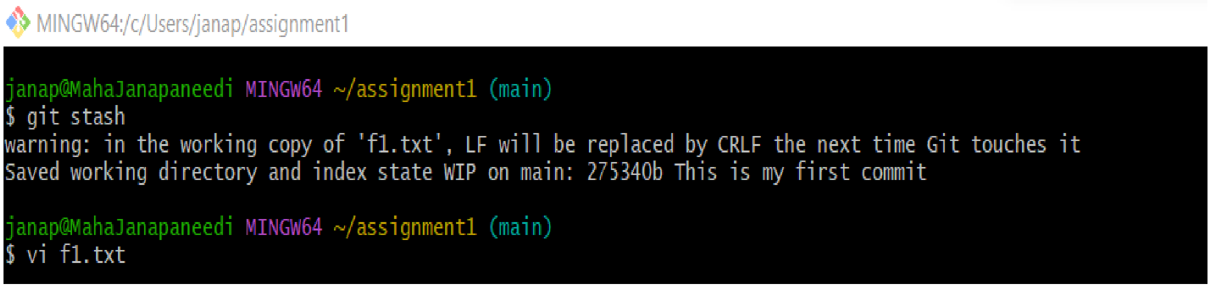
```
MINGW64:/c/Users/janap/assignment1
Hello
this is first file
This is git stash which stores the changes in stash stack
.
.
~
~
~
~
~
~
~
~
~
```

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 the git stash command.

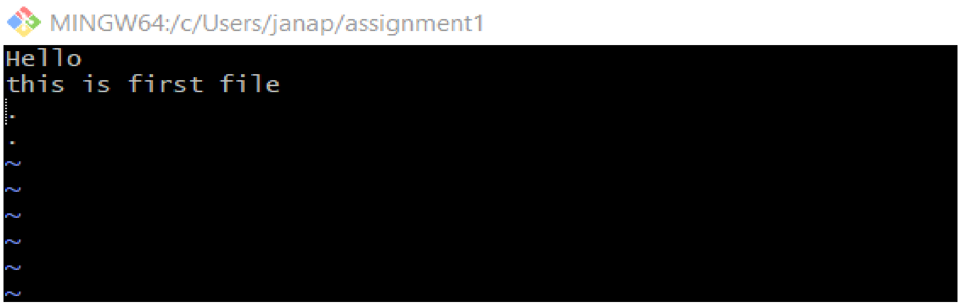


```
janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ vi f1.txt

janap@MahaJanapaneedi MINGW64 ~/assignment1 (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:   f1.txt

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

Now changes are removed.



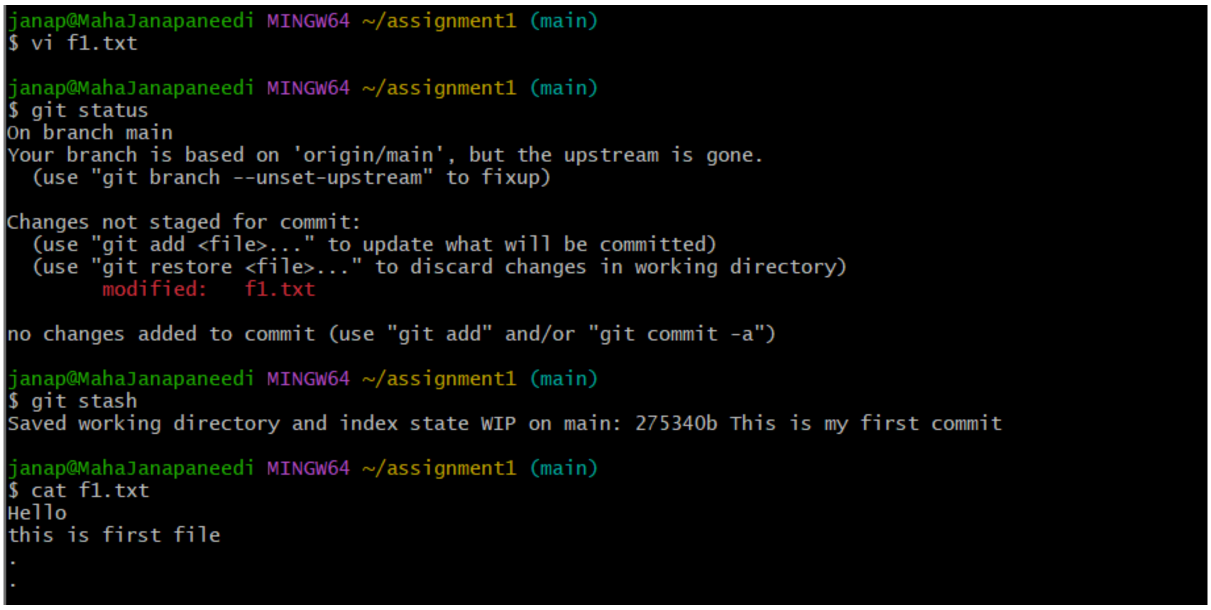
```
MINGW64:/c/Users/janap/assignment1

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash
warning: in the working copy of 'f1.txt', LF will be replaced by CRLF the next time Git touches it
Saved working directory and index state WIP on main: 275340b This is my first commit

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ vi f1.txt
```



```
MINGW64:/c/Users/janap/assignment1
Hello
this is first file
.
.
~
~
~
~
~
~
```

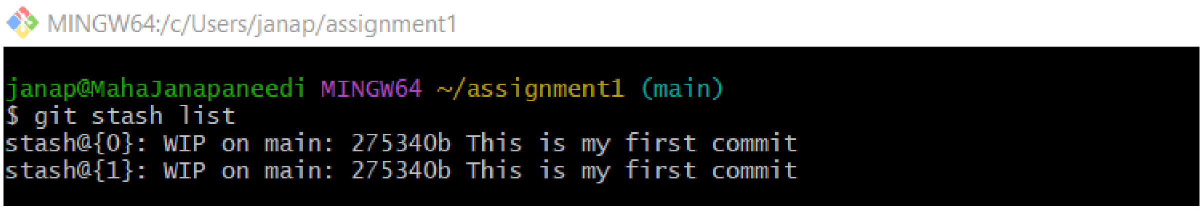
The file is now stashed and it is in an untracked state.

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



```
janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ vi f1.txt

janap@MahaJanapaneedi MINGW64 ~/assignment1 (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:   f1.txt

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

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash
Saved working directory and index state WIP on main: 275340b This is my first commit

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ cat f1.txt
Hello
this is first file
.
.
```

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



```
MINGW64:/c/Users/janap/assignment1

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash list
stash@{0}: WIP on main: 275340b This is my first commit
stash@{1}: WIP on main: 275340b This is my first commit
```

3. Providing additional message:

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

git stash save “message”

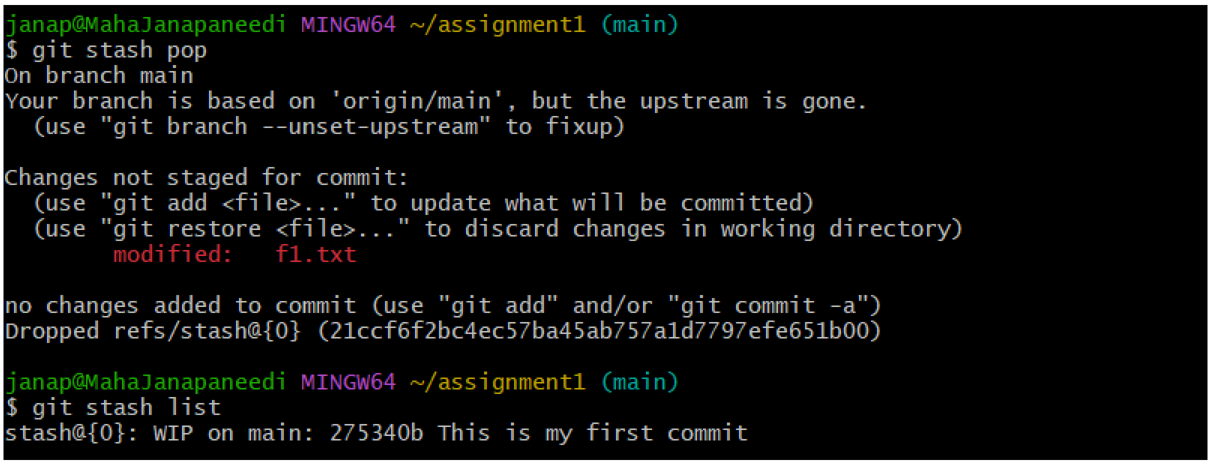
4. Getting back stashed changes:

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

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

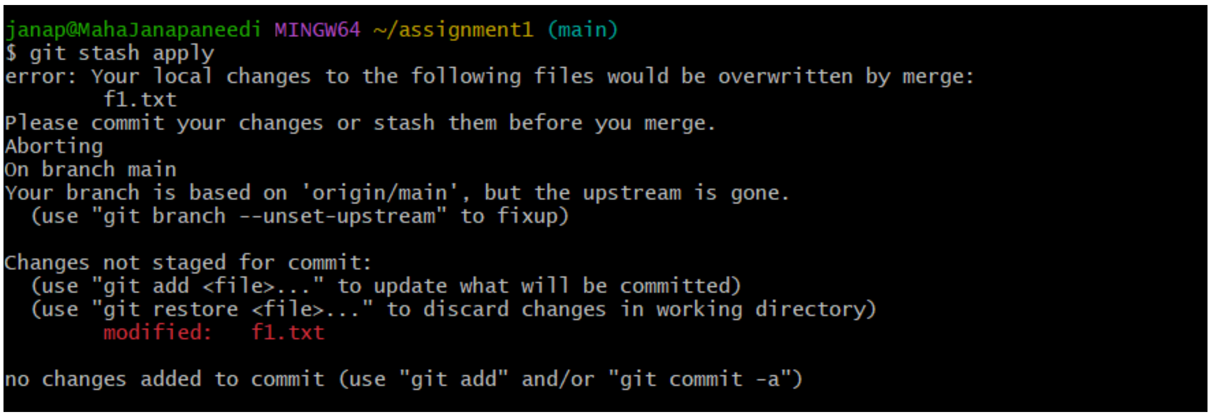
2.‘git stash apply’ do not remove changes .but reapplies the changes in working copy.

Now check whether stash is removed or not.

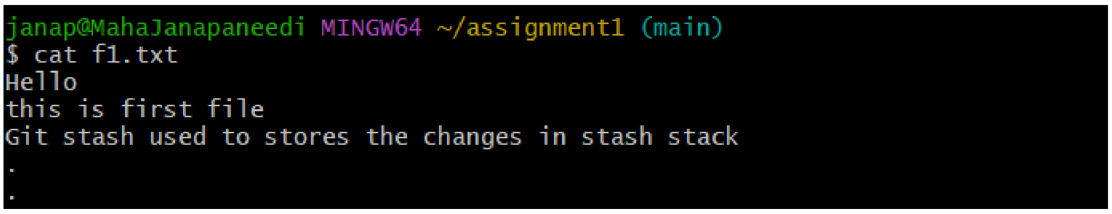


```
janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash pop
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

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:   f1.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (21ccf6f2bc4ec57ba45ab757a1d7797efe651b00)

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash list
stash@{0}: WIP on main: 275340b This is my first commit
```

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



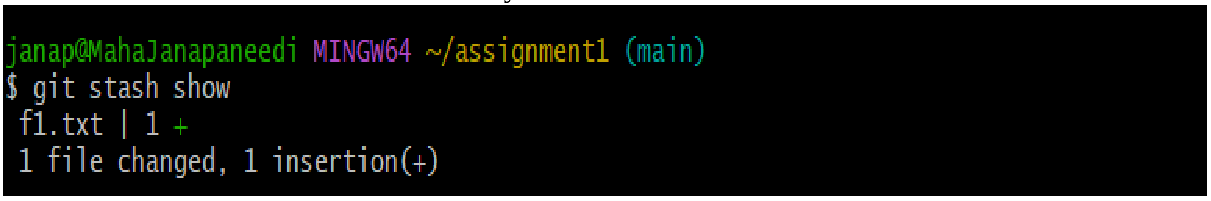
```
janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash apply
error: Your local changes to the following files would be overwritten by merge:
        f1.txt
Please commit your changes or stash them before you merge.
Aborting
On branch main
Your branch is based on 'origin/main', but the upstream 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:   f1.txt

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



```
janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ cat f1.txt
Hello
this is first file
Git stash used to stores the changes in stash stack
.
.
```

5.To view the stash summary:

Git stash show is used to view the summary



```
janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash show
 f1.txt | 1 +
 1 file changed, 1 insertion(+)
```

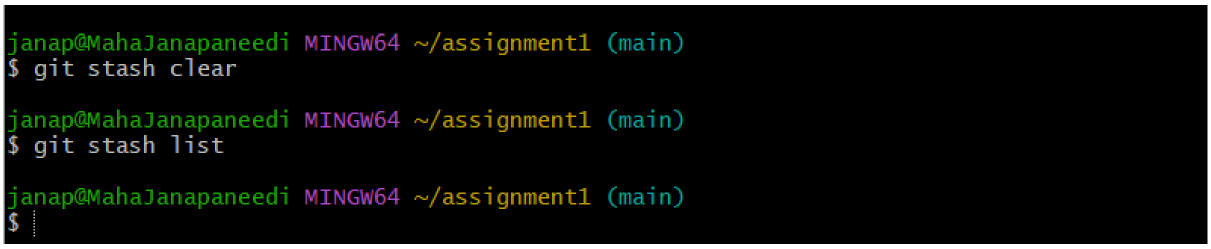
6. Deleting stashes:

To delete a particular stash:

git stash drop stash@{1}

To delete all stashes at once, use the below command.

git stash clear



```
janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash clear

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ git stash list

janap@MahaJanapaneedi MINGW64 ~/assignment1 (main)
$ 
```

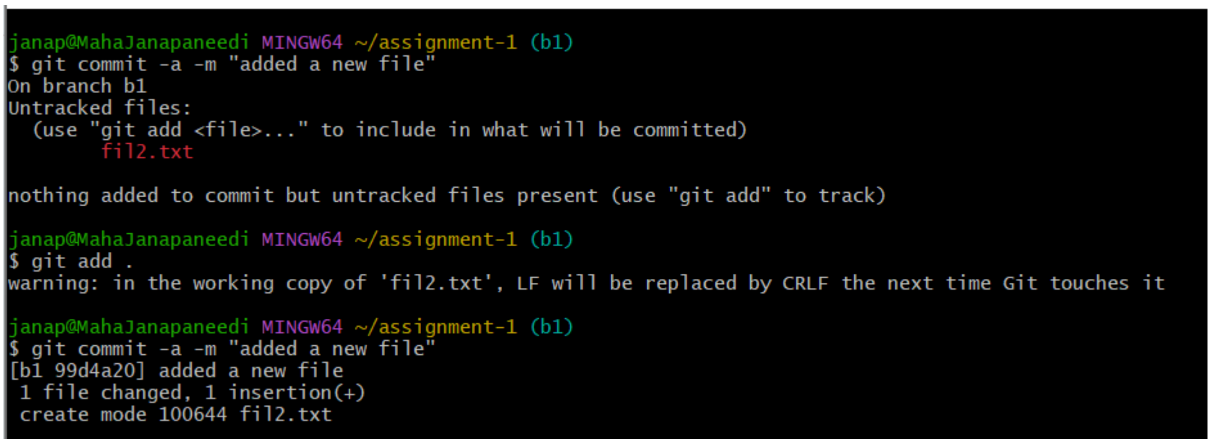
Q2. By using a sample example of your choice, use the git fetch command and also use the git merge command and describe the whole process through a screenshot with all the commands and their output in git bash.

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.

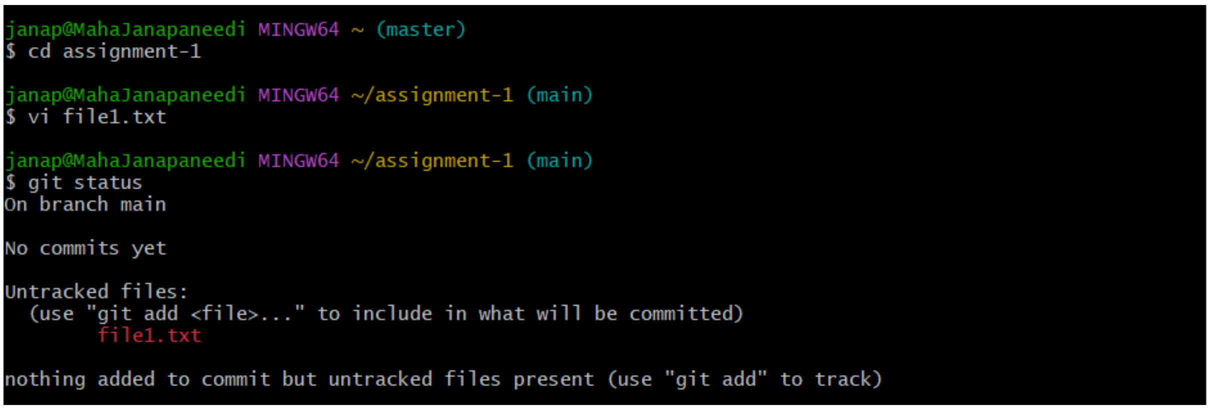
Git pull=git fetch+git merge

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).

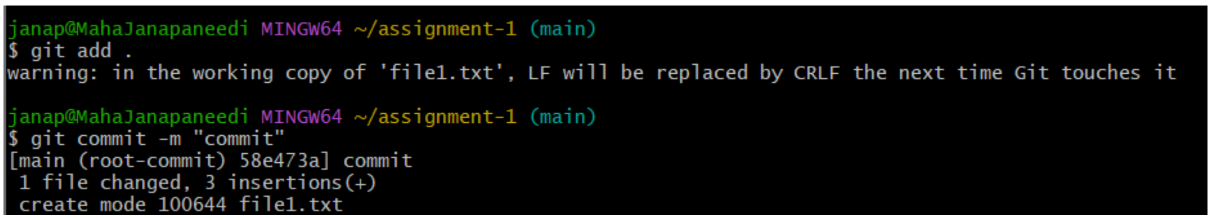
The "git merge" command. The git merge command is used to merge the branches. The syntax for the git merge command is as: $ git merge <query>



```
janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git commit -a -m "added a new file"
On branch b1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        fil2.txt

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

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git add .
warning: in the working copy of 'fil2.txt', LF will be replaced by CRLF the next time Git touches it

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git commit -a -m "added a new file"
[b1 99d4a20] added a new file
 1 file changed, 1 insertion(+)
 create mode 100644 fil2.txt
```



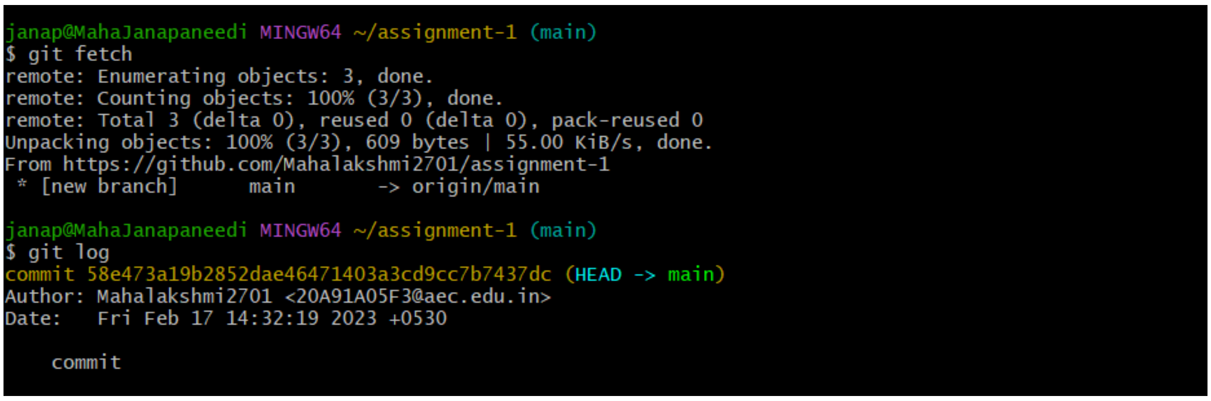
```
janap@MahaJanapaneedi MINGW64 ~ (master)
$ cd assignment-1

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (main)
$ vi file1.txt

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (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)
```



```
janap@MahaJanapaneedi MINGW64 ~/assignment-1 (main)
$ git add .
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (main)
$ git commit -m "commit"
[main (root-commit) 58e473a] commit
 1 file changed, 3 insertions(+)
 create mode 100644 file1.txt
```

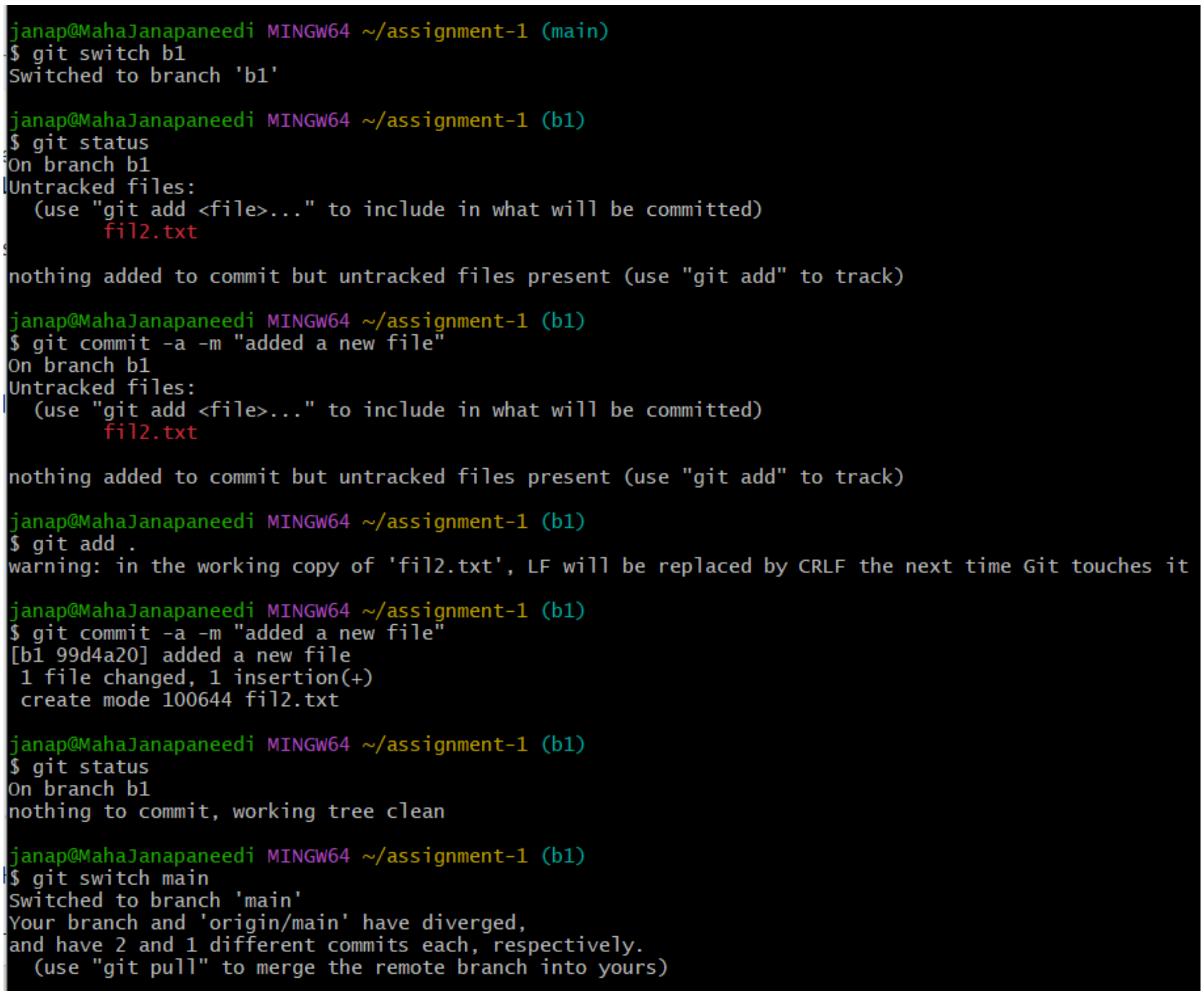
Git Fetch

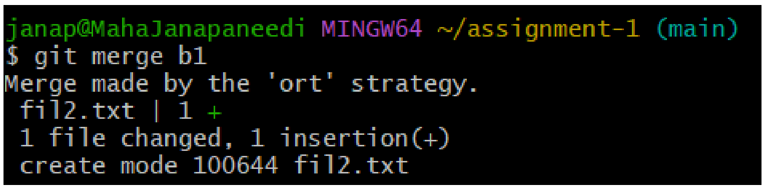
git fetch is the command that tells your local git to retrieve the latest meta-data info from the original (yet doesn’t do any file transferring. It’s more like just checking to see if there are any changes available).


```
janap@MahaJanapaneedi MINGW64 ~/assignment-1 (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), 609 bytes | 55.00 KiB/s, done.
From https://github.com/Mahalakshmi2701/assignment-1
 * [new branch]      main       -> origin/main

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (main)
$ git log
commit 58e473a19b2852dae46471403a3cd9cc7b7437dc (HEAD -> main)
Author: Mahalakshmi2701 <20A91A05F3@aec.edu.in>
Date:   Fri Feb 17 14:32:19 2023 +0530

    commit
```

Git Merge :

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


```
janap@MahaJanapaneedi MINGW64 ~/assignment-1 (main)
$ git switch b1
Switched to branch 'b1'

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git status
On branch b1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        fil2.txt

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

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git commit -a -m "added a new file"
On branch b1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        fil2.txt

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

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git add .
warning: in the working copy of 'fil2.txt', LF will be replaced by CRLF the next time Git touches it

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git commit -a -m "added a new file"
[b1 99d4a20] added a new file
 1 file changed, 1 insertion(+)
 create mode 100644 fil2.txt

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git status
On branch b1
nothing to commit, working tree clean

janap@MahaJanapaneedi MINGW64 ~/assignment-1 (b1)
$ git switch main
Switched to branch 'main'
Your branch and 'origin/main' have diverged,
and have 2 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
```


```
janap@MahaJanapaneedi MINGW64 ~/assignment-1 (main)
$ git merge b1
Merge made by the 'ort' strategy.
 fil2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 fil2.txt
```

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.

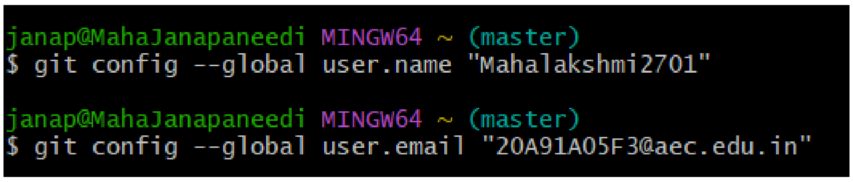
Git fetch: Git fetch is a command that allows you to download objects from a remote repository but it doesn't integrate any of this new data into your working files.

Fetch just downloads the objects and references from a remote repository and normally updates the remote tracking branches.
Git pull: Git pull is a command that allows you to fetch from and integrate with another repository or local branch. It updates your current HEAD branch with the latest changes from the remote server.

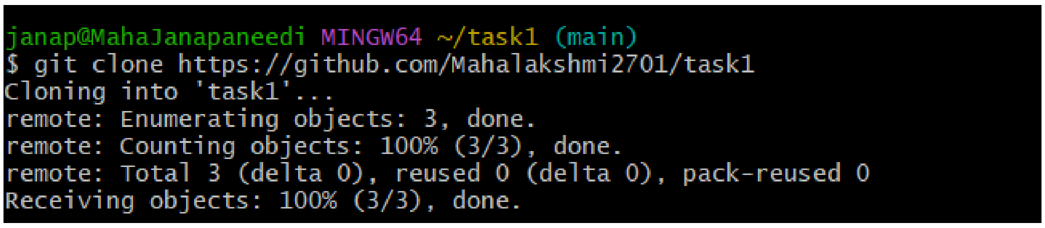
Pull, however, will not only download the changes but also merges them - it is a combination of fetch and merges. The configured remote-tracking branch is selected automatically.
Git pull=git fetch + git merge

Step 1)Git clone an empty repository

First, give the username and email using the git config. Then create a new repository named task1 and clone using the git clone command.

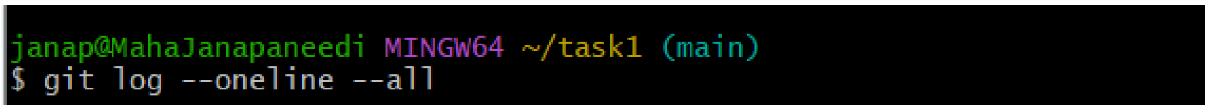


```
janap@MahaJanapaneedi MINGW64 ~ (master)
$ git config --global user.name "Mahalakshmi2701"

janap@MahaJanapaneedi MINGW64 ~ (master)
$ git config --global user.email "20A91A05F3@aec.edu.in"
```



```
janap@MahaJanapaneedi MINGW64 ~/task1 (main)
$ git clone https://github.com/Mahalakshmi2701/task1
Cloning into 'task1'...
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.
```

Step 2) Git log –oneline –all

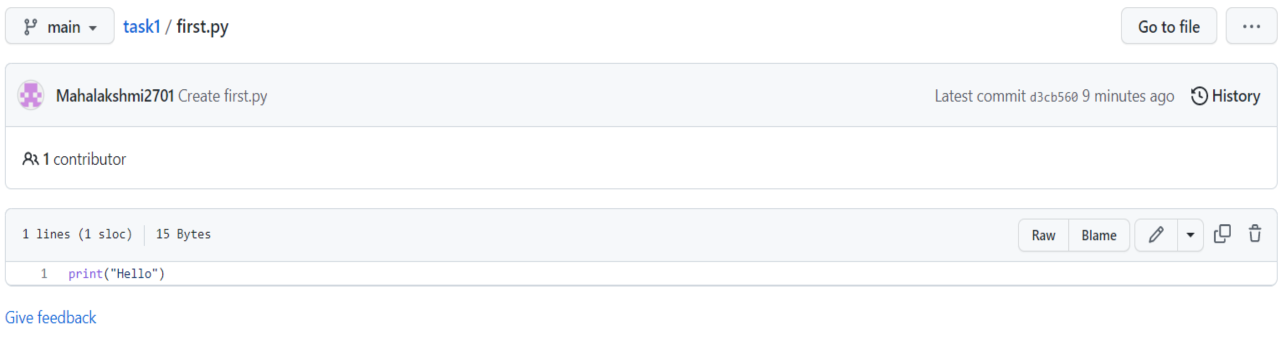


```
janap@MahaJanapaneedi MINGW64 ~/task1 (main)
$ git log --oneline --all
```

Step 3)Go to the repository and create a file.

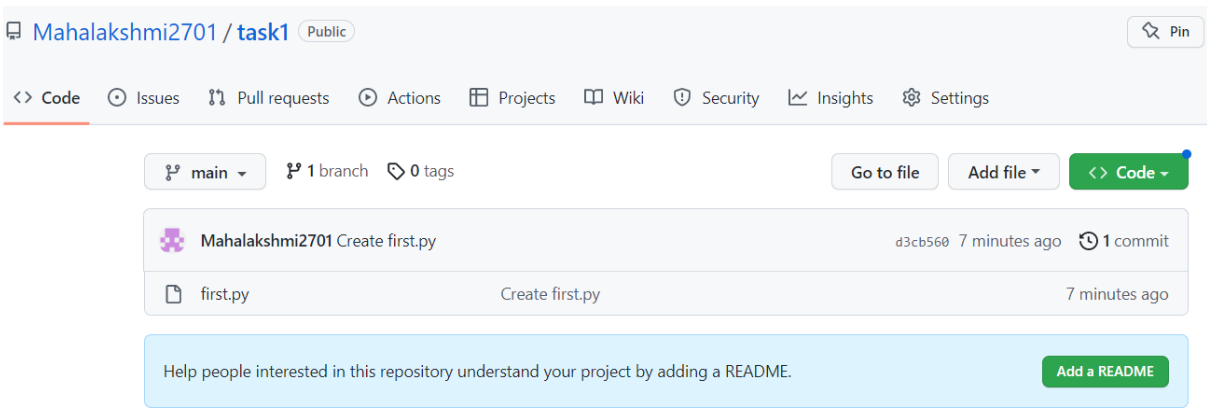
This command gives no output as we have not committed anything yet.

Go to GitHub and go to the repository task1 and create a file and then update it.



main
task1 / first.py
Go to file
Mahalakshmi2701 Create first.py
Latest commit d3cb560 9 minutes ago
History
1 contributor
1 lines (1 sloc) | 15 Bytes
Raw
Blame
Give feedback

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

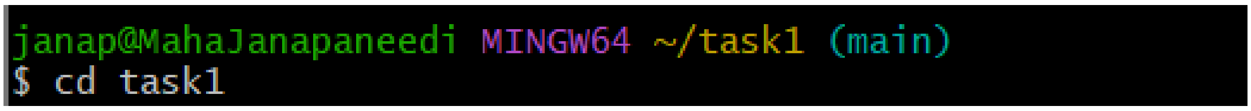


Mahalakshmi2701 Create first.py d3cb560 7 minutes ago 1 commit

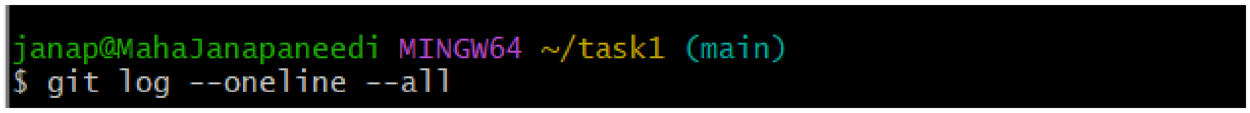
first.py Create first.py 7 minutes ago

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

Step 4)In the git bash, go to the task1 repository.



```
janap@MahaJanapaneedi MINGW64 ~/task1 (main)
$ cd task1
```

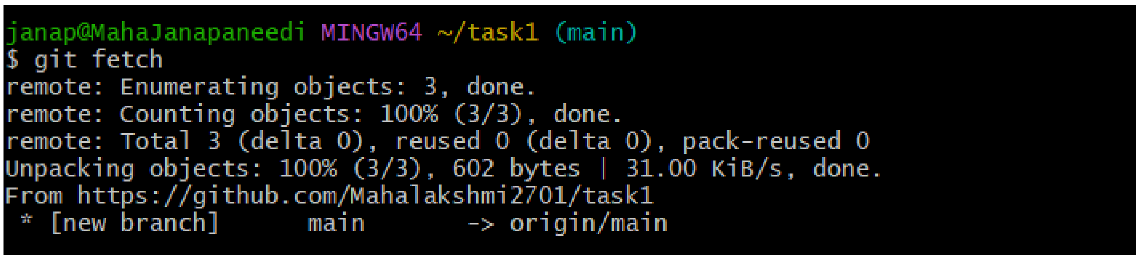


```
janap@MahaJanapaneedi MINGW64 ~/task1 (main)
$ git log --oneline --all
```

We have moved to the task1 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 5)Git fetch

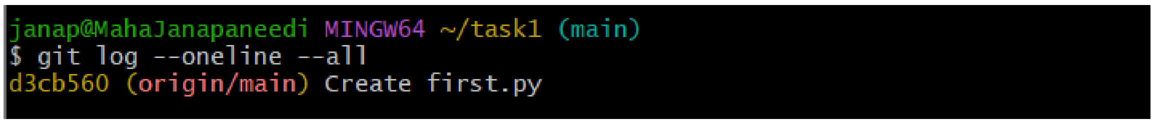
Git fetch will download the changes in the remote repository, fetch commands just downloads the changes made in the remote repository.



```
janap@MahaJanapaneedi MINGW64 ~/task1 (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), 602 bytes | 31.00 KiB/s, done.
From https://github.com/Mahalakshmi2701/task1
 * [new branch]      main       -> origin/main
```

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

Git log –oneline –all

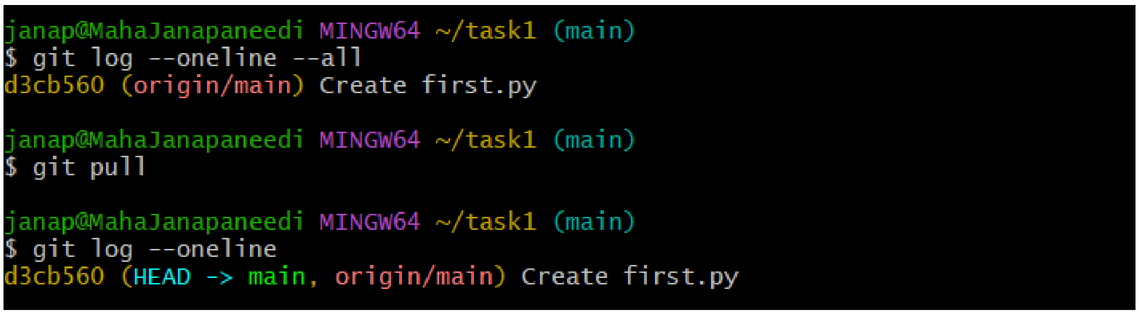


```
janap@MahaJanapaneedi MINGW64 ~/task1 (main)
$ git log --oneline --all
d3cb560 (origin/main) Create first.py
```

Here when the 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.

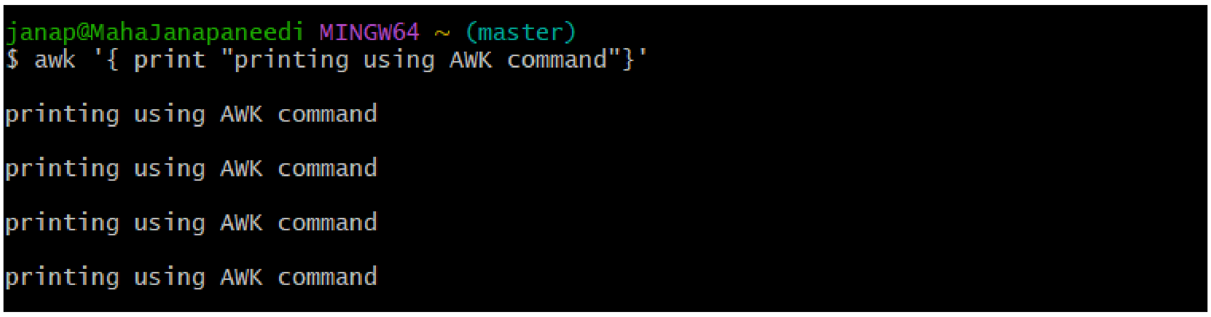
Git pull------downloads and merges


```
janap@MahaJanapaneedi MINGW64 ~/task1 (main)
$ git log --oneline --all
d3cb560 (origin/main) Create first.py

janap@MahaJanapaneedi MINGW64 ~/task1 (main)
$ git pull

janap@MahaJanapaneedi MINGW64 ~/task1 (main)
$ git log --oneline
d3cb560 (HEAD -> main, origin/main) Create first.py
```

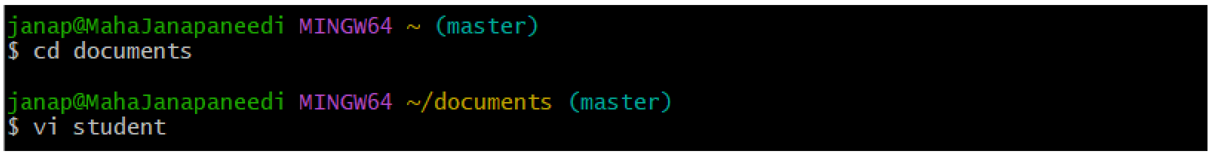
Now the commits are applied to the main branch by using the git pull command.

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.

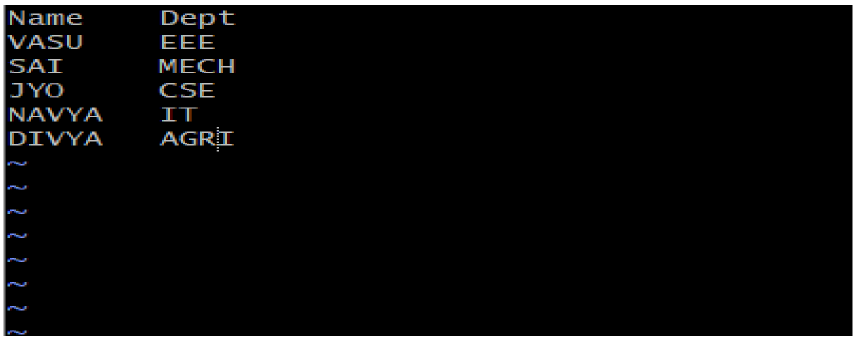
AWK:
The Awk is a powerful scripting language used for **text scripting**. It searches and replaces the texts and sorts, validates, and indexes the database. It performs various actions on a file like searching a specified text and more.



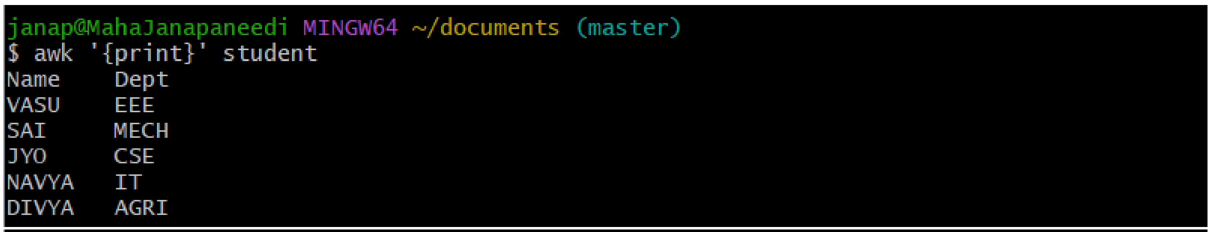
```
janap@MahaJanapaneedi MINGW64 ~ (master)
$ awk '{ print "printing using AWK command"}'

printing using AWK command

printing using AWK command

printing using AWK command

printing using AWK command
```



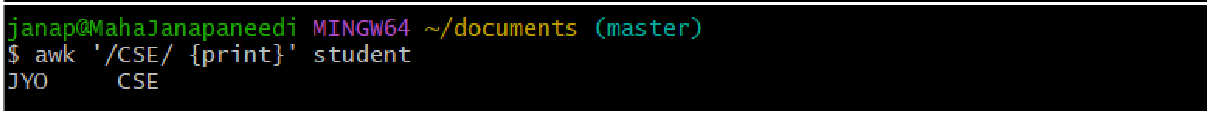
```
janap@MahaJanapaneedi MINGW64 ~ (master)
$ cd documents

janap@MahaJanapaneedi MINGW64 ~/documents (master)
$ vi student
```



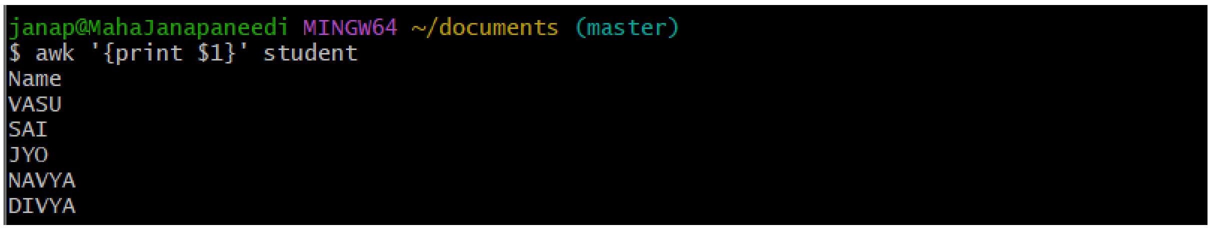
```
Name      Dept
VASU      EEE
SAI       MECH
JYO       CSE
NAVYA     IT
DIVYA     AGRI
~
~
~
~
~
~
~
~
```



```
janap@MahaJanapaneedi MINGW64 ~/documents (master)
$ awk '{print}' student
Name     Dept
VASU     EEE
SAI      MECH
JYO      CSE
NAVYA    IT
DIVYA    AGRI
```



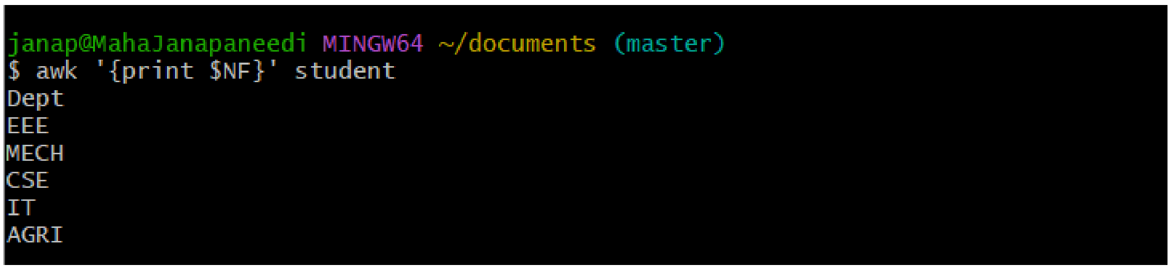
```
janap@MahaJanapaneedi MINGW64 ~/documents (master)
$ awk '/CSE/ {print}' student
JYO      CSE
```



```
janap@MahaJanapaneedi MINGW64 ~/documents (master)
$ awk '{print $1}' student
Name
VASU
SAI
JYO
NAVYA
DIVYA
```

- above command will print column1

- *$NF: Used to display the last field of the file.



```
janap@MahaJanapaneedi MINGW64 ~/documents (master)
$ awk '{print $NF}' student
Dept
EEE
MECH
CSE
IT
AGRI
```

Steps to follow bash scripting:

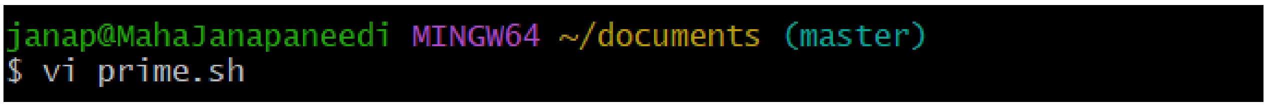
Step 1) create the file with the extension .sh.

Step 2)open the shell and write the script.

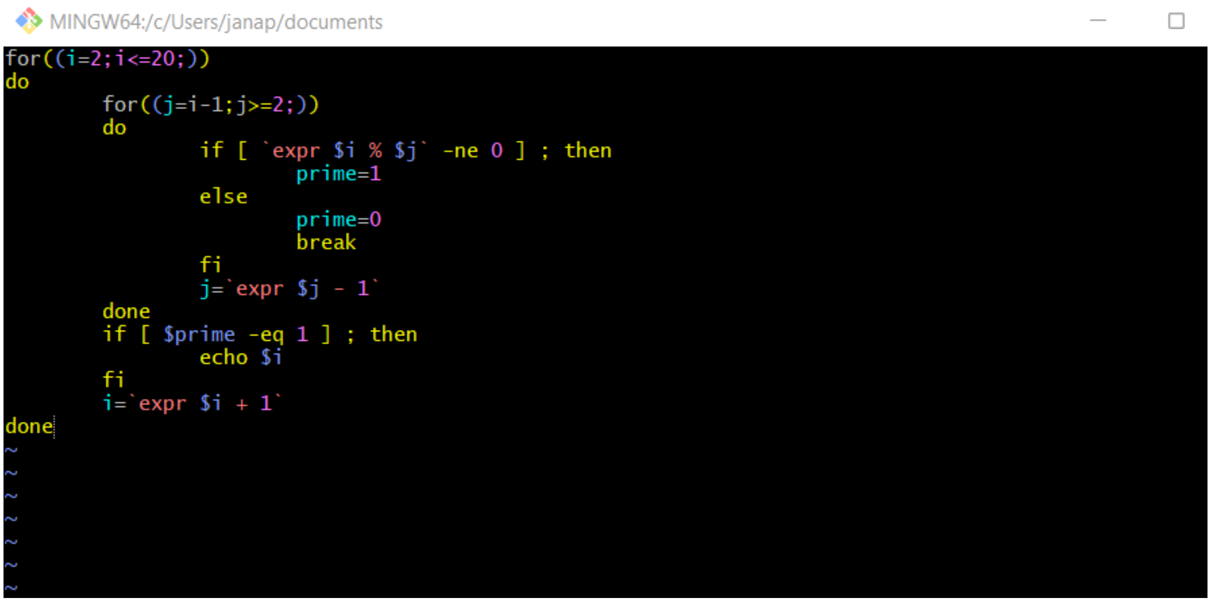
Step 3)save the code and run the code.

To run the run a code

Syntax: bash filename. sh

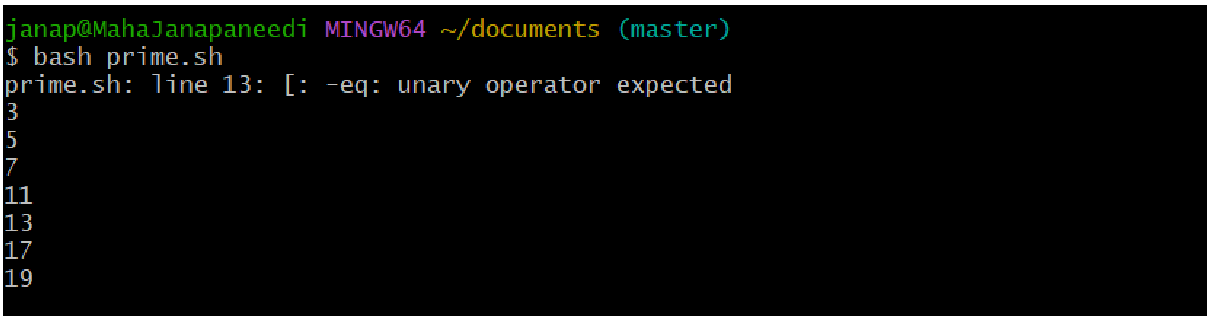


```
janap@MahaJanapaneedi MINGW64 ~/documents (master)
$ vi prime.sh
```



MINGW64:/c/Users/janap/documents

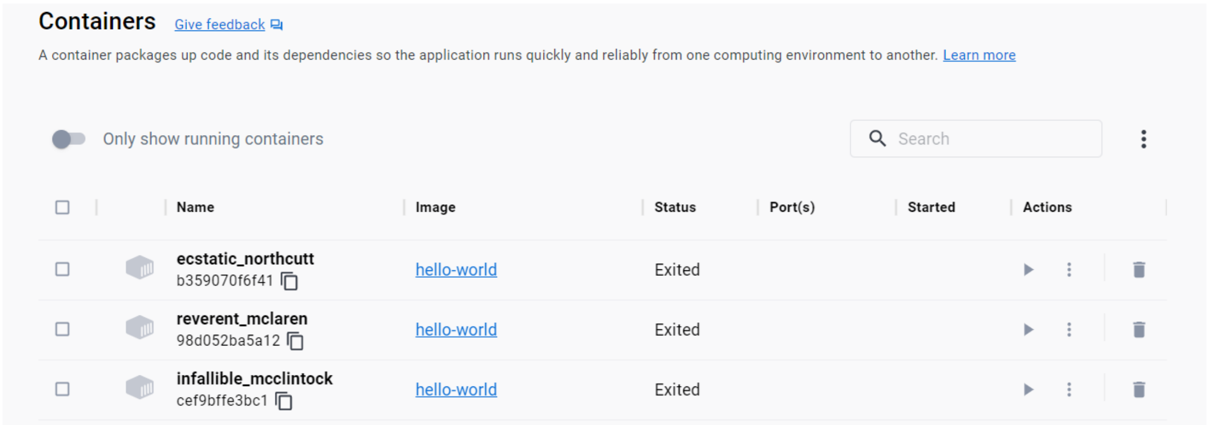
```
for((i=2;i<=20;))
do
        for((j=i-1;j>=2;))
        do
                if [ `expr $i % $j` -ne 0 ] ; then
                        prime=1
                else
                        prime=0
                        break
                fi
                j=`expr $j - 1`
        done
        if [ $prime -eq 1 ] ; then
                echo $i
        fi
        i=`expr $i + 1`
done
~
~
~
~
~
~
~
```



```
janap@MahaJanapaneedi MINGW64 ~/documents (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.

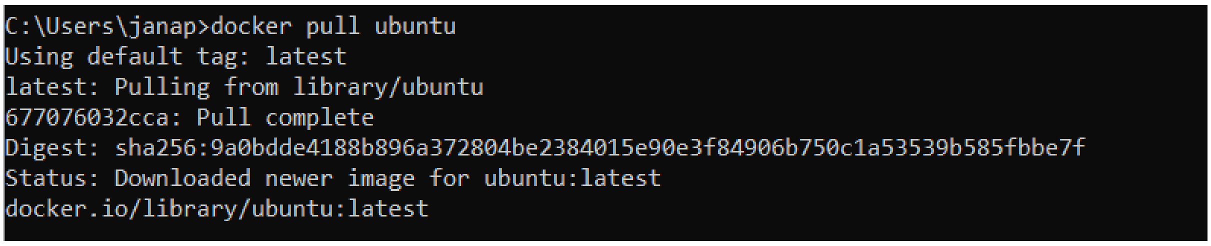
Image: Images are used to create containers. It uses a private container registry to share container images within the enterprise and also uses a public container registry to share container images with the whole world.
Container: Containers are used to hold the entire package that is needed to run the 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.


Containers Give feedback

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Only show running containers

Name	Image	Status	Port(s)	Started	Actions
ecstatic_northcutt b359070f6f41	hello-world	Exited			
reverent_mclaren 98d052ba5a12	hello-world	Exited			
infallible_mcclintock cef9bffe3bc1	hello-world	Exited			


```
C:\Users\janap>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
```

*For setting up a container and run the ubuntu os,

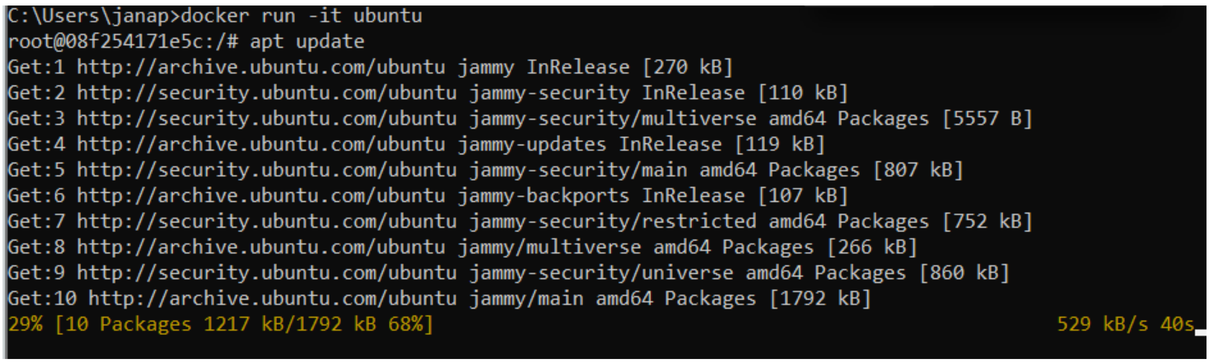
→First we need to download the image of Ubuntu from the docker hub using the command docker pull **ubuntu**.

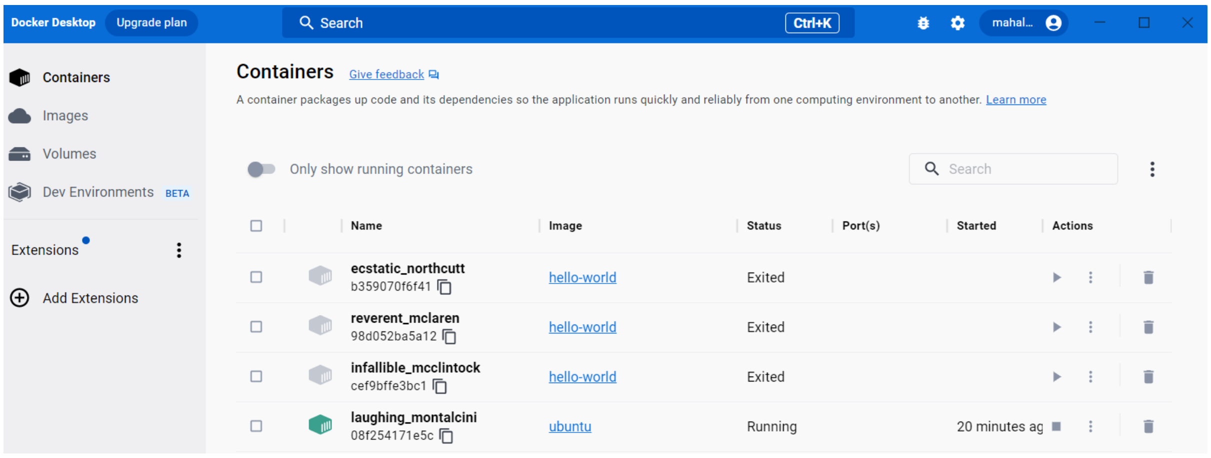
→To create a container and execute the image use the command docker run -it ubuntu .

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

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

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


```
C:\Users\janap>docker run -it ubuntu
root@08f254171e5c:/# 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/multiverse amd64 Packages [5557 B]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [807 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [752 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [860 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
29% [10 Packages 1217 kB/1792 kB 68%]                                    529 kB/s 40s
```

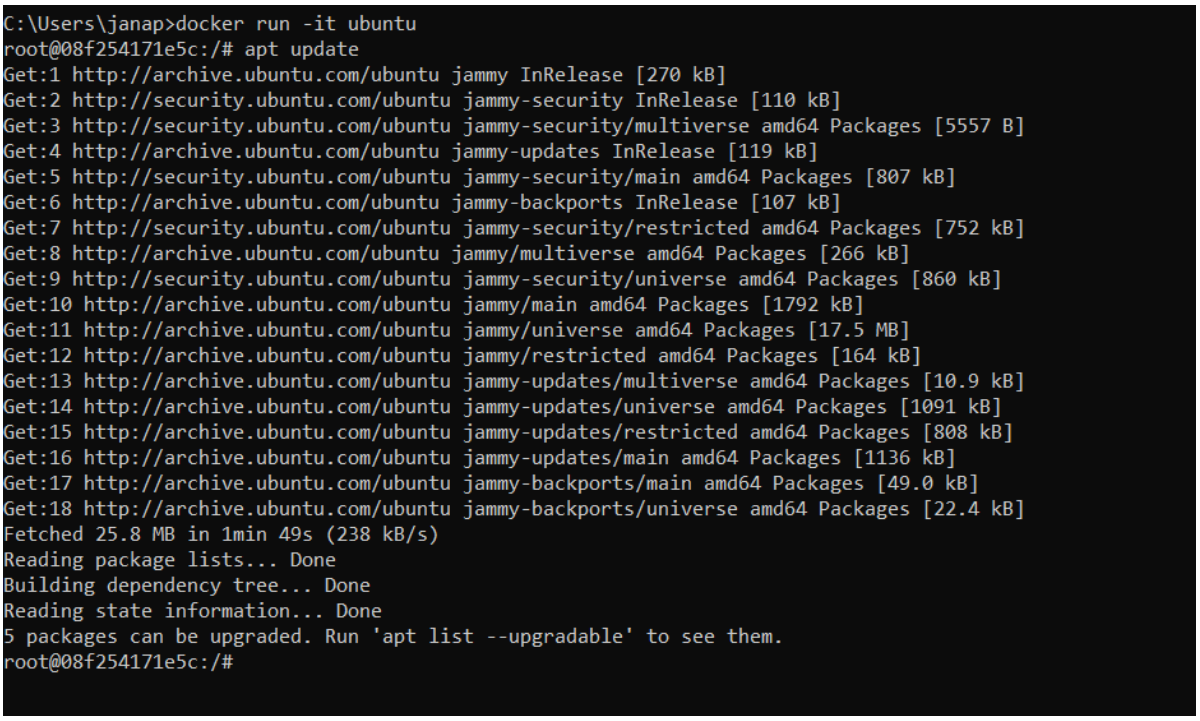


Containers

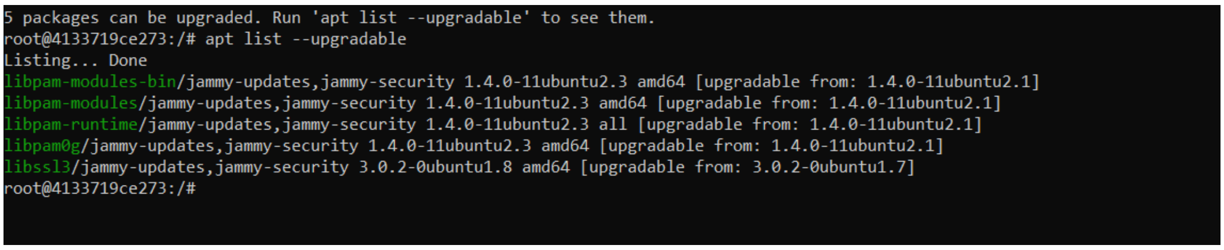
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Only show running containers

Name	Image	Status	Port(s)	Started	Actions
ecstatic_northcutt b359070f6f41	hello-world	Exited			
reverent_mclaren 98d052ba5a12	hello-world	Exited			
infallible_mcclintock cef9bffe3bc1	hello-world	Exited			
laughing_montalcini 08f254171e5c	ubuntu	Running		20 minutes ag	



```
C:\Users\janap>docker run -it ubuntu
root@08f254171e5c:/# 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/multiverse amd64 Packages [5557 B]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [807 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [752 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [860 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [10.9 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1091 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [808 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1136 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 1min 49s (238 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@08f254171e5c:/#
```



```
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@4133719ce273:/# 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@4133719ce273:/#
```

**Now a container got created for the ubuntu image.