

NAME: AAKASH NAMALA

ROLL NO: 20A91A0544

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git config user.name
aakash-namala

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git config user.email
n.aakash345@gmail.com

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ pwd
/d/hero_asm1

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ █
```

### Question 1:

**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.**

The git stash command is used to save changes temporarily that are not yet ready to be committed or are not yet ready to be pushed to a remote repository. It allows you to "stash" changes in a separate branch, so you can switch to another branch, make changes, and then come back to your original branch and restore your stashed changes. The git stash command enables you to switch branches without committing the current branch.

Let us initialize a repository. Create a new file "file1.txt", add the content, add it to staging area and then commit it.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat >file1.txt
This is a demo file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ ls
file1.txt

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git status
On branch master

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)

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git add .
warning: LF will be replaced by CRLF in file1.txt.
The file will have its original line endings in your working directory

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git status
On branch master

No commits yet

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

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git commit -m "first commit"
[master (root-commit) 4c8302e] first commit
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
```

Now made the changes to the file1.txt and show the status.

```
create mode 100644 file1.txt

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat >>file1.txt
This is to modify the file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file1.txt
This is a demo file.
This is to modify the file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (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:   file1.txt

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

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

## git stash

Use the **git stash** command to stash the changes.

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

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash
warning: LF will be replaced by CRLF in file1.txt.
The file will have its original line endings in your working directory
Saved working directory and index state WIP on master: 4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git status
On branch master
nothing to commit, working tree clean

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

## git stash list

It is used to check the stored stashes.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

Start

## git stash show

To track the stashes and their changes.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash show
file1.txt | 1 +
1 file changed, 1 insertion(+)

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

After stashing the changes, the file content will be as follows

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

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash
warning: LF will be replaced by CRLF in file1.txt.
The file will have its original line endings in your working directory
Saved working directory and index state WIP on master: 4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git status
On branch master
nothing to commit, working tree clean

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash show
file1.txt | 1 +
1 file changed, 1 insertion(+)

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file1.txt
This is a demo file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

## git stash apply

git stash apply command is used to re-apply the changes that we have stashed.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (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:   file1.txt

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

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file1.txt
This is a demo file.
This is to modify the file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
```

git

## stash pop

This command will apply re-apply the changes which are on the top of the stash stack and also delete the stash from the stash list.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat >>file1.txt
This is to modify the file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (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:   file1.txt

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

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash
warning: LF will be replaced by CRLF in file1.txt.
The file will have its original line endings in your working directory
Saved working directory and index state WIP on master: 4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (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:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (843bd0db93327432ad8f3650b6db8f7bec7b0b50)

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
```

## git stash clear

This command is used to delete the all-available stashes at once.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 4c8302e first commit
stash@{1}: WIP on master: 4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash clear

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ gi stash list
bash: gi: command not found

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

## git stash drop

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

[master 7d3090d] second commit
2 files changed, 2 insertions(+)
create mode 100644 file.txt
create mode 100644 file2.txt

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file1.txt
This is a demo file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file2.txt
This is the second file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat >>file1.txt
This is the first modification.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash
warning: LF will be replaced by CRLF in file1.txt.
The file will have its original line endings in your working directory
Saved working directory and index state WIP on master: 7d3090d second commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 7d3090d second commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat >>file2.txt
This is the second modification.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash
warning: LF will be replaced by CRLF in file2.txt.
The file will have its original line endings in your working directory
Saved working directory and index state WIP on master: 7d3090d second commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 7d3090d second commit
stash@{1}: WIP on master: 7d3090d second commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 7d3090d second commit
stash@{1}: WIP on master: 7d3090d second commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash drop
Dropped refs/stash@{0} (7210cbf999f04a3bd1a7c90cd771130783afbdbc)

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: WIP on master: 7d3090d second commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

When we apply or pop the stash. The changes will be added only to the file1.txt, because we have dropped the changes related to file2.txt.

```
$ git stash list

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file1.txt
This is a demo file.
This is the first modification.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file2.txt
This is the second file.
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

### git stash save

If we want to push the changes into the stash with a message, then we can use the git stash save command.

Syntax: git stash save "message"

```
This is the second file.
Aakash007@AAKASH MINGW64 /d/hero_asm1 (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:   file1.txt

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

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash save "Pushing the changes into the stash"
Saved working directory and index state On master: Pushing the changes into the stash

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash list
stash@{0}: On master: Pushing the changes into the stash

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
```

## git stash branch

By using the git stash branch we can create a new branch in which all the stash will be reflected in that newly created branch.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

4c8302e first commit

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ ls
file1.txt  file2.txt

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file1.txt
This is a demo file.
This is change.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file2.txt
This is the second file.
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat >>file2.txt
This is a modified file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat file2.txt
This is the second file.This is a modified file.

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash
warning: LF will be replaced by CRLF in file2.txt.
The file will have its original line endings in your working directory
Saved working directory and index state WIP on master: dc1171d commit1

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ git stash branch newbranch
Switched to a new branch 'newbranch'
Removing file.txt
On branch newbranch
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    file.txt
        modified:   file2.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (ed5c9463866e5506e8770301c982c14553bdc704)

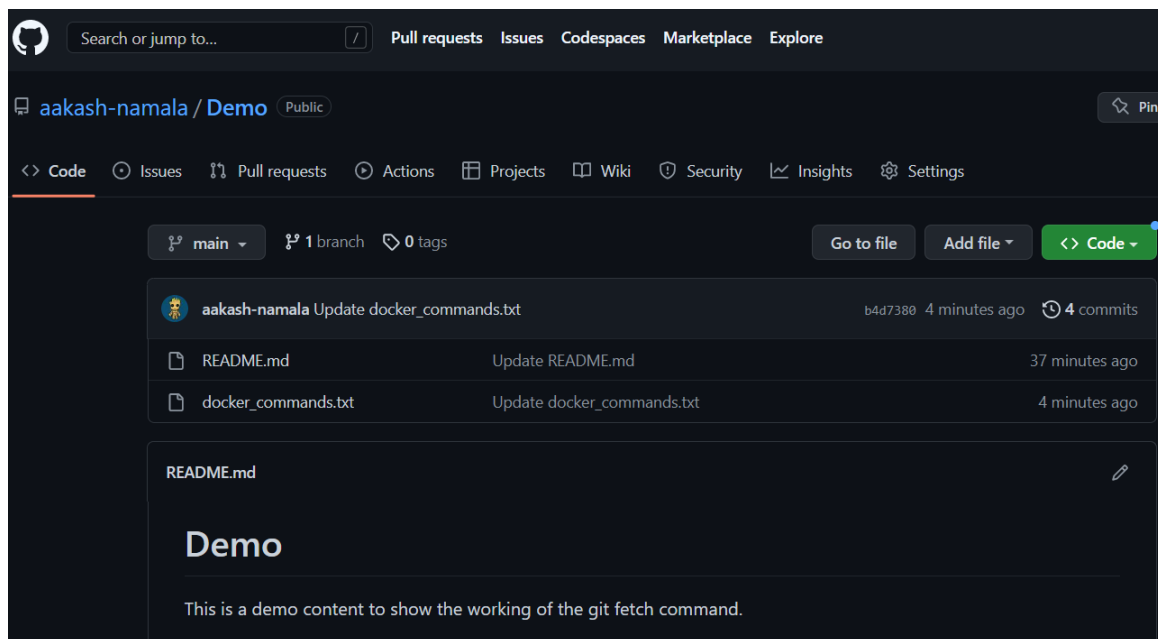
Aakash007@AAKASH MINGW64 /d/hero_asm1 (newbranch)
$ █
```

## Question 2:

**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 is a Git command that allows you to download changes from a remote repository into your local repository. The command fetches all the changes that have been made in the remote repository since the last time you fetched or cloned it. git fetch only downloads the changes and does not merge them with your local branch. This means that you can review the changes before deciding whether to merge them or not.

This is the Demo repository, in which we have 2 files.



Let us take a new and clean local repository for this example and clone the Remote Demo repository into it using the clone command.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cd Question2

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2 (master)
$ ls

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2 (master)
$ git clone https://github.com/aakash-namala/Demo.git
Cloning into 'Demo'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 12 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (12/12), done.
Resolving deltas: 100% (1/1), done.

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2 (master)
```



After cloning move to the Demo directory. We can see the 2 files in it. As we are moving from master to sub directory the branch changes to main by default.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2 (master)
$ ls
Demo/

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2 (master)
$ cd Demo

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2/Demo (main)
$ ls
docker_commands.txt  README.md

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2/Demo (main)
$
```

Open the docker\_commands.txt file and see it.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2/Demo (main)
$ cat docker_commands.txt
DOCKER BASIC COMMANDS
=> docker version

=> docker run hello-world => it will run the hello-world, the docker daemon will search for the image in the local machine, if it does not find it,
then it(it acts as an hypervisor b/w client and docker hub) will reach the docker hub and pull the image from the docker hub
to the local machine, then it run the image.

=> docker search MySQL => it searches for the image and gives the description about the image.
=> docker pull image_name => if we want to download the image and we dont want to run it, it will download the image an it show us the status.

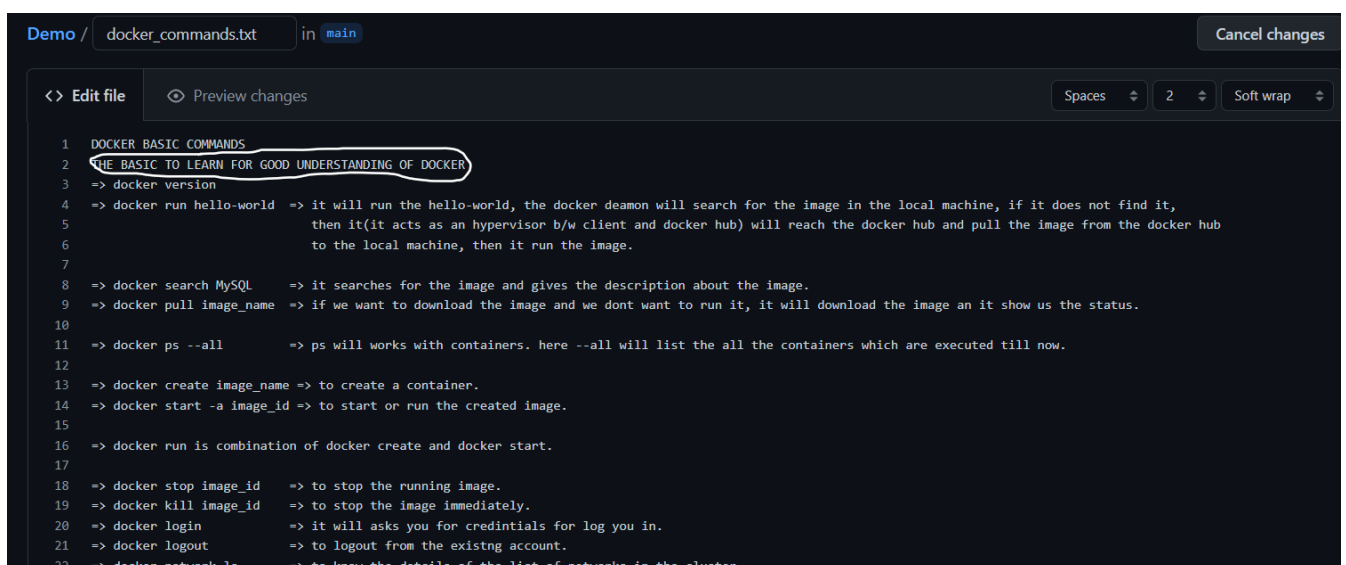
=> docker ps --all => ps will works with containers. here --all will list the all the containers which are executed till now.

=> docker create image_name => to create a container.
=> docker start -a image_id => to start or run the created image.

=> docker run is combination of docker create and docker start.

=> docker stop image_id => to stop the running image.
=> docker kill image_id => to stop the image immediately.
=> docker login => it will asks you for credintials for log you in.
=> docker logout => to logout from the existng account.
=> docker network ls => to know the details of the list of networks in the cluster.
=> docker rmi => is a tagged command to remove the image.
=> docker rm => untagged command to remove the image. using rm we can remove one or more images.
```

Now open the GitHub and made some changes to the remote repository.



```
Demo / docker_commands.txt in main Cancel changes

<> Edit file Preview changes Spaces 2 Soft wrap

1 DOCKER BASIC COMMANDS
2 THE BASIC TO LEARN FOR GOOD UNDERSTANDING OF DOCKER
3 => docker version
4 => docker run hello-world => it will run the hello-world, the docker daemon will search for the image in the local machine, if it does not find it,
5 then it(it acts as an hypervisor b/w client and docker hub) will reach the docker hub and pull the image from the docker hub
6 to the local machine, then it run the image.
7
8 => docker search MySQL => it searches for the image and gives the description about the image.
9 => docker pull image_name => if we want to download the image and we dont want to run it, it will download the image an it show us the status.
10
11 => docker ps --all => ps will works with containers. here --all will list the all the containers which are executed till now.
12
13 => docker create image_name => to create a container.
14 => docker start -a image_id => to start or run the created image.
15
16 => docker run is combination of docker create and docker start.
17
18 => docker stop image_id => to stop the running image.
19 => docker kill image_id => to stop the image immediately.
20 => docker login => it will asks you for credintials for log you in.
21 => docker logout => to logout from the existng account.
22 => docker network ls => to know the details of the list of networks in the cluster.
```

Now fetch the changes using the fetch command. Even after fetching, it won't reflect in our local repository until we merge them.

```

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2/Demo (main)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 757 bytes | 1024 bytes/s, done.
From https://github.com/aakash-namala/Demo
   b4d7380..587aa3d  main    -> origin/main

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2/Demo (main)
$ cat docker_commands.txt
DOCKER BASIC COMMANDS
=> docker version
=> docker run hello-world => it will run the hello-world, the docker daemon will search for the image in the local machine, if it does not find it,
                           then it(it acts as an hypervisor b/w client and docker hub) will reach the docker hub and pull the image from the docker hub
                           to the local machine, then it run the image.

=> docker search MySQL    => it searches for the image and gives the description about the image.
=> docker pull image_name => if we want to download the image and we dont want to run it, it will download the image an it show us the status.

=> docker ps --all        => ps will works with containers. here --all will list the all the containers which are executed till now.

```

Now use the merge Command to merge the changes to the main branch. After merging we can observe the changes.

```

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2/Demo (main)
$ git merge
Updating b4d7380..587aa3d
Fast-forward
 docker_commands.txt | 1 +
 1 file changed, 1 insertion(+)

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question2/Demo (main)
$ cat docker_commands.txt
DOCKER BASIC COMMANDS
THE BASIC TO LEARN FOR GOOD UNDERSTANDING OF DOCKER
=> docker version
=> docker run hello-world => it will run the hello-world, the docker daemon will search for the image in the local machine, if it does not find it,
                           then it(it acts as an hypervisor b/w client and docker hub) will reach the docker hub and pull the image from the docker hub
                           to the local machine, then it run the image.

=> docker search MySQL    => it searches for the image and gives the description about the image.
=> docker pull image name => if we want to download the image and we dont want to run it, it will download the image an it show us the status.

```

### Question 3:

**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** downloads the latest changes from a remote repository but does not apply them to your local branch. It simply updates the tracking branches in your local repository to reflect the state of the remote repository. You can then review the changes and decide whether or not to merge them into your local branch.

**git pull** also downloads the latest changes from a remote repository, but it automatically merges those changes into your local branch.

Clone the Remote Demo repository into it using the clone command. After cloning move to the Demo directory. We can see the 2 files in it. As we are moving from master to sub directory the branch changes to main by default.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cd Question3

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3 (master)
$ git clone https://github.com/aakash-namala/Demo.git
Cloning into 'Demo'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 15 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (15/15), done.
Resolving deltas: 100% (2/2), done.

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3 (master)
$

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3 (master)
$ ls
Demo/

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3 (master)
$ cd Demo

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3/Demo (main)
$ ls
docker_commands.txt  README.md

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3/Demo (main)
$
```

Open the docker\_commands.txt file and view it.

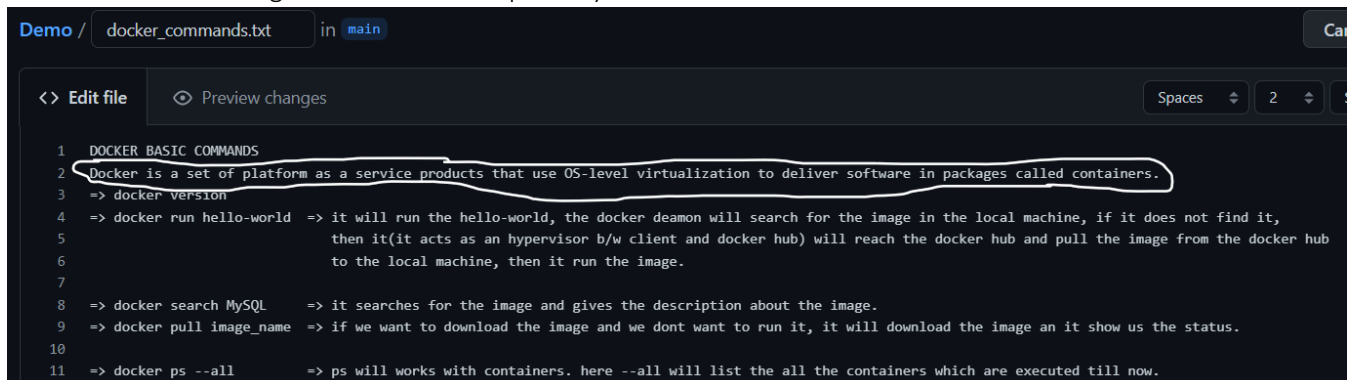
```
Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3/Demo (main)
$ ls
docker_commands.txt  README.md

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3/Demo (main)
$ cat docker_commands.txt
DOCKER BASIC COMMANDS
THE BASIC TO LEARN FOR GOOD UNDERSTANDING OF DOCKER
=> docker version
=> docker run hello-world => it will run the hello-world, the docker daemon will
                           then it(it acts as an hypervisor b/w client and do
                           to the local machine, then it run the image.

=> docker search MySQL    => it searches for the image and gives the descriptio
=> docker pull image_name => if we want to download the image and we dont want

=> docker ps --all        => ps will works with containers. here --all will lis
> docker create image_name -> to create a container
```

Now made some changes in the remote repository.



```
Demo / docker_commands.txt in main
<> Edit file Preview changes Spaces 2
1 DOCKER BASIC COMMANDS
2 Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.
3 => docker version
4 => docker run hello-world => it will run the hello-world, the docker daemon will search for the image in the local machine, if it does not find it,
5 then it(it acts as an hypervisor b/w client and docker hub) will reach the docker hub and pull the image from the docker hub
6 to the local machine, then it run the image.
7
8 => docker search MySQL => it searches for the image and gives the description about the image.
9 => docker pull image_name => if we want to download the image and we dont want to run it, it will download the image an it show us the status.
10
11 => docker ps --all => ps will works with containers. here --all will list the all the containers which are executed till now.
```

Now use the fetch command. If we fetch, then the changes will be downloaded but the changes will not get merged.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3/Demo (main)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 821 bytes | 4.00 KiB/s, done.
From https://github.com/aakash-namala/Demo
 587aa3d..9bd00bb main -> origin/main

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3/Demo (main)
$ cat docker_commands.txt
DOCKER BASIC COMMANDS
THE BASIC TO LEARN FOR GOOD UNDERSTANDING OF DOCKER
=> docker version
=> docker run hello-world => it will run the hello-world, the docker daemon will search for the
                           then it(it acts as an hypervisor b/w client and docker hub) will r
                           to the local machine, then it run the image.

=> docker search MySQL    => it searches for the image and gives the description about the imag
=> docker pull image_name => if we want to download the image and we dont want to run it, it wi
```

Now use the git pull command. It will download the changes and merge the changes automatically into the local repository.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3/Demo (main)
$ git pull
Updating 587aa3d..9bd00bb
Fast-forward
 docker_commands.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Aakash007@AAKASH MINGW64 /d/hero_asm1/Question3/Demo (main)
$ cat docker_commands.txt
DOCKER BASIC COMMANDS
Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.
=> docker version
=> docker run hello-world => it will run the hello-world, the docker daemon will search for the image in the local machine, if it does not find
                           then it(it acts as an hypervisor b/w client and docker hub) will reach the docker hub and pull the image from the d
                           to the local machine, then it run the image.

=> docker search MySQL    => it searches for the image and gives the description about the image.
=> docker pull image_name => if we want to download the image and we dont want to run it, it will download the image an it show us the status.

=> docker ps --all        => ps will works with containers. here --all will list the all the containers which are executed till now.
```

#### Question 4:

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` command:

"`awk`" is a command-line tool used for manipulating and processing text files, usually in the form of structured data. It is particularly useful for working with files that have a consistent structure, such as log files, CSV files, or tabular data.

**Syntax:** `awk options 'selection_criteria {action }' input-file > output-file`

Create a demo file as `student.txt` and add the content to it.

Now run the command => `awk -F ',' '{print $1, $2}' student.txt`

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ cat student.txt
Name, Marks, Grade
Aakash, 96, A
Sudarshan, 97, A+
Satya, 98, A++
Rayudu, 89, B+

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ awk -F ',' '{print $1, $2}' student.txt
Name Marks
Aakash 96
Sudarshan 97
Satya 98
Rayudu 89

Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

`-F` or `--field-separator`:

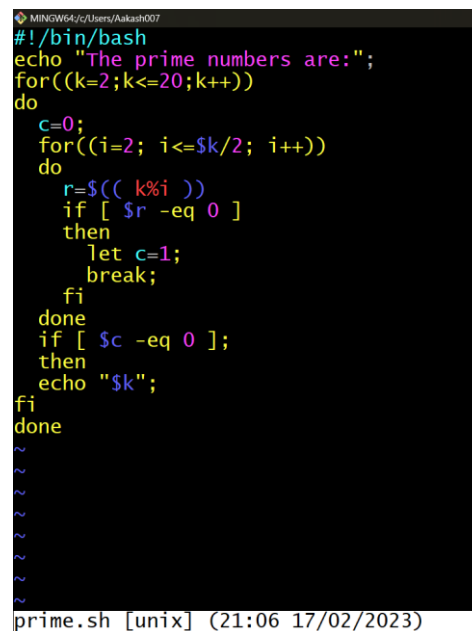
specifies the field separator for splitting the input. By default, `awk` splits the input by whitespace.

```
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$ awk -F ',' '{print $1, $2, $3}' student.txt
Name Marks Grade
Aakash 96 A
Sudarshan 97 A+
Satya 98 A++
Rayudu 89 B+

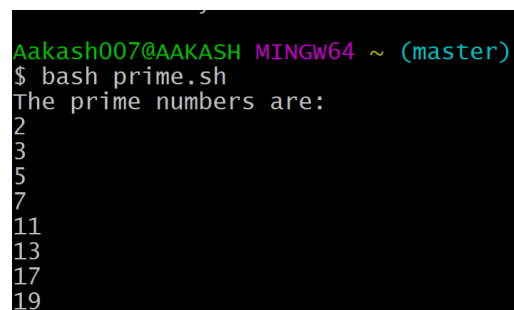
Aakash007@AAKASH MINGW64 /d/hero_asm1 (master)
$
```

## Bash script file to find out the prime number from the range 1 to 20.

```
#!/bin/bash
echo "The prime numbers are:";
for((k=2;k<=20;k++))
do
    c=0;
    for((i=2; i<=$k/2; i++))
    do
        r=$(( k%i ))
        if [ $r -eq 0 ]
        then
            let c=1;
            break;
        fi
    done
    if [ $c -eq 0 ];
    then
        echo "$k";
    fi
done
```



```
MINGW64/c/Users/Aakash007
#!/bin/bash
echo "The prime numbers are:";
for((k=2;k<=20;k++))
do
    c=0;
    for((i=2; i<=$k/2; i++))
    do
        r=$(( k%i ))
        if [ $r -eq 0 ]
        then
            let c=1;
            break;
        fi
    done
    if [ $c -eq 0 ];
    then
        echo "$k";
    fi
done
~
~
~
~
~
~
~
prime.sh [unix] (21:06 17/02/2023)
```



```
Aakash007@AAKASH MINGW64 ~ (master)
$ bash prime.sh
The prime numbers are:
2
3
5
7
11
13
17
19
```



```
Aakash007@AAKASH MINGW64 ~ (master)
$ history 10
319 clear
320 vi prime.sh
321 bash prime.sh
322 vi prime.sh
323 vi prime.sh
324 bash prime.sh
325 vi prime.sh
326 bash prime.sh
327 history
328 history 10
Aakash007@AAKASH MINGW64 ~ (master)
$
```

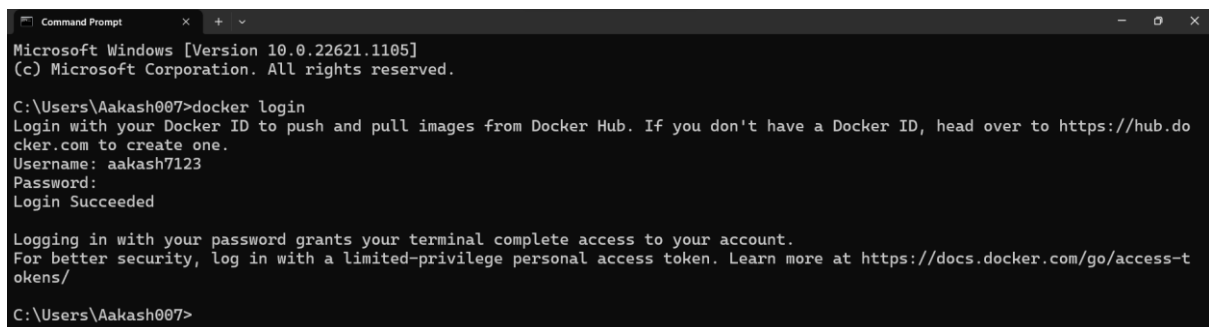
### Question 5:

**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.**

Firstly, login to your docker account using the docker login command.

Syntax: docker login

It will asks credentials to enter.



```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

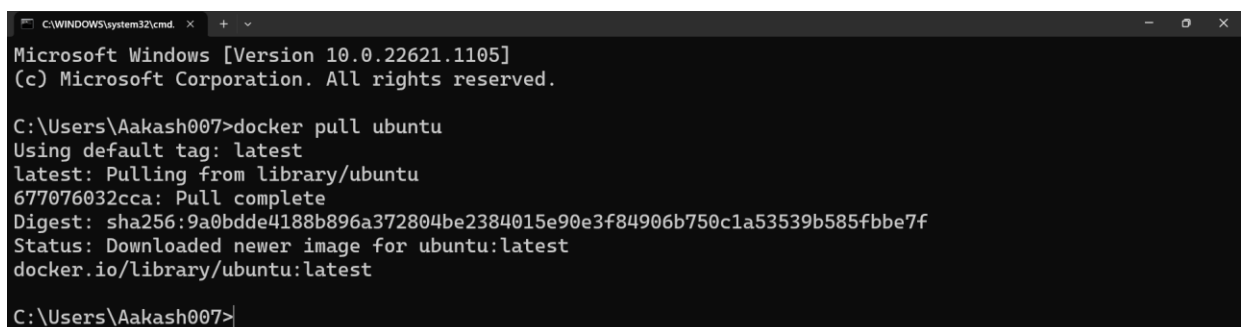
C:\Users\Aakash007>docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: aakash7123
Password:
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/

C:\Users\Aakash007>
```

Now we need to pull the ubuntu from the Docker Hub. For this we can use the docker pull command.

Syntax: docker pull image\_name



```
C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

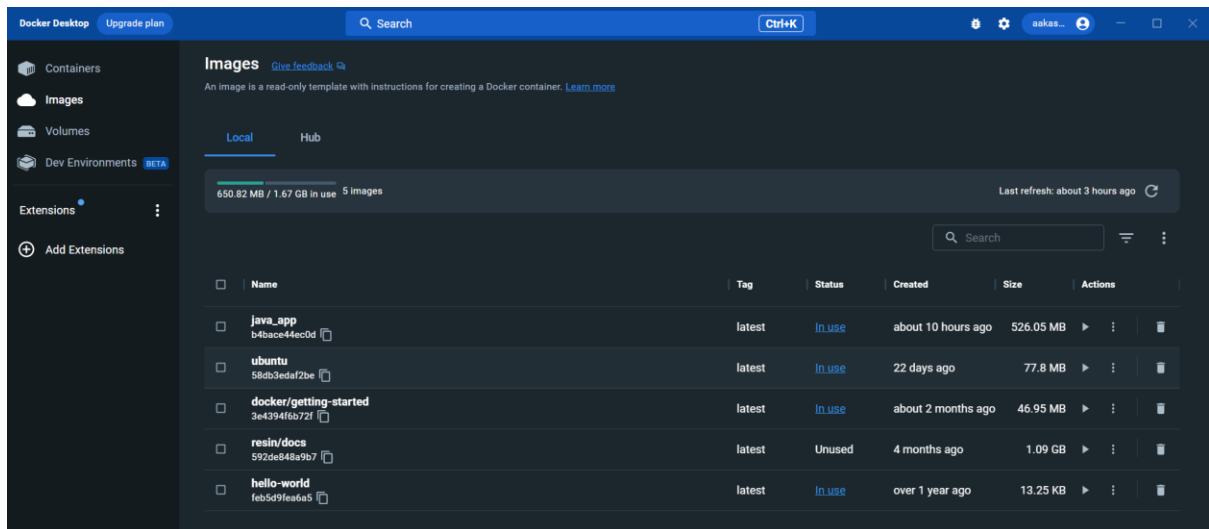
C:\Users\Aakash007>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
677076032cca: Pull complete
Digest: sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbb7f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

C:\Users\Aakash007>
```

Now ubuntu is successfully downloaded into the docker.

We can check this in the GUI interface.





Now run the ubuntu by using the run command. To run this, we can use the docker run command.

To run in the interactive mode use -it (or) -i -t

Now it will start a new container with Ubuntu OS in interactive mode. The -it option stands for "interactive" and "tty", which means that you can interact with the container through a command prompt.

```
C:\Users\Aakash007>docker run -it ubuntu
root@1c2f9155fcff:/#
root@1c2f9155fcff:/#
root@1c2f9155fcff:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@1c2f9155fcff:/# ls -a
. .. .dockerenv bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp
usr var
root@1c2f9155fcff:/# |
```

Now the ubuntu is running in the interactive mode, try checking it with some Linux commands as

ls, ls -a