

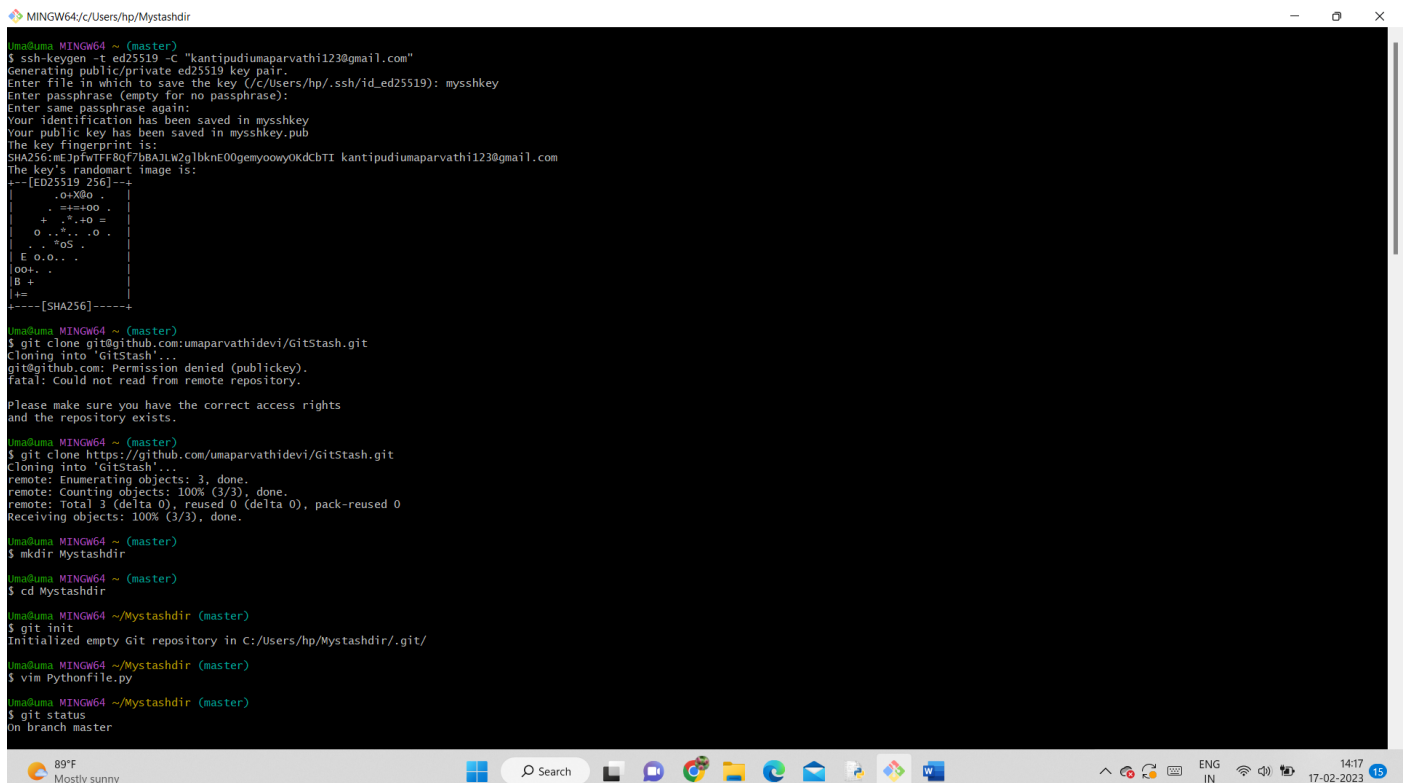
Q1. Describe the usage of the git stash command by using an example and also state the process by giving the screenshot of all the commands written in git bash.

A : git stash : Git stash is a command that is used when you want to work with the current state of the working directory later.

For example : You are working with a directory but suddenly you have to work on another directory but want to keep the current working directory without any lose. In this case you stash the current working directory so that you can use it later. You stash the current working directory after adding the working directory into the staging area.

Process:

1. Create a directory and initialize it using commands
 - a. mkdir directory_name : to create directory
 - b. git init : Initializing the directory into git repository
2. Now create a file in that directory, add it into the staging area and then commit changes
 - a. vim filename : to create a file and input data into it
 - b. git add filename : Moving the file into staging area
 - c. git commit -m "commit message" : To commit changes
3. Then create a new branch, checkout into it and create a file in it.
 - a. git branch Branch_name : to create a new branch.
 - b. git checkout branch_name
4. Right now add the file into the staging area
5. After stash the current working directory using the following command
 - a. git stash save "Git message"
6. Now to restore the stash we use the below command
 - a. git stash pop
7. Push your branches into the remote repository.



```
MINGW64/c/Users/hp/Mystashdir
umma@umma MINGW64 ~ (master)
$ ssh-keygen -t ed25519 -C "kantipudiumaparvathi123@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c:/Users/hp/.ssh/id_ed25519): mysshkey
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in mysshkey
Your public key has been saved in mysshkey.pub
The key fingerprint is:
SHA256:mE3pfwTFF8qf7bBAJLW2g1bknE00gemyoowyOKdcBTI kantipudiumaparvathi123@gmail.com
The key's randomart image is:
--[ED25519 256]--+
|
| . . O+X8o
| . =+==+00 .
| + . + .+0 =
| O .+ . .O .
| . + "OS .
| E O.O . .
| oo+ . .
| B +
|+
+-----[SHA256]-----+
umma@umma MINGW64 ~ (master)
$ git clone git@github.com:umapavathidevi/GitStash.git
Cloning into 'GitStash'...
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
umma@umma MINGW64 ~ (master)
$ git clone https://github.com/umapavathidevi/GitStash.git
Cloning into 'GitStash'...
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.
umma@umma MINGW64 ~ (master)
$ mkdir Mystashdir
umma@umma MINGW64 ~ (master)
$ cd Mystashdir
umma@umma MINGW64 ~/Mystashdir (master)
$ git init
Initialized empty Git repository in c:/Users/hp/Mystashdir/.git/
umma@umma MINGW64 ~/Mystashdir (master)
$ vim Pythonfile.py
umma@umma MINGW64 ~/Mystashdir (master)
$ git status
on branch master
```

```

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git status
On branch master

No commits yet

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

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

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git add .
warning: in the working copy of 'PythonFile.py', LF will be replaced by CRLF the next time Git touches it

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git status
On branch master

No commits yet

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

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git commit -m "Even or odd"
[master (root-commit) 0a60485] Even or odd
1 file changed, 5 insertions(+)
create mode 100644 PythonFile.py

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git log --oneline
0a60485 (HEAD -> master) Even or odd

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git branch Stashbranch

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git checkout Stashbranch
Switched to branch 'Stashbranch'

Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ vim sample.html

Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ git add .
warning: in the working copy of 'sample.html', LF will be replaced by CRLF the next time Git touches it

Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ git stash save "Stashing"
Saved working directory and index state On Stashbranch: Stashing

Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ git checkout master

```



```

Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ git checkout master
Switched to branch 'master'

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git pull origin master
fatal: 'origin' does not appear to be a git repository
fatal: could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git push origin master
fatal: 'origin' does not appear to be a git repository
fatal: could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git remote add AC

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git remote add https://github.com/umaparvathidevi/GitStash.git
usage: git remote add [<options>] <name> <url>

    -f, --fetch           fetch the remote branches
    --tags               import all tags and associated objects when fetching
                        or do not fetch any tag at all (--no-tags)
    -t, --track <branch> branch(es) to track
    -m, --master <branch> master branch
    --mirror[=(push|fetch)] set up remote as a mirror to push to or fetch from

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git remote add origin https://github.com/umaparvathidevi/GitStash.git

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 302 bytes | 302.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/umaparvathidevi/GitStash/pull/new/master
remote:
To https://github.com/umaparvathidevi/GitStash.git
 * [new branch]      master -> master

Uma@Uma MINGW64 ~/Mystashdir (master)
$ git checkout Stashbranch

```



```
MINGW64~/Users/hp/Mystashdir
Uma@Uma MINGW64 ~/Mystashdir (master)
$ git remote add origin https://github.com/umaparthidevi/GitStash.git
Uma@Uma MINGW64 ~/Mystashdir (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 302 bytes | 302.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/umaparthidevi/GitStash/pull/new/master
remote:
to https://github.com/umaparthidevi/GitStash.git
* [new branch]      master -> master
Uma@Uma MINGW64 ~/Mystashdir (master)
$ git checkout Stashbranch
Switched to branch 'Stashbranch'
Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ git stash pop
On branch Stashbranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   sample.html
Dropped refs/stash@{0} (dfbcfc262c9e498be884d4584a4e2aa905d9079a)
Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ git commit -m "Committed the unstash"
[Stashbranch 2d18274] Committed the unstash
 1 file changed, 8 insertions(+)
 create mode 100644 sample.html
Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ git push origin master
Everything up-to-date
Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$ git push origin Stashbranch
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 377 bytes | 377.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'Stashbranch' on GitHub by visiting:
remote:   https://github.com/umaparthidevi/GitStash/pull/new/Stashbranch
remote:
to https://github.com/umaparthidevi/GitStash.git
* [new branch]      Stashbranch -> Stashbranch
Uma@Uma MINGW64 ~/Mystashdir (Stashbranch)
$
```

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.

A: Git Fetch : git fetch command is used to fetch code from the git repository.

Git Merge : git merge command is used to merge two branches into one.

Steps I followed and the commands I used in this process are:

1. I created a new repository in my git hub account and then connected it to my local repository.
2. Then I created a file in master branch and committed it.
3. I fetched my content from remote repository.
4. After I created a new branch and then again created a file in this new branch
5. Later I committed all changes I made in my new file.
6. At last but not least I merged the new branch into the master branch.
7. Finally I push all my changes into the remote repository.

Commands I used in the process are:

1. git init : to initialize the directory.
2. git remote add origin "link" : to connect local and git remote repository.
3. git status : to check the status of the repository.
4. git add . : to add changes made to the repository into the staging area.
5. git commit -m "message" : To commit the changes.
6. git branch branch_name : to create a new branch.
7. git fetch origin : to fetch contents from remote repository.
8. git merge branch_name : to merge the mention branch into the current branch.
9. git log --oneline : to view the history of commits.

No commits yet

Uma@uma MINGW64 ~/FetchMerge (master)

```
MINGW64/c/Users/hp/FetchMerge
$ git branch Merging
Uma@Uma MINGW64 ~/FetchMerge (master)
$ git checkout Merging
Switched to branch 'Merging'
Uma@Uma MINGW64 ~/FetchMerge (Merging)
$ vim Subtraction.py
Uma@Uma MINGW64 ~/FetchMerge (Merging)
$ git add .
warning: in the working copy of 'Subtraction.py', LF will be replaced by CRLF the next time Git touches it
Uma@Uma MINGW64 ~/FetchMerge (Merging)
$ git commit -m "Subtraction"
[Merging 337e243] Subtraction
1 file changed, 4 insertions(+)
create mode 100644 Subtraction.py
Uma@Uma MINGW64 ~/FetchMerge (Merging)
$ git log --oneline
337e243 (HEAD -> Merging) Subtraction
ad5ecbb (master) Addition
Uma@Uma MINGW64 ~/FetchMerge (Merging)
$ git checkout master
Switched to branch 'master'
Uma@Uma MINGW64 ~/FetchMerge (master)
$ git merge Merging
Updating ad5ecbb..337e243
Fast-forward
 Subtraction.py | 4 +++
 1 file changed, 4 insertions(+)
 create mode 100644 Subtraction.py
Uma@Uma MINGW64 ~/FetchMerge (master)
$ git log --oneline
337e243 (HEAD -> master, Merging) Subtraction
ad5ecbb Addition
Uma@Uma MINGW64 ~/FetchMerge (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 535 bytes | 535.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/umaparthidevi/Fetch-and-Merge/pull/new/master
remote:
to https://github.com/umaparthidevi/Fetch-and-Merge.git
 * [new branch]      master -> master
Uma@Uma MINGW64 ~/FetchMerge (master)
$
```

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.

A : Git fetch : git fetch command is used to fetch data from the remote repository and the syntax we use is “git fetch origin”

Git pull : git pull command is used to pull the data from the remote repository and the syntax we use is “git pull origin master”

The major difference between git fetch and git pull is that git fetch fetches the data from remote repository and doesn't make any changes to the local repository whereas git pull integrates the data it fetches from remote repository into the local repository.

```
MINGW64/c/Users/hp/FetchPull
Uma@Uma MINGW64 ~ (master)
$ git clone https://github.com/umaparthidevi/Fetch-vs-Pull.git
fatal: destination path 'Fetch-vs-Pull' already exists and is not an empty directory.
Uma@Uma MINGW64 ~ (master)
$ mkdir FetchPull
Uma@Uma MINGW64 ~ (master)
$ cd FetchPull
Uma@Uma MINGW64 ~/FetchPull (master)
$ git init
Initialized empty Git repository in C:/Users/hp/FetchPull/.git/
Uma@Uma MINGW64 ~/FetchPull (master)
$ git remote add origin https://github.com/umaparthidevi/Fetch-vs-Pull.git
Uma@Uma MINGW64 ~/FetchPull (master)
$ vim Welcome.txt
Uma@Uma MINGW64 ~/FetchPull (master)
$ git add .
warning: in the working copy of 'Welcome.txt', LF will be replaced by CRLF the next time Git touches it
Uma@Uma MINGW64 ~/FetchPull (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Welcome.txt
Uma@Uma MINGW64 ~/FetchPull (master)
$ git commit -m "Welcome file"
[master (root-commit) 11ca1fb] Welcome file
1 file changed, 2 insertions(+)
create mode 100644 Welcome.txt
Uma@Uma MINGW64 ~/FetchPull (master)
$ git fetch origin
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 | 33.00 KiB/s, done.
From https://github.com/umaparthidevi/Fetch-vs-Pull
 * [new branch]      main -> origin/main
Uma@Uma MINGW64 ~/FetchPull (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 254 bytes | 254.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
```

```
MINGW64/c/Users/hp/Pulling
Uma@Uma MINGW64 ~ (master)
$ mkdir Pulling
Uma@Uma MINGW64 ~ (master)
$ cd Pulling
Uma@Uma MINGW64 ~/Pulling (master)
$ git init
Initialized empty Git repository in C:/Users/hp/Pulling/.git/
Uma@Uma MINGW64 ~/Pulling (master)
$ git remote add origin https://github.com/umaparvathidevi/Fetch-vs-Pull.git
Uma@Uma MINGW64 ~/Pulling (master)
$ git pull origin master
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 234 bytes | 19.00 KiB/s, done.
From https://github.com/umaparvathidevi/Fetch-vs-Pull
* branch      master       -> FETCH_HEAD
* [new branch] master       -> origin/master
Uma@Uma MINGW64 ~/Pulling (master)
$
```

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.

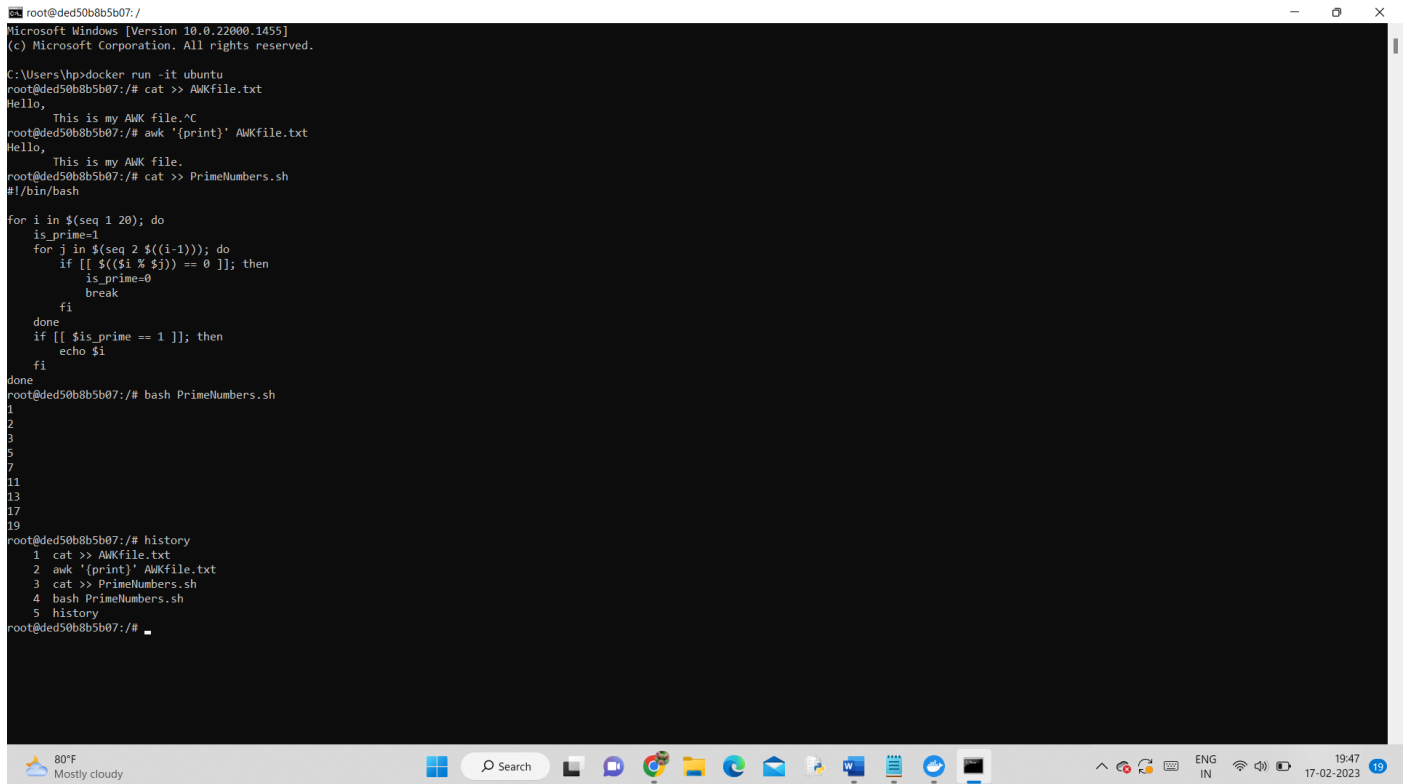
A: AWK command : Awk command is used for text process in Linux. It allows users to process and manipulate data and produce formatted reports.

Here I used `awk '{print}' filename` command to print the content of the file.

```
root@ded50b8b5b07:/
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>docker run -it ubuntu
root@ded50b8b5b07:/# cat >> AwKfile.txt
Hello,
    This is my AwK file.^C
root@ded50b8b5b07:/# awk '{print}' AwKfile.txt
Hello,
    This is my AwK file.
root@ded50b8b5b07:/# cat >> PrimeNumbers.sh
#!/bin/bash

for i in $(seq 1 20); do
    is_prime=1
    for j in $(seq 2 $((i-1))); do
        if [[ $((i % j)) == 0 ]]; then
            is_prime=0
            break
        fi
    done
    if [[ $is_prime == 1 ]]; then
        echo $i
    fi
done
root@ded50b8b5b07:/# bash PrimeNumbers.sh
1
2
3
5
7
11
13
17
19
root@ded50b8b5b07:/# history
1 cat >> AwKfile.txt
2 awk '{print}' AwKfile.txt
3 cat >> PrimeNumbers.sh
4 bash PrimeNumbers.sh
5 history
root@ded50b8b5b07:/#
```



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.

A: Setting up a container: For setting up a container we use pull command ,pull command in docker is used to pull the images from the docker hub to our client machine.

To run a container: To the container in docker we use docker run -tf command .

In this process docker daemon is an intermediary that is used to pull images from docker hub to the client machine.

```
root@13752106227b:/  
Microsoft Windows [Version 10.0.22000.1455]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\hp>cd desktop  
  
C:\Users\hp\Desktop>docker pull ubuntu  
Using default tag: latest  
latest: Pulling from library/ubuntu  
677076032cca: Pull complete  
Digest: sha256:9a0bdd4188b896a372804be2384015e90e3f84906b750c1a53539b585fbbe7f  
Status: Downloaded newer image for ubuntu:latest  
docker.io/library/ubuntu:latest  
  
C:\Users\hp\Desktop>docker run -it ubuntu  
root@13752106227b:/# ls  
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var  
root@13752106227b:/#
```

93°F
Sunny



Search



ENG
IN

16:08
17-02-2023

