Experiment No : 08

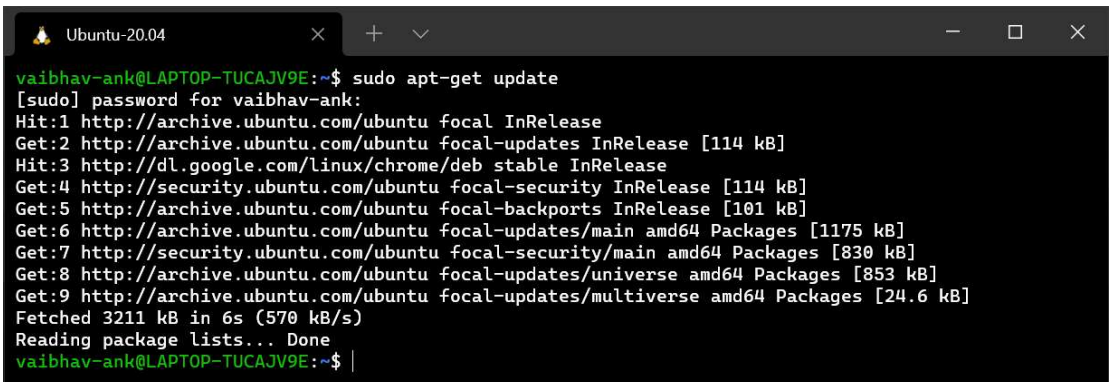
Aim : To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.

Theory :

Option 1: Installing Docker from Official Repository :

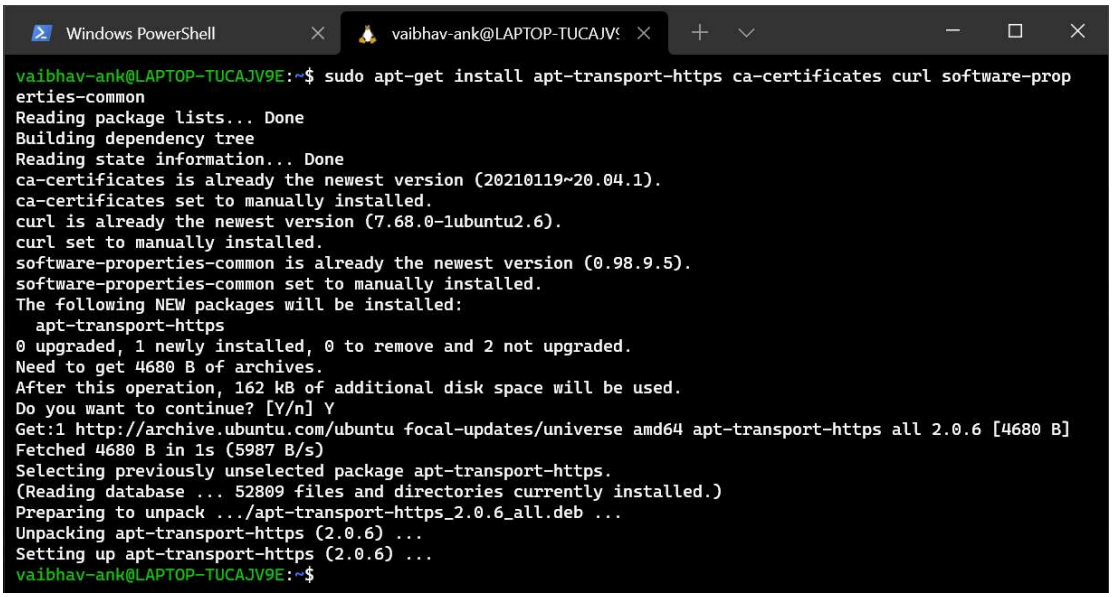
Step 1. Update the apt package index:

```
$ sudo apt-get update
```



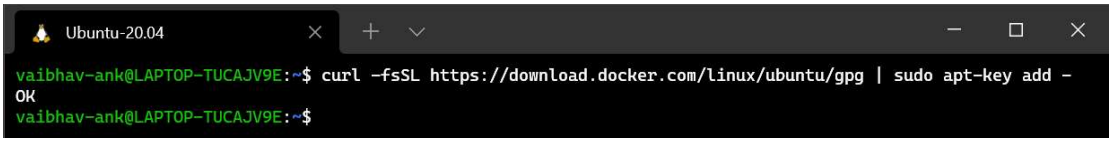
Step 2. Install packages to allow apt to use a repository over HTTPS:

```
$ sudo apt-get install apt-transport-https ca-certificates curl
software-properties-common
```



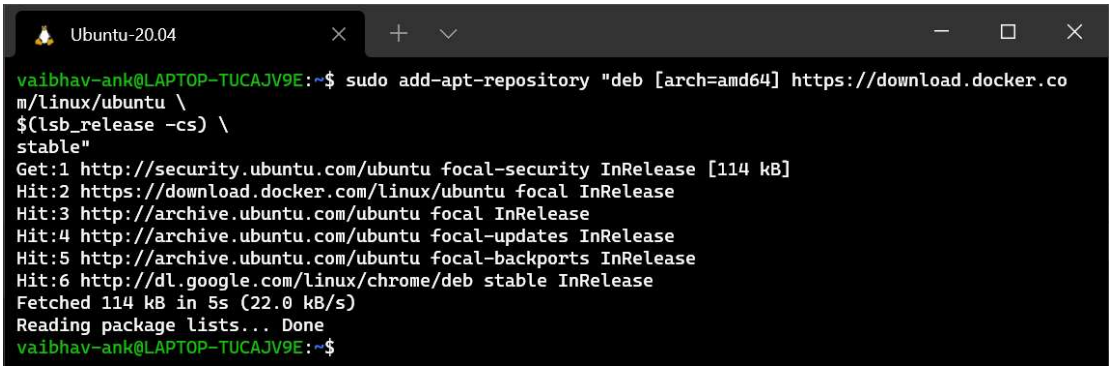
Step 3. Add Docker's official GPG key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
```



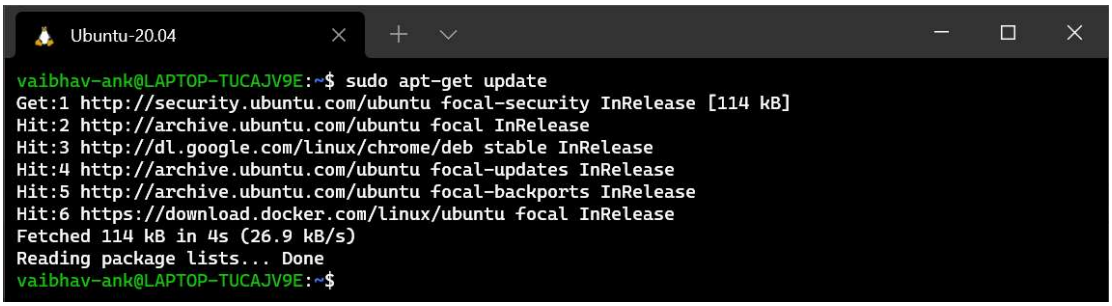
Step 4. Set up the stable repository (command depends on system architecture):

```
$ sudo add-apt-repository \
> "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
> $(lsb_release -cs) \
> stable"
```



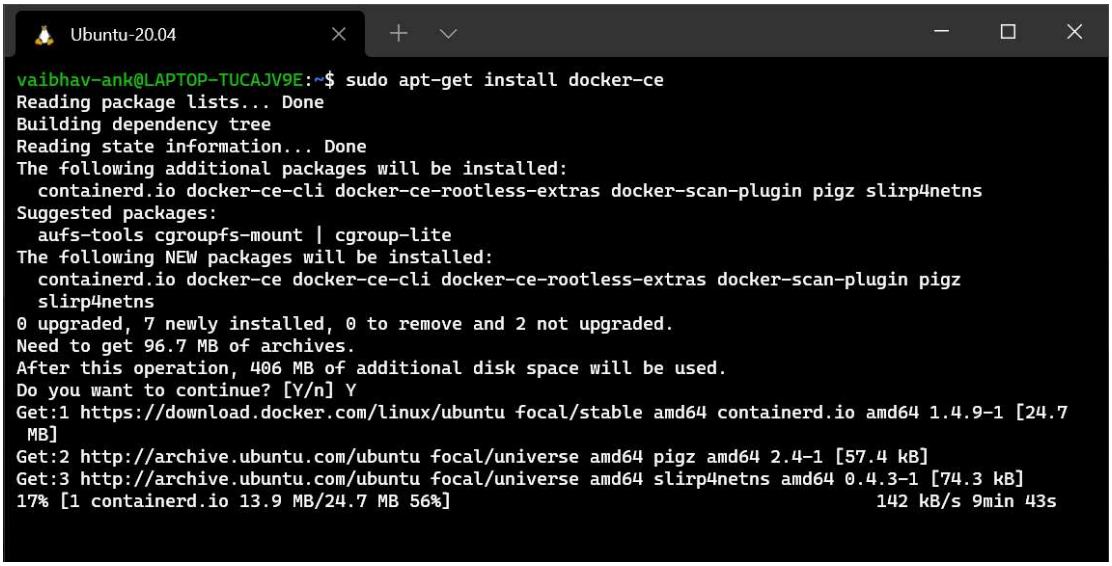
Step 5. Update the package index again:

```
$ sudo apt-get update
```



Step 6. Install the docker-ce package (the service should start automatically after installation):

```
$ sudo apt-get install docker-ce
```



Step 7. To confirm the installation, check the version of Docker:

```
$ docker --version
```

Option 2: Installing Docker from Default Repositories :

To install docker run:

```
$ sudo apt install docker.io
```

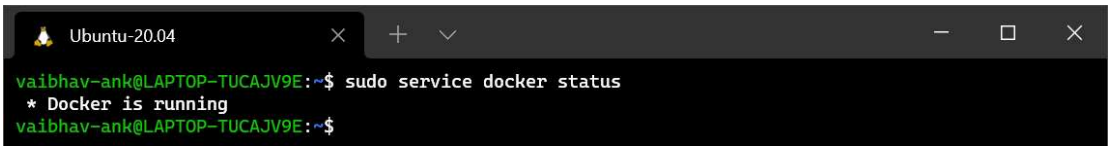
Running Docker Commands with sudo :

Step 1. To start the Docker service run the following commands:

```
$ sudo systemctl start docker OR $ sudo service docker start
```

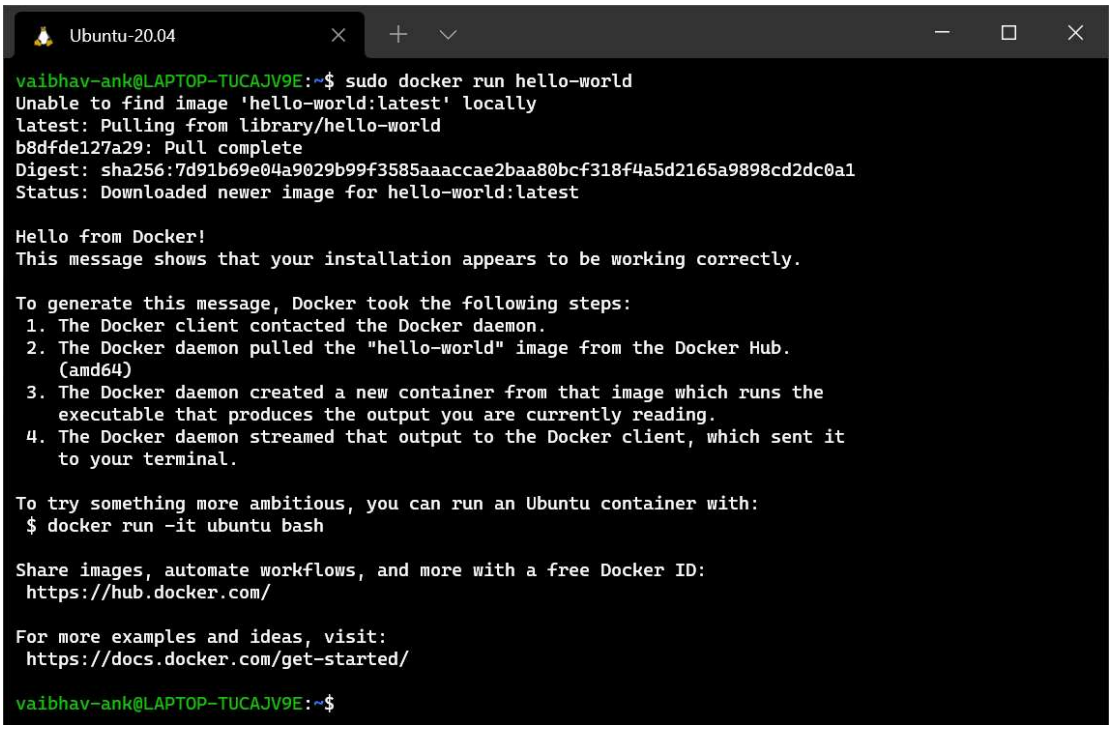
To check the status of the service, use the command:

```
$ sudo systemctl status docker OR $ sudo service docker status
```



Step 2. Verify that docker installed properly:

```
$ sudo docker run hello-world
```



Running Docker Commands without sudo :

By default, you can only use the docker commands with root privileges. Ubuntu requires the use of the sudo prefix. For example, if you try to run a hello-world container, the output displays permission was denied.

```
Ubuntu-20.04
vaibhav-ank@LAPTOP-TUCAJV9E:~$ docker run hello-world
docker: Got permission denied while trying to connect to the Docker daemon socket at unix:
///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/create":
dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
vaibhav-ank@LAPTOP-TUCAJV9E:~$
```

It is advisable to keep the settings as is. However, you can bypass typing sudo every time. Adding the user to the docker group grants privileges equivalent to root.

Step 1. First, create the docker group with the command:

```
$ sudo groupadd docker
```

Step 2. Then, type the following command (making sure to replace [user] with your username):

```
$ sudo usermod -aG docker [user]
```

Step 3. Enable the new settings with:

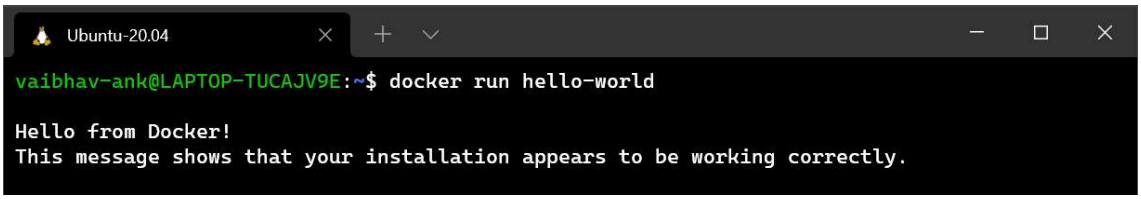
```
$ su - [user]
```

Step 4. Lastly, check to confirm the user is now a part of the docker group by running:

```
$ id -nG
```

```
Ubuntu-20.04
vaibhav-ank@LAPTOP-TUCAJV9E:~$ sudo groupadd docker
groupadd: group 'docker' already exists
vaibhav-ank@LAPTOP-TUCAJV9E:~$ sudo usermod -aG docker vaibhav-ank
vaibhav-ank@LAPTOP-TUCAJV9E:~$ su - vaibhav-ank
Password:
vaibhav-ank@LAPTOP-TUCAJV9E:~$ id -nG
vaibhav-ank adm dialout cdrom floppy sudo audio dip video plugdev netdev wireshark docker
vaibhav-ank@LAPTOP-TUCAJV9E:~$
```

Step 5. Now you can run the docker run hello-world command without the sudo prefix



Working With Docker Images

Docker images are files that contain the source code, libraries, dependencies, tools, and other files a container need. You can create Docker images with Docker files or use existing ones available on Docker Hub.

To download a new Docker image, use the command:

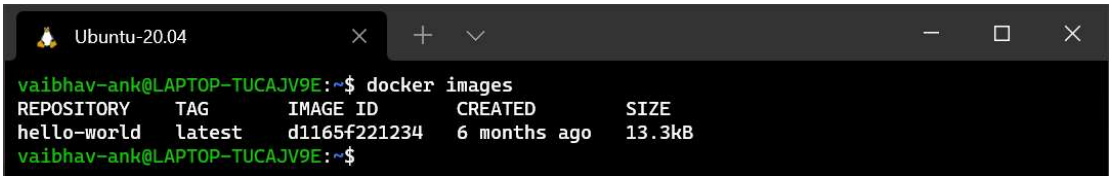
```
$ docker pull [image-name]
```

If you don't know the exact name of the image, search for it in Docker's repository with:

```
$ docker search ubuntu
```

After working with Docker for some time, you will collect a local registry of images. Display a list of all Docker images on the system with:

```
$ docker images
```



Working With Docker Containers

Docker containers are isolated virtual environments that run based on the Docker image assigned to them.

To run a container based on an existing Docker image, use the command:

```
$ docker run [image_name]
```

Using the command above runs a container but doesn't move you inside of it. To run a container in interactive mode

and change to the container command prompt, run:

```
$ docker run -it [image_name]
```

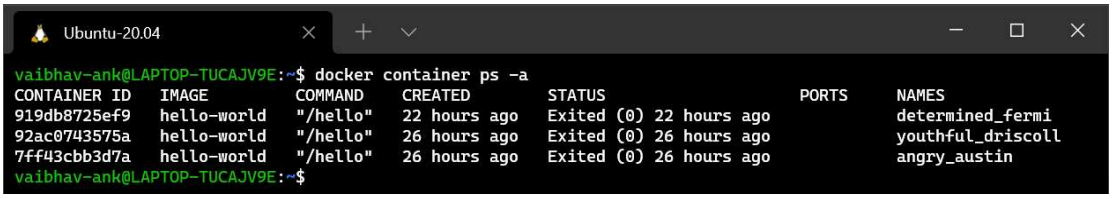
Note: Learn how to run a container in How to Use Docker Run Command with Examples.

Another useful docker command is listing all the containers on the system. To list all active containers, type:

```
$ docker container ps
```

To view all containers (active and inactive), run:

```
$ docker container ps -a
```



Working With Docker Volumes

The best way to preserve data generated within a container is mounting Docker volumes on to them. Mounted volumes don't depend on the container life cycle and can share data between containers.

Create a new Docker volume with:

```
$ docker volume create [volume_name]
```

To create a container and mount a volume to it, follow the syntax:

```
$ docker run [docker_image] \
> --mount source=[volume_name],destination=[path_in_container]
```

Conclusion :

Docker is essentially a toolkit that enables developers to build, deploy, run, update, and stop containers using simple commands and work-saving automation through a single API. Thus, we have successfully installed docker and executed docker commands to manage images and interact with containers.