



Lab Record
of
Container Technologies
(CSF364)



Submitted to:

Mr. Somil Kumar Gupta
Assistant Prof.
School of Computing
DIT University

Submitted by:

Student Name :Mayank Jadoun
Rollno :200102040
CSE-G-P1
3rd Year (VIth Sem)

Session 2022-23



Index

S.No	Title of Experiment/Objective	Date of Conduction	Signature of Faculty
1	Installation of Docker Engine on Ubuntu OS.	25-01-2023	
2	Pull the “hello-world” image from Docker Hub and run this image into aContainer.	1-02-2023	
3	(i) List all the images in Docker Host. (ii) List all the Containers in Docker Host. (iii) Remove all the images in a single command. Remove all the dangling containers.	8-02-2023	
4	Run a container in interactive mode with image ubuntu. Use tag “jammy”for the image. Inside the container install a vim package.	15-02-2023	
5	Create and run a container in interactive and detached mode with imageubuntu. Stop the container. Demonstrate different ways to interact withthe container.	15-02-2023	
6	(i) List all the networks. (ii) Create a custom network. (iii) Connect a previously running container to your custom network. show the IP address assigned from your custom network	15-03-2023	
7	Demonstrate the port mapping to run application on host that are running in a container.	15-03-2023	
8	Demonstrate the ways to build a custom image. (i) through Commit method (ii)through Dockerfile	22-03-2023	
9	Build a custom image and push this image onto a public repository on Docker Hub.	29-03-2023	
10	Create a private registry and push a custom image onto this private registry.	29-02-2023	
11	Build a web server, run it through a container and access it from Docker Host.	29-02-2023	



Practical -1

Objective – Installation of Docker engine on Ubuntu operating System through official repository.

Experiment –

Steps: -

Step 1: Update the Package Repository

Run the following command to update the system's package repository and ensure the latest prerequisite packages are installed:

```
sudo apt update
```

When prompted, enter your root password and press **Enter** to proceed with the update.

Step 2: Install Prerequisite Packages

The [apt package manager](#) requires a few prerequisite packages on the system to use packages over HTTPS. Run the following command to allow Ubuntu to access the Docker repositories over HTTPS:

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

```
bosko@pnap:~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20211016~20.04.1).
ca-certificates set to manually installed.
software-properties-common is already the newest version (0.99.9.8).
software-properties-common set to manually installed.
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  apt-transport-https curl libcurl4
0 upgraded, 3 newly installed, 0 to remove and 5 not upgraded.
Need to get 163 kB/398 kB of archives.
After this operation, 1,283 kB of additional disk space will be used.
```



The command above:

- Allows **apt** to transfer files and data over https.
- Allows the system to check security certificates.
- Installs **curl**, a data-transfer utility.
- Adds scripts for software management.

Step 3: Add GPG Key

- A GPG key verifies the authenticity of a software package. Add the Docker repository GPG key to your system by running:

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

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

Step 4: Add Docker Repository

Run the following command to add the Docker repository to **apt** sources:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

```
bosko@pnap:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
Get:1 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:4 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [18.5 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [40.7 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [77.6 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [278 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [2,464 B]
```

The command adds the official Docker repository and updates the package database with the latest Docker packages.

Step 5: Specify Installation Source

Execute the **apt-cache** command to ensure the Docker installation source is the Docker repository, not the Ubuntu repository. The **apt-cache** command queries the package cache of the **apt** package manager for the Docker packages we have previously added.

Run the following command:

```
apt-cache policy docker-ce
```



```
bosko@pnap:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.18~3-0~ubuntu-focal
  Version table:
   5:20.10.18~3-0~ubuntu-focal 500
     500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
   5:20.10.17~3-0~ubuntu-focal 500
     500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
   5:20.10.16~3-0~ubuntu-focal 500
     500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
   5:20.10.15~3-0~ubuntu-focal 500
     500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
   5:20.10.14~3-0~ubuntu-focal 500
```

The output states which version is the latest in the added source repository.

Step 6: Install Docker

Install Docker by running:

```
sudo apt install docker-ce -y
```

```
bosko@pnap:~$ sudo apt install docker-ce -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bridge-utils ubuntu-fan
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  containerd.io docker-ce-cli docker-ce-rootless-extras docker-scan-plugin
  slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following packages will be REMOVED:
  containerd docker.io runc
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-scan-plugin slirp4netns
0 upgraded, 6 newly installed, 3 to remove and 5 not upgraded.
Need to get 102 MB of archives.
After this operation, 64.1 MB of additional disk space will be used.
```



Wait for the installation process to complete.

Step 7: Check Docker Status

Check if Docker is installed, the daemon started, and the process is enabled to start onboot. Run the following command:

```
sudo systemctl status docker
```

```
bosko@pnap:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
   Active: active (running) since Thu 2022-10-06 18:17:02 EDT; 3min 52s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 2735 (dockerd)
      Tasks: 9
     Memory: 28.2M
    CGroup: /system.slice/docker.service
            └─2735 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>
```

The output states that the Docker daemon is up and running.



Practical -2

Objective: - Pull the “hello-world” image from Docker Hub and run this image into a Container.

Experiments:

Step 1: start the docker daemon by starting docker service.

```
>> sudo service docker start
```

Step 2: To pull the “hello-world” image, issue the following command on docker CLI.

```
>> docker pull hello-world
```

```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33e
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$
```

Step 3: To check the downloaded image, issue the following command.

```
>> docker images OR >> docker image ls
```

Step 4: Now, to run this image into a container, issue the following command.

```
>> docker run hello-world
```

```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$
```




Practical -3

Objective: -

- (i) List all the images in Docker Host.
- (ii) List all the Containers in Docker Host.
- (iii) Remove all the images in a single command.
- (iv) Remove all the dangling containers.

Experiments:

(i) List all the images in Docker Host.

To list all the images downloaded, issue the following command.

```
>> docker images
```

OR

```
>> docker image ls
```

```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
jenkins/jenkins     latest             e701a1b6fb83       3 weeks ago        471MB
jenkins/jenkins     <none>             215dfff17ee4       3 weeks ago        471MB
nginx                latest             080ed0ed8312       4 weeks ago        142MB
myimage              2.0                80a3623344c7       4 weeks ago        228MB
mywebserver          latest             3895bee0e41f       4 weeks ago        228MB
mps123456/myreo      1.0                d14b06aaa4b2       4 weeks ago        176MB
dockertest           latest             7ae6bc46ca7a       4 weeks ago        120MB
<none>               <none>             a3cc808aa505       4 weeks ago        120MB
<none>               <none>             1187f1837c51       4 weeks ago        120MB
node                 latest             0e0ab07dbedd       4 weeks ago        999MB
httpd                2.4                192d41583429       4 weeks ago        145MB
python               latest             254fb6647d87       6 weeks ago        921MB
jenkins/jenkins     lts                d5ed2ceef0ec       7 weeks ago        471MB
ubuntu               latest             08d22c0ceb15       7 weeks ago        77.8MB
registry             2                  0d153fadf70b       2 months ago       24.2MB
openjdk              latest             71260f256d19       2 months ago       470MB
hello-world          latest             feb5d9fea6a5       19 months ago      13.3kB
localhost:5000/centos latest             5d0da3dc9764       19 months ago      231MB
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$
```

It will list all the images either build or download.

(ii) List all the Containers in Docker Host.

- To list all the *running* containers, issue the following command.

```
>> docker container ps
```

```
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker container ps
CONTAINER ID   IMAGE          COMMAND         CREATED        STATUS        PORTS          NAMES
```

- To list all the containers, including *stopped containers*, issue the following command.

```
>> docker container ps -a
```

```
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker container ps -a
CONTAINER ID   IMAGE          COMMAND         CREATED        STATUS        PORTS          NAMES
407135cce298   hello-world    "/hello"        4 minutes ago  Exited (0)    3 minutes ago  mag
737f682308ec   ubuntu:latest  "/bin/bash"     3 weeks ago    Exited (255)  2 weeks ago    rec
b3d3d992adf7   jenkins/jenki  "/usr/bin/tini -- /u..."  3 weeks ago    Exited (255)  2 weeks ago    8080/tcp, 50000/tcp  jen
```

(iii) Remove all the images in a single command.

- To remove a single image, issue the following command.

```
>> docker image rm <image_name or image_id>
```

```
>> docker image rm hello-world
```

**** it is necessary to stop/remove those containers first, which are running the image to which we wish to remove.**

**** to stop/remove a container, command is given in next experiment.**

```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker container rm 4071
4071
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker image rm hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcl
Untagged: hello-world@sha256:ffb13da98453e0f04d33a6eee5bb8e46ee50d08ebe17735fc0779d0349
Deleted: sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412
Deleted: sha256:e07ee1baac5fae6a26f30cabfe54a36d3402f96afda318fe0a96cec4ca393359
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$
```

- To remove all images, issue the following command.

```
>> docker image rmi -f $(docker images -aq)
```

```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker image rmi -f $(docker images -aq)
Untagged: jenkins/jenkins:latest
Untagged: jenkins/jenkins@sha256:562e24559714de46f3fe80a001eccc6507e03c1f32f73bbf51f6113eb45ebc
Deleted: sha256:e701a1b6fb8349e97b3f25d0ba4e00fb3f9668ccb0acf1a6d74d90cbb8f716fb
Untagged: jenkins/jenkins@sha256:6a5d013c4441f83c3e9a36d6bf52897dccc1a93fc1b64860bf5ea66b99cd84
Deleted: sha256:215dfff17ee455087ecbe6cdc00266d52ceb289679d77856b3ecf738319391b4
Deleted: sha256:ab7a92f5ce28fade561a549f4f8bf6d667558c05f6bdf9e0fbf8b30681b3a0b5
Deleted: sha256:b06c7f6c7fd3ceb821d6dda6687bf5e68371cb91e930509fe050109812432504
Deleted: sha256:68669ffcd07e9a7098ba4ff422078bf83a061431fc96d1c97751f3b27cdf0016
Deleted: sha256:b06962fb11b79fdfa4325619b98efd00936bf2b2ebeec6f11e7bbf1618023ebf
Deleted: sha256:bc8865ef3bc7e3acf7b2a34f1c12674de343deeba9b09dcc33a24ffc0f698b1b
Deleted: sha256:c3aefea1bcfbf56d1a1b57cd84d362295a738e06bbf6b5c0e5396718b1bbfe14
Deleted: sha256:547b6acc002bee6a7bad7a56e20251ea606648d5c16d78598547301852a6e681
Deleted: sha256:5ff45f550a34af8ffac6463561f7919d9f566724529d233cb56e35ee2c05928
Deleted: sha256:f4b4df0c8516db9a43b9f778c3b9a0e3b8bb8a0a945440cd2c219d0762181a05
Deleted: sha256:1b1490ed985ce11709b41cb50f6731a4fff531c54996bb655d62a63bbd5fe8f
Deleted: sha256:3b1ff5e0eeba657f0300886a4ddea488ecb0d7210a9c8ad8f27c69080cb6382
Deleted: sha256:3a95fbb1abf9e346d09b856d7798f91f940ab880c2266ad9a9f327f9089e5cde
Untagged: nginx:latest
Untagged: nginx@sha256:2ab30d6ac53580a6db8b657abf0f68d75360ff5cc1670a85acb5bd85ba1b19c0
Deleted: sha256:080ed0ed8312deca92e9a769b518cdfa20f5278359bd156f3469dd8fa532db6b
Deleted: sha256:d78ffa8b3ff6145eb277087027ab2f07317ef3c8155dea0c68aba0be0dc9e357
Deleted: sha256:52c16551e27714b78f78d51673e87d8327b6878f61f5ed37bbda99b30bd6258f
Deleted: sha256:f71bcd5f9b7b85bc15acf26db665dbd64f1e59fc13cbdfa7b7cac4fed5c3ac1
Deleted: sha256:e5804950d08e966c6a00a433c67a12cff6347e5e2108865dfde6148f8d18130e
Deleted: sha256:4da23111cc0445f1a069ac60b0575acfc86218f0783d48440f32b35bde461a41
Untagged: myimage:2.0
Deleted: sha256:80a3623344c748e67c605dbac77d66d1a0c9b07ecc030e42eeaeef75131e35f8
Untagged: mywebserver:latest
Deleted: sha256:3895bee0e41f8dde3c375c8208b9431bd4024e531fb0fdb3a9aba09910c2793e
```

(iv) Remove all the dangling containers.

To remove all the dangling/stopped containers, issue the following command.

```
>> docker container prune
```

```
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
737f682308ec0c5481668d9c304e7b9708d35fe3549b6dc5c97814cf1a1503e5
b3d3d992adf7327e25b12fe796ca7f154354010bb6ab849f841b473f8b50cb20

Total reclaimed space: 3.419MB
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$
```

** to remove all the unused containers, images and volumes at once, you can issue the following command.

```
>> docker system prune -a
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker system prune -a
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all images without at least one container associated to them
- all build cache

Are you sure you want to continue? [y/N]
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$
```

Practical -4

Objective: - Run a container in interactive mode with image ubuntu. Use tag “jammy” for the image. Inside the container install a vim package.

Experiment:

Step 1: to run a container in interactive mode, issue the following command.

```
>> docker run -it ubuntu:jammy /bin/bash
```

```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker run -it ubuntu:jammy /bin/bash
Unable to find image 'ubuntu:jammy' locally
jammy: Pulling from library/ubuntu
Digest: sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babbc295a0428a6d21
Status: Downloaded newer image for ubuntu:jammy
root@75a22b5060ee:/#
```

- This command will download image “ubuntu” with tag “jammy” and with the help of “-it” parameter, run command will run a container having image ubuntu:jammy in interactive mode.
- As we can see, after the successful deployment of this image into a container, we can access the terminal (/bin/bash) of this new ubuntu image.

Step 2: Here, to install vim package issue the following command.

```
>> apt-get install vim
```

```
root@79c2207e734a:/# apt-get install vim
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
vim is already the newest version (2:8.2.3995-1ubuntu2.7).
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
```




To verify the installation of vim package, issue the following command.

```
>> vim -info
```

```
root@79c2207e734a:/# vim -info
VIM - Vi IMproved 8.2 (2019 Dec 12, compiled Apr 18 2023 11:40:57)
Garbage after option argument: "-info"
More info with: "vim -h"
```

Practical -5

Objective: - Create and run a container in interactive and detached mode with image ubuntu. Stop the container. Demonstrate different ways to interact with the container.

Experiment:

Step 1: to run a container in interactive and detached mode with image ubuntu, issue the following command.

```
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker run -it -d ubuntu /bin/
d1f57b27927e36e1a8276fdf6b95cc212a85efd5db320dc7abc6c389f697c05d
>> docker run -it -d ubuntu /bin/bash
```

To see the container running in detached mode, issue the following command.

```
>> docker ps
```

```
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
d1f57b27927e   ubuntu    "/bin/bash"             24 seconds ago Up 23 seconds          nervou
```

Step 2: to connect with this running container, there are multiple ways.

(a) with the help of “attach” command.

```
>> docker attach <container_ID>
```

```
>> docker container attach a3591f76a095
```

```
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker container attach d1f571
root@d1f57b27927e:/#
```

(b) with the help of “exec” command.

```
>> docker exec -it <container_ID> <command>
```

```
>> docker exec -it a3591f76a095 /bin/bash
```

```
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker start 75a22b5060ee
75a22b5060ee
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker exec -it 75a22b5060ee /bin/ba
root@75a22b5060ee:/#
```

Step 3: to stop the running container, issue the following command.

```
>> docker container stop <container_ID>
```

```
>> docker container stop a3591f76a095
```

```
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$ docker container stop 75a22b5060ee
75a22b5060ee
mayank@DESKTOP-4ERP5E:/mnt/c/Windows/system32$
```

Practical -6

Objective: -

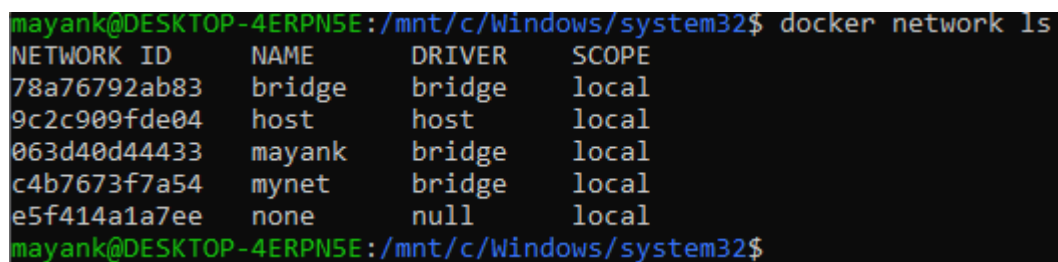
- (i) List all the networks.
- (ii) Create a custom network.
- (iii) Connect a previously running container to your custom network.
- (iv) show the IP address assigned from your custom network

Experiment:

(i) List all the networks.

Step 1: to list all the networks, issue the following command.

```
>> docker network ls
```



```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker network ls
NETWORK ID          NAME       DRIVER  SCOPE
78a76792ab83        bridge    bridge  local
9c2c909fde04        host      host    local
063d40d44433        mayank    bridge  local
c4b7673f7a54        mynet     bridge  local
e5f414a1a7ee        none      null    local
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$
```

** this will list all the system defined (bridge, host and null) and custom-madenetworks.

(ii) Create a custom network.

Step 1: to create a custom network, issue the following command.

```
>> docker network create --
driver=bridge --subnet=<subnet-
address> <net_name>
```

```
>> docker network create --
driver=bridge --
subnet=192.168.1.0/24 mynet
```

```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker network create --driver=bridge --subnet=192.168.1.0/24 mynet
c4b7673f7a549c813ef9c47a724c41baf91768d48ef5eb705da9755596e5f036

mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker network ls
NETWORK ID        NAME        DRIVER        SCOPE
78a76792ab83     bridge     bridge        local
9c2c909fde04     host       host          local
063d40d44433     mayank     bridge        local
c4b7673f7a54     mynet      bridge        local
e5f414a1a7ee     none      null          local
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$
```

(iii) Connect a previously running container to your custom network.

(a) Now, to connect this custom network “mynet”, to previously running container, issue the following command.

```
>> docker network connect <custom-net> <container-ID>
```

```
>> docker network connect mynet 858e94f7c9b4
```

```
mayank@DESKTOP-4ERPNS5:~$ docker container ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
d1f57b27927e   ubuntu    "/bin/bash"             24 hours ago   Exited (0)   24 hours ago   nervous_pare
75a22b5060ee   ubuntu:jammy "/bin/bash"             24 hours ago   Exited (137) 24 hours ago   ecstatic_ca
6ec2553d3639   ubuntu    "/bin/bash"             24 hours ago   Exited (0)   24 hours ago   musing_mats

mayank@DESKTOP-4ERPNS5:~$
mayank@DESKTOP-4ERPNS5:~$ docker network connect mynet d1f57b27927e
mayank@DESKTOP-4ERPNS5:~$ docker network inspect mynet

mayank@DESKTOP-4ERPNS5:~$ docker container inspect nervous_pare
[
  {
    "Id": "d1f57b27927e36e1a8276fdf6b95cc212a85efd5db320dc7abc6c389f697c05d",
```


mayank@DESKTOP-4ERPNS5: ~

```
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"IPAddress": "172.17.0.2",
"IPPrefixLen": 16,
"IPv6Gateway": "",
"MacAddress": "02:42:ac:11:00:02",
"Networks": {
  "bridge": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": null,
    "NetworkID": "78a76792ab831aed86e45c2b2524134729b1e4a612e8cd808d31f85f35d62562",
    "EndpointID": "cda5209cb7584c6b83e0eb831158d99f5ce600a536de707d17a1d919cc38128",
    "Gateway": "172.17.0.1",
    "IPAddress": "172.17.0.2",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "MacAddress": "02:42:ac:11:00:02",
    "DriverOpts": null
  },
  "mynet": {
    "IPAMConfig": {},
    "Links": null,
    "Aliases": [
      "d1f57b27927e"
    ],
    "NetworkID": "c4b7673f7a549c813ef9c47a724c41baf91768d48ef5eb705da9755596e5f036",
    "EndpointID": "00854d000f35bac2fa833e8d7bd12acf09d883fd68ac375fb4a11b154fd95d2",
    "Gateway": "192.168.1.1",
    "IPAddress": "192.168.1.2",
    "IPPrefixLen": 24,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "MacAddress": "02:42:c0:a8:01:02",
    "DriverOpts": {}
  }
}
```

(a) Now, to connect a new container with this custom network, issue the following command.

```
>> docker container run -itd -
name=<container_name> --net=<custom-net> ubuntu
```

```
>> docker container run -itd --name=mycontainernet --net=mynet
ubuntu
```

```
mayank@DESKTOP-4ERPNS5:~$ docker container run -itd --name=mycontainer --net=mynet ubuntu
ce3ddd7698e30cc14a7bd11d2e0602e77c20bfe16679816f0943a80ac5e8e999
```

** to see the connected network(s) of this container “mycontainernet”, inspect the container by issuing following command.

```
>> docker inspect <container-name>
```

```
>> docker inspect mycontainernet
```

```
ayank@DESKTOP-4ERP5E:~$ docker inspect mycontainer
[
  {
    "Id": "ce3ddd7698e30cc14a7bd11d2e0602e77c20bfe16679816f0943a80ac5e8e999",
    "Created": "2023-04-27T16:02:23.257524795Z",
    "Path": "/bin/bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 1782,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-04-27T16:02:24.08702083Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Networks": {
      "mynet": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": [
          "ce3ddd7698e3"
        ],
        "NetworkID": "c4b7673f7a549c813ef9c47a724c41baf91768d48ef5eb705da9755596e5f0",
        "EndpointID": "abc9be137f5c4a6fd005d51706984c56e9c088314983d2c525f7c27705548",
        "Gateway": "192.168.1.1",
        "IPAddress": "192.168.1.3",
        "IPPrefixLen": 24,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:c0:a8:01:03",
        "DriverOpts": null
      }
    }
  }
]
```

(i) Show the IP address assigned from your custom network

To show the assigned IP address from the connected custom network

“mycontainernet”, the file received from “inspect” command will display

the result. As, IP packet assigned to our custom network “mynet” is 192.168.1.0/24. The IP address assigned to container “mycontainernet” is 192.168.1.1.

```
"Networks": {
  "mynet": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": [
      "ce3ddd7698e3"
    ],
    "NetworkID": "c4b7673f7a549c813ef9c47a724c41baf91768d48ef5eb705da9755596e5f036",
    "EndpointID": "abc9be137f5c4a6fd005d51706984c56e9c088314983d2c525f7c27705548407",
    "Gateway": "192.168.1.1",
    "IPAddress": "192.168.1.3",
    "IPPrefixLen": 24,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "MacAddress": "02:42:c0:a8:01:03",
    "DriverOpts": null
  }
}
```

Practical -7

Objective: - Demonstrate the port mapping to run application on host that are running in a container.

Experiment:

Step 1 – First, you need to do a simple sign-up on Docker Hub.

Step 2 – Once you have signed up, you will be logged into Docker Hub.

Step 3 – Next, let's browse and find the Jenkins image.

Step 4 – If you scroll down on the same page, you can see the Docker pull command. This will be used to download the Jenkins Image onto the local Ubuntu server.

Step 5 – Now go to the Ubuntu server and run the command –

```
>>sudo docker pull Jenkins/Jenkins:lts
```

Step 6 – To understand what ports are exposed by the container, you should use the *Docker inspect* command to inspect the image.

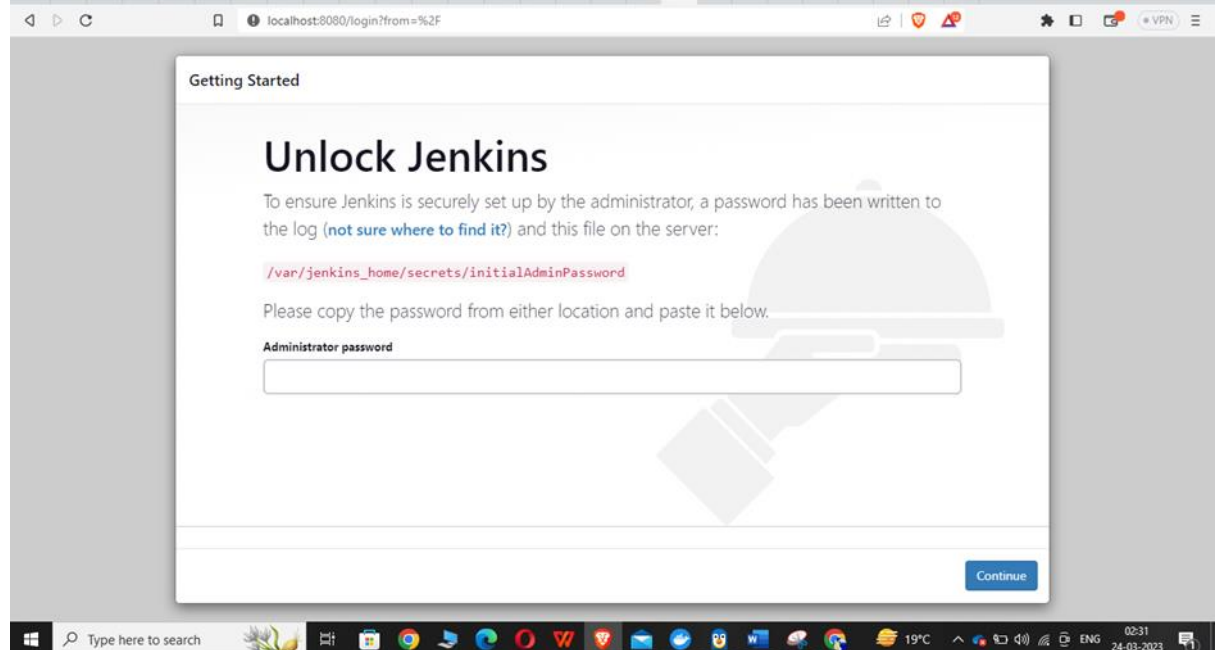
- The output of the inspect command gives a JSON output. If we observe the output, we can see that there is a section of "ExposedPorts" and see that there are two ports mentioned. One is the data port of 8080 and the other is the control port of 50000.
- To run Jenkins and map the ports, you need to change the Docker run command and add the 'p' option which specifies the port mapping. So, you need to run the following command –

```
>> sudo docker run -p 8080:8080 -p  
50000:50000jenkins/jenkins:lts
```

- The left-hand side of the port number mapping is the Docker host port to map to and the right-hand side is the Docker container port number.

When you open the browser and navigate to the Docker host on port 8080, you will see Jenkins up and running

```
ayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ sudo docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts
Running from: /usr/share/jenkins/jenkins.war
Webroot: /var/jenkins_home/war
023-03-23 17:49:45.462+0000 [id=1] INFO winstone.Logger#logInternal: Beginning extraction from war file
023-03-23 17:49:50.699+0000 [id=1] WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
023-03-23 17:49:50.933+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: jetty-10.0.13; built: 2022
a98ac084c766d; jvm 11.0.18+10
023-03-23 17:49:52.202+0000 [id=1] INFO o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support fo
t
023-03-23 17:49:52.584+0000 [id=1] INFO o.e.j.s.s.DefaultSessionIdManager#doStart: Session workerName=node0
023-03-23 17:49:54.420+0000 [id=1] INFO hudson.WebAppMain#contextInitialized: Jenkins home directory: /var/
INS_HOME")
023-03-23 17:49:54.883+0000 [id=1] INFO o.e.j.s.handler.ContextHandler#doStart: Started w.@39c385d6{Jenkins
var/jenkins_home/war}
023-03-23 17:49:54.957+0000 [id=1] INFO o.e.j.server.AbstractConnector#doStart: Started ServerConnector@74a
023-03-23 17:49:55.029+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: Started Server@3543df7d{ST
023-03-23 17:49:55.047+0000 [id=24] INFO winstone.Logger#logInternal: Winstone Servlet Engine running: contr
023-03-23 17:49:56.187+0000 [id=31] INFO jenkins.InitReactorRunner$1#onAttained: Started initialization
023-03-23 17:49:56.267+0000 [id=30] INFO jenkins.InitReactorRunner$1#onAttained: Listed all plugins
023-03-23 17:50:00.836+0000 [id=33] INFO jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
023-03-23 17:50:00.855+0000 [id=33] INFO jenkins.InitReactorRunner$1#onAttained: Started all plugins
023-03-23 17:50:00.878+0000 [id=29] INFO jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
023-03-23 17:50:02.247+0000 [id=31] INFO jenkins.InitReactorRunner$1#onAttained: System config loaded
023-03-23 17:50:02.249+0000 [id=32] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
023-03-23 17:50:02.250+0000 [id=32] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
023-03-23 17:50:02.263+0000 [id=36] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs
023-03-23 17:50:02.405+0000 [id=49] INFO hudson.util.Retrier#start: Attempt #1 to do the action check update
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v7.Java7$1 (file:/var/jenkins_home/war/WEB-INF/L
oke.MethodHandles$Lookup(java.lang.Class,int)
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.vmplugin.v7.Java7$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
023-03-23 17:50:04.947+0000 [id=34] INFO jenkins.install.SetupWizard#init:
```



Practical -8

Objective: - Demonstrate the ways to build a custom image.

(i) through Commit method

(ii) through Dockerfile

Experiment:

(i) through Commit method

Step 1: pull a Docker image and deploy a container with this image.

For this experiment, we have pulled an ubuntu image from docker hub and we have noted its image-id as below.

```
mayank@DESKTOP-4ERP5E:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	jammy	08d22c0ceb15	7 weeks ago	77.8MB
ubuntu	latest	08d22c0ceb15	7 weeks ago	77.8MB

The image-id for this ubuntu:latest image is 08d22c0ceb15. Now deploy a container with this image.

```
>> docker run -it 08d22c0ceb15 bin/bash
```

Step 2: Modify the Container

Now we are in the container, we can modify the image. In the following experiment, we will add the **Nmap software** for network discovery and security auditing.

```
>> apt-get install nmap
```



```
root@404aeb44039:/# apt-get install nmap
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  dbus libapparmor1 libblas3 libdbus-1-3 libexpat1 liblinear4 liblua5.3-0 libpcap0.8 lua-lpeg nmap-common
Suggested packages:
  default-dbus-session-bus | dbus-session-bus liblinear-tools liblinear-dev ncat ndiff zenmap
The following NEW packages will be installed:
  dbus libapparmor1 libblas3 libdbus-1-3 libexpat1 liblinear4 liblua5.3-0 libpcap0.8 lua-lpeg nmap nmap-common
0 upgraded, 11 newly installed, 0 to remove and 5 not upgraded.
root@404aeb44039:/# nmap --version
Nmap version 7.80 ( https://nmap.org )
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.3.6 openssl-3.0.2 nmap-libssh2-1.8.2 libz-1.2.11 libpcap-1.10.1 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
```

Now, exit from this container. Now list the containers to note the container-id of previous running container.

```
mayank@DESKTOP-4ERPNS5:/mnt/c/windows/system32$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS          NAMES
7df811717aaf   08d22c0ceb15   "bin/bash"              3 minutes ago   Exited (100) 2 minutes ago          fervent_chandrasekhar
```

Step 3: Commit changes to image.

create a new image by committing the changes using the following syntax:

```
>> docker commit [CONTAINER_ID] [new_image_name]
```

```
>> docker commit 404aeb44039 ubuntu-nmap
```

Now, if you list the images, a new image named “ubuntu-nmap” will be shown.

```
mayank@DESKTOP-4ERPNS5:/mnt/c/windows/system32$ docker commit 7df811717aaf ubuntu-nmap
sha256:a2bfd18c17d1a68913c0922a52de54b6a9626fcd2c93ce72e42dec9bba16e21a
mayank@DESKTOP-4ERPNS5:/mnt/c/windows/system32$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu-nmap   latest    a2bfd18c17d1   20 seconds ago  77.8MB
ubuntu        jammy     08d22c0ceb15   7 weeks ago    77.8MB
ubuntu        latest    08d22c0ceb15   7 weeks ago    77.8MB
mayank@DESKTOP-4ERPNS5:/mnt/c/windows/system32$
```

(i) through Dockerfile

Step 1 – Create a file called Docker File and edit it using vim. Please notethat the name of the file must be "Dockerfile" with "D" as capital.

```
mayank@DESKTOP-4ERP5E:~$ vim Dockerfile
```

Step 2 – Build your Docker File using the following instructions.

```
#This is a
sample ImageFROM
ubuntu
MAINTAINER
demousr@gmail.comRUN
apt-get update
RUN apt-get install -y nginx
CMD ["echo","Image created"]
```

******The following points need to be noted about the previous file –

- The first line "***#This is a sample Image***" is a **comment**. You can add comments to the Docker File with the help of the # command
- The next line has to start with the **FROM** keyword. It tells docker, from whichbase image you want to base your image from. In our example, we are creatingan image from the ubuntu image.
- The next command is the person who is going to ***maintain*** this image. Here you specify the **MAINTAINER** keyword and just mention the ***email ID***.
- The **RUN** command is used to run instructions against the image. In our case,we first update our Ubuntu system and then install the **nginx server** on our **ubuntu image**.
- The last command is used to **display a message** to the user.**

```

mayank@DESKTOP-4ERPNS5E: ~
FROM ubuntu
MAINTAINER mayankjadoun2002@gmail.com
RUN apt-get update
RUN apt-get install -y nginx
CMD ["echo","Image updated"]

```

The Docker File can be built with the following command –

```
>> docker build
```

This method allows the users to build their own Docker images.

- Syntax

```
>> docker build -t ImageName:TagName dir
```

- Options

- -t – is to mention a tag to the image
- ImageName – This is the name you want to give to your image.
- TagName – This is the tag you want to give to your image.
- Dir – The directory where the Docker File is present.

- Return Value

None

```
>> sudo docker build -t myimage:0.1 .
```

- Here, “**myimage**” is the name we are giving to the Image and 0.1 is the tag number we are giving to our image.
- Since the Docker File is in the present working directory, we used “.” at the end of the command to signify the present working directory.



```
mayank@DESKTOP-4ERPNS5:~$ docker build -t myimage:1.0 .
[+] Building 0.3s (7/7) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 378                                                0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 28                                                    0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest                 0.0s
=> [1/3] FROM docker.io/library/ubuntu                                          0.0s
=> CACHED [2/3] RUN apt-get update                                              0.0s
=> CACHED [3/3] RUN apt-get install -y nginx                                    0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:d14b06aaa4b29215d48fdc2cc7c74284dac2209ae390eeae865ada94942ee240 0.0s
=> => naming to docker.io/library/myimage:1.0                                    0.0s
mayank@DESKTOP-4ERPNS5:~$ docker images

=> => transferring context: 28                                                    0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest                 0.0s
=> [1/3] FROM docker.io/library/ubuntu                                          0.0s
=> CACHED [2/3] RUN apt-get update                                              0.0s
=> CACHED [3/3] RUN apt-get install -y nginx                                    0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:d14b06aaa4b29215d48fdc2cc7c74284dac2209ae390eeae865ada94942ee240 0.0s
=> => naming to docker.io/library/myimage:1.0                                    0.0s
mayank@DESKTOP-4ERPNS5:~$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
myimage             1.0         d14b06aaa4b2 3 hours ago   176MB
httpd               2.4         192d41583429 5 days ago    145MB
python              latest      254fb6647d87 2 weeks ago   921MB
jenkins/jenkins     lts         d5ed2ceef0ec 2 weeks ago   471MB
ubuntu              latest      08d22c0ceb15 2 weeks ago   77.8MB
registry            2           0d153fadf70b 6 weeks ago   24.2MB
openjdk             latest      71260f256d19 6 weeks ago   470MB
hello-world         latest      feb5d9fea6a5 18 months ago 13.3kB
localhost:5000/centos latest      5d0da3dc9764 18 months ago 231MB
mayank@DESKTOP-4ERPNS5:~$ docker tag d14b06aaa4b2 mps123456/myreo:1.0
mayank@DESKTOP-4ERPNS5:~$ docker push mps123456/myreo:1.0
The push refers to repository [docker.io/mps123456/myreo]
cb68466d22a9: Pushed
c78edd87e18b: Pushed
b93c1bd012ab: Mounted from library/ubuntu
1.0: digest: sha256:771276fbaa5f406ab6a5c2ba7ed371b9e3c262a303ecbad94d21c87c40078140 size: 953
mayank@DESKTOP-4ERPNS5:~$
```


Practical -9

Objective: - Build a custom image and push this image onto a public repository on Docker Hub.

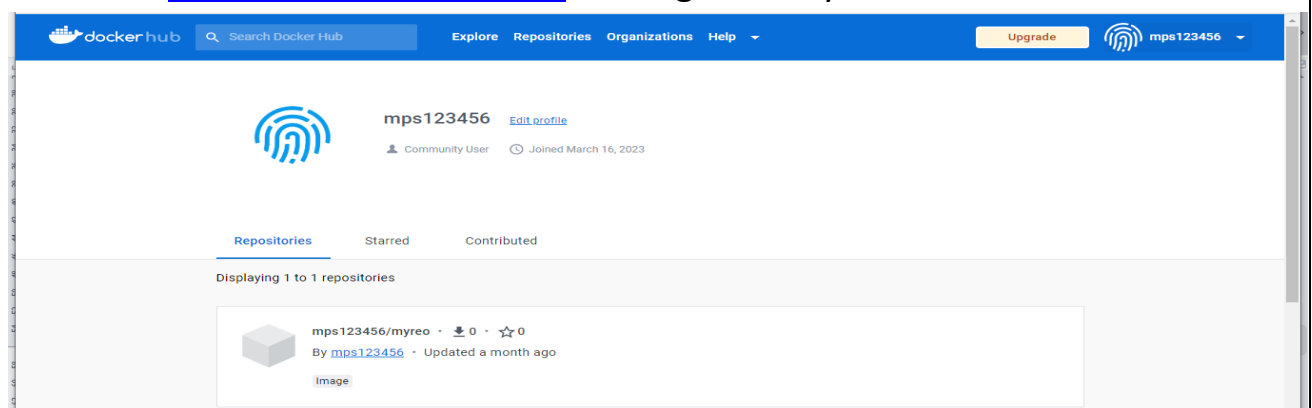
Experiment:

- In the previous practical (08), we have created an image “myimage:0.1”. Now we will upload (**push**) this image to Docker public repository in following steps.

```
ERROR response from daemon: conflict: unable to delete 5d0da3dc9764 (can
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	2.4	192d41583429	5 days ago	145MB
python	latest	254fb6647d87	2 weeks ago	921MB
jenkins/jenkins	lts	d5ed2ceef0ec	2 weeks ago	471MB
ubuntu	latest	08d22c0ceb15	2 weeks ago	77.8MB
registry	2	0d153fadf70b	6 weeks ago	24.2MB
openjdk	latest	71260f256d19	6 weeks ago	470MB
hello-world	latest	feb5d9fea6a5	18 months ago	13.3kB
localhost:5000/centos	latest	5d0da3dc9764	18 months ago	231MB

- Step 1** – Log into Docker Hub and create your repository. This is the repository where your image will be stored. Go to <https://hub.docker.com/> and log in with your credentials.



- In the previous step, we have created an image “**myimage:0.1**”. Now we will upload (**push**) this image to Docker public repository in following steps.

Step 2 – Click the button "Create Repository" on the above screen and create a repository with the name *demorep*. Make sure that the visibility of the repository is public.

- Once the repository is created, make a note of the pull command which is attached to the repository.
- The pull command which will be used in our repository is as follows

```
>>docker pull demouser/demorep
```

Step 3 – Now go back to the Docker Host. Here we need to tag our “myimage” to the new repository created in Docker Hub. We can do this via the Docker tag command.

```
>>docker tag imageID Repositoryname
```

Step 4 – Issue the Docker login command to login into the Docker Hub repository from the command prompt. The Docker login command will prompt you for the username and password to the Docker Hub repository.

Step 5 – Once the image has been tagged, it's now time to push the image to the Docker Hub repository. We can do this via the Docker push command.




```
=> => transferring context: 28 0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 0.0s
=> [1/3] FROM docker.io/library/ubuntu 0.0s
=> CACHED [2/3] RUN apt-get update 0.0s
=> CACHED [3/3] RUN apt-get install -y nginx 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:d14b06aaa4b29215d48fdc2cc7c74284dac2209ae390eeae865ada94942ee240 0.0s
=> => naming to docker.io/library/myimage:1.0 0.0s
mayank@DESKTOP-4ERPNS5:~$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
myimage             1.0            d14b06aaa4b2   3 hours ago    176MB
httpd               2.4            192d41583429   5 days ago     145MB
python              latest         254fb6647d87   2 weeks ago    921MB
jenkins/jenkins     lts           d5ed2ceef0ec   2 weeks ago    471MB
ubuntu              latest         08d22c0ceb15   2 weeks ago    77.8MB
registry            2             0d153fadb70b   6 weeks ago    24.2MB
openjdk             latest         71260f256d19   6 weeks ago    470MB
hello-world         latest         feb5d9fea6a5   18 months ago  13.3kB
localhost:5000/centos latest         5d0da3dc9764   18 months ago  231MB
mayank@DESKTOP-4ERPNS5:~$ docker tag d14b06aaa4b2 mps123456/myreo:1.0
mayank@DESKTOP-4ERPNS5:~$ docker push mps123456/myreo:1.0
The push refers to repository [docker.io/mps123456/myreo]
cb68466d22a9: Pushed
c78edd87e18b: Pushed
p93c1bd012ab: Mounted from library/ubuntu
1.0: digest: sha256:771276fb0a5f406ab6a5c2ba7ed371b9e3c262a303ecbad94d21c87c40078140 size: 953
mayank@DESKTOP-4ERPNS5:~$
```


mps123456 / myreo

Description



this is used for 

 Last pushed: a few seconds ago

Tags

 IMAGE INSIGHTS INACTIVE
[Activate](#)

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
 1.0		Image	---	6 minutes ago

[See all](#)

[Go to Advanced Image Management](#)

Practical -10

Objective: - Create a private registry and push a custom image onto this privateregistry.

Experiment:

Step 1 – Use the Docker run command to download the private registry. This can be done using the following command.

```
>> sudo docker run -d -p 5000:5000 --name registry registry:2
```

- The following points need to be noted about the above command –
1. **registry** is the container managed by Docker which can be used to host privaterepositories.
 2. The **port number** exposed by the container is 5000. Hence with the **-p** command, we are mapping the same port number to the 5000-port number on our localhost.
 3. We are just tagging the **registry container** as “2”, to differentiate it on the Docker host.
 4. The **-d** option is used to run the container in detached mode. This is so that the container can run in the background.

Step 2 – Let's do a `docker ps` to see that the registry container is indeed running.

```
nayank@DESKTOP-4ERPNS5:/mnt/c/windows/system32$ docker run -d -p 5000:5000 --name registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
ef5531b6e74e: Pull complete
a52704366974: Pull complete
dda5a8ba6f46: Pull complete
eb9a2e8a8f76: Pull complete
25bb6825962e: Pull complete
Digest: sha256:41f413c22d6156587e2a51f3e80c09808b8c70e82be149b82b5e0196a88d49b4
Status: Downloaded newer image for registry:2
6c9fce15cc44d1cde17d738388cb831b53445cd291afcb60bfa3881048bd6c12
```

We have now confirmed that the registry container is indeed running.

Step 3 – Now let's tag one of our existing images so that we can push it to our local repository. In our example, since we have the



centos image available locally, we are going to tag it to our private repository and add a tag name of centos.

```
>>sudo docker tag 67591570dd29 localhost:5000/centos
```

```
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker tag 5d0da3dc9764 localhost:5000/centos
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker push localhost:5000/centos
Using default tag: latest
The push refers to repository [localhost:5000/centos]
74ddd8ec08fa: Pushed
latest: digest: sha256:a1801b843b1bfaf77c501e7a6d3f709401a1e0c83863037fa3aab063a7fdb9dc size: 529
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker rmi centos:latest
Untagged: centos:latest
Untagged: centos@sha256:a27fd8080b517143cbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
mayank@DESKTOP-4ERPNS5:/mnt/c/Windows/system32$ docker rmi 5d0da3dc9764
Error response from daemon: conflict: unable to delete 5d0da3dc9764 (cannot be forced) - image is being used by running container 2c6963025cdf
```

- The following points need to be noted about the above command –

1. 67591570dd29 refers to the Image ID for the centos image.
2. localhost:5000 is the location of our private repository.
3. We are tagging the repository name as centos in our private repository.

Step 4 – Now let's use the Docker push command to push the repository to our private repository.

```
>> sudo docker push localhost:5000/centos
```

- Here, we are pushing the centos image to the private repository hosted at localhost:5000.

Step 5 – Now let's delete the local images we have for centos using the docker rmi commands. We can then download the required centos image from our private repository.

```
>>sudo docker rmi centos:latest
```

```
>>sudo docker rmi 67591570dd29
```

Step 6 – Now that we don't have any centos images on our local machine, we can now use the following Docker pull command to pull the centos image from our private repository.

```
>>sudo docker pull localhost:5000/centos
```

- Here, we are pulling the centos image to the private repository hosted at localhost:5000.

- If you now see the images on your system, you will see the centosimage as well.

```
demo@ubuntudemo:~$ sudo docker pull localhost:5000/centos
Using default tag: latest
latest: Pulling from centos
45a2e645736c: Pull complete
Digest: sha256:c577af3197aacedf79c5a204cd7f493c8e07ffbbe7f88f7600bf19c688c38799
Status: Downloaded newer image for localhost:5000/centos:latest
demo@ubuntudemo:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
demoursr/demorep	1.0	ab0c1d3744dd	24 hours ago	225.3 MB
localhost:5000/centos	latest	67591570dd29	3 days ago	191.8 MB
jenkins	latest	ff6f0851ef57	2 weeks ago	714.1 MB
registry	2	c9bd19d022f6	8 weeks ago	33.3 MB

```
demo@ubuntudemo:~$
```

```
Error response from daemon: conflict: unable to delete 5d0da3dc9764 (can
mayank@DESKTOP-4ERPNS5E:/mnt/c/Windows/system32$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	2.4	192d41583429	5 days ago	145MB
python	latest	254fb6647d87	2 weeks ago	921MB
jenkins/jenkins	lts	d5ed2ceef0ec	2 weeks ago	471MB
ubuntu	latest	08d22c0ceb15	2 weeks ago	77.8MB
registry	2	0d153fADF70b	6 weeks ago	24.2MB
openjdk	latest	71260f256d19	6 weeks ago	470MB
hello-world	latest	feb5d9fea6a5	18 months ago	13.3kB
localhost:5000/centos	latest	5d0da3dc9764	18 months ago	231MB

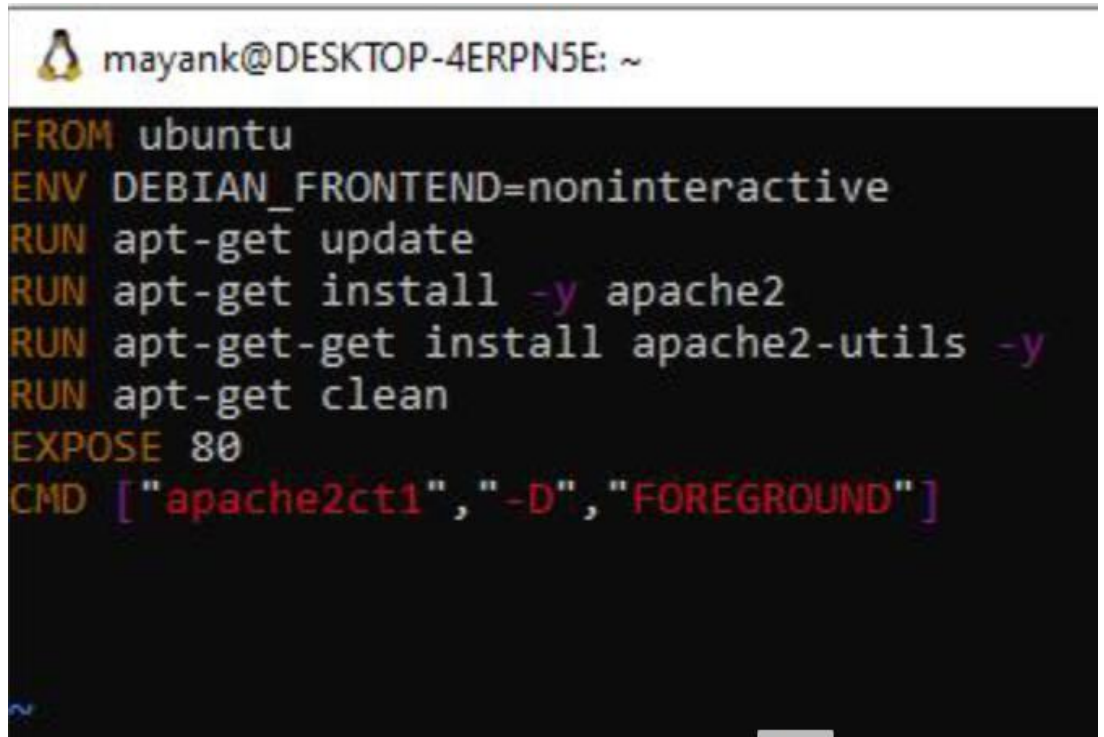
Practical -11

Objective: - Build a web server, run it through a container and access it from DockerHost.

Experiment:

Step -1: The first step is to build Docker file with the help of “vim editor”.

```
>> vim Dockerfile
```



```
FROM ubuntu
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update
RUN apt-get install -y apache2
RUN apt-get-get install apache2-utils -y
RUN apt-get clean
EXPOSE 80
CMD [\"apache2ctl\", \"-D\", \"FOREGROUND\"]
```

- Ubuntu is the base image in which server will be launched.
- In the second line, is to set a non-interactive environment.
- In the third line, the apt-get update command is used to update all the packages on Ubuntu.
- In the fourth line, we are installing apache2 on our image.
- In the fifth line, we are installing all the necessary utility Apache packages.
- In the sixth line, the apt-get clean command cleans all

the unnecessary files from the system.

- In the seventh line, the EXPOSE command is used to expose the port 80 of Apache in the container.
- The last command is used to run apache2 in the background.

Step -2: Next step is to build the docker file by using the docker build command.

```
>> docker build -t mywebserver .
```

After the building of the image is finished, a message will print in the end that the image is build.

- In the seventh line, the EXPOSE command is used to expose the port 80 of Apache in the container.
- The last command is used to run apache2 in the background.

Step -2: Next step is to build the docker file by using the docker build command.

```
>> docker build -t mywebserver .
```

After the building of the image is finished, a message will print in the end that the image is build.

- In the seventh line, the EXPOSE command is used to expose the port 80 of Apache in the container.
- The last command is used to run apache2 in the background.

Step -2: Next step is to build the docker file by using the docker build command.

```
>> docker build -t mywebserver .
```

After the building of the image is finished, a message will print in the end that the image is build.

```
mayank@DESKTOP-4ERPNS5: ~  
mayank@DESKTOP-4ERPNS5:~$ docker build -t mywebserver .  
[+] Building 0.3s (9/9) FINISHED  
=> [internal] load build definition from Dockerfile  
0.0s  
=> => transferring dockerfile: 246B  
0.0s  
=> [internal] load .dockerignore  
0.0s  
=> => transferring context: 2B  
0.0s  
=> [internal] load metadata for docker.io/library/ubuntu:latest  
0.0s  
=> [1/5] FROM docker.io/library/ubuntu  
0.0s  
=> CACHED [2/5] RUN apt-get update  
0.0s  
=> CACHED [3/5] RUN apt-get install apache2 -y  
0.0s  
=> CACHED [4/5] RUN apt-get install apache2-utils -y  
0.0s  
=> CACHED [5/5] RUN apt-get clean  
0.0s  
=> exporting to image  
0.0s  
=> => exporting layers  
0.0s  
=> => writing image sha256:3895bee0e41f8dde3c375c8208b9431bd4024e531fb0fdb3a9aba09910c2793e  
0.0s  
=> => naming to docker.io/library/mywebserver  
0.0s
```

Step 3: - The web server file has been built; the next step is to create a container from the image for that we use the docker run command.

```
>> docker run -d -p 80:80 mywebserver
```

```
mayank@DESKTOP-4ERP5E: ~
=> CACHED [2/5] RUN apt-get update 0.0s
=> CACHED [3/5] RUN apt-get install apache2 -y 0.0s
=> CACHED [4/5] RUN apt-get install apache2-utils -y 0.0s
=> CACHED [5/5] RUN apt-get clean 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:3895bee0e41f8dde3c375c8208b9431bd4024e531fb0fdb3a9aba09910c2793e 0.0s
=> => naming to docker.io/library/mywebserver 0.0s
mayank@DESKTOP-4ERP5E:~$ docker run -d -p 80:80 mywebserver
1478148e976175ed16afe469284317a4643175aca43d38ba47910a515ce52a1b
```

- Commands:

-d: This option is used to run the container in detached mode i.e the container can run in the background

- **-p:** This option is used to map our port number with 5000 portnumbers on our localhost.

Step 4: - Now run the docker images command to see the built image.

>> docker images

```
mayank@DESKTOP-4ERP5E:~$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
mywebserver         latest      3895bee0e41f  27 minutes ago 228MB
myimage             1.0        d14b06aaa4b2  4 hours ago   176MB
mps123456/myreo     1.0        d14b06aaa4b2  4 hours ago   176MB
httpd               2.4        192d41583429  5 days ago    145MB
python              latest      254fb6647d87  2 weeks ago   921MB
jenkins/jenkins     lts        d5ed2ceef0ec  2 weeks ago   471MB
ubuntu              latest      08d22c0ceb15  2 weeks ago   77.8MB
registry            2          0d153fadf70b  6 weeks ago   24.2MB
openjdk             latest      71260f256d19  6 weeks ago   470MB
hello-world         latest      feb5d9fea6a5  18 months ago 13.3kB
localhost:5000/centos latest      5d0da3dc9764  18 months ago 231MB
```

Step 5: - If you go to your web browser and write **localhost_ip:80** your Apache server is up and running on that port.





