Assignment 7- Sharams Kunwar – Docker Container

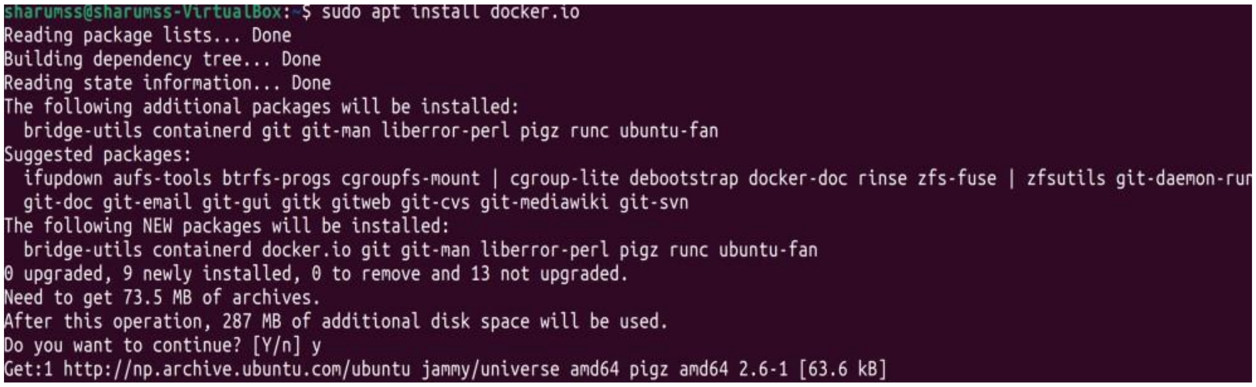
Hands-On:

At first, docker was installed in the system.

Command: sudo apt install docker.io

Command: sudo snap install docker (for other dependencies)

Screenshot:

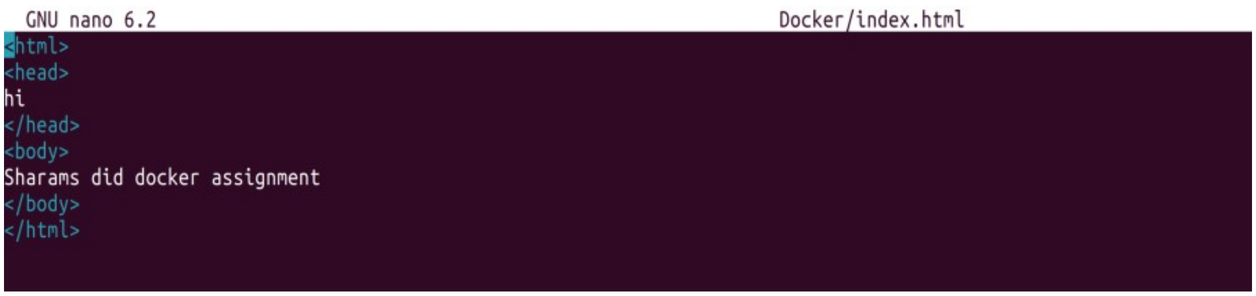


```
sharumss@sharumss-VirtualBox:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils git-daemon-ru
  git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 13 not upgraded.
Need to get 73.5 MB of archives.
After this operation, 287 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
```

Figure 1 Installing Docker

Then a separate directory named Docker was created, inside of which Dockerfile, index.html and index.conf were placed.

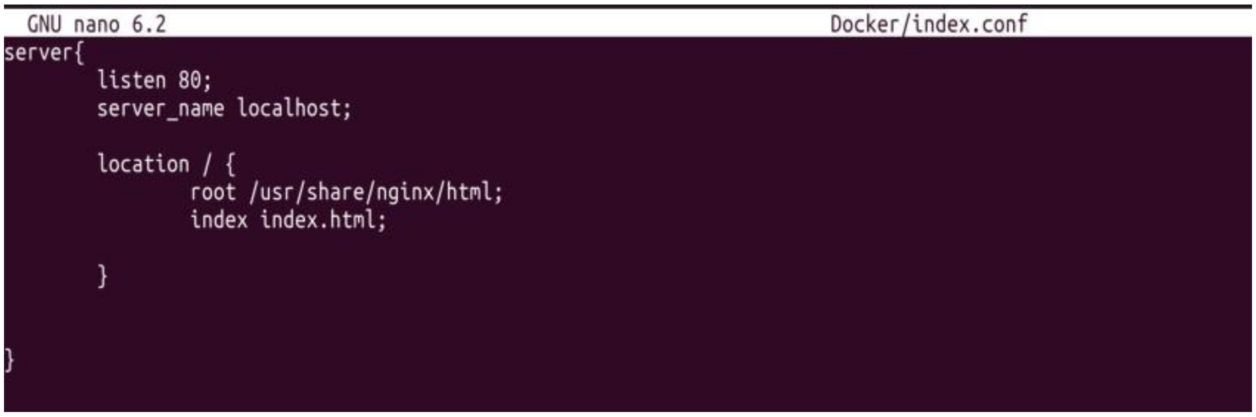
Screenshots:



```
  GNU nano 6.2                                                Docker/index.html
<html>
<head>
hi
</head>
<body>
Sharams did docker assignment
</body>
</html>
```

Figure 2 index.html

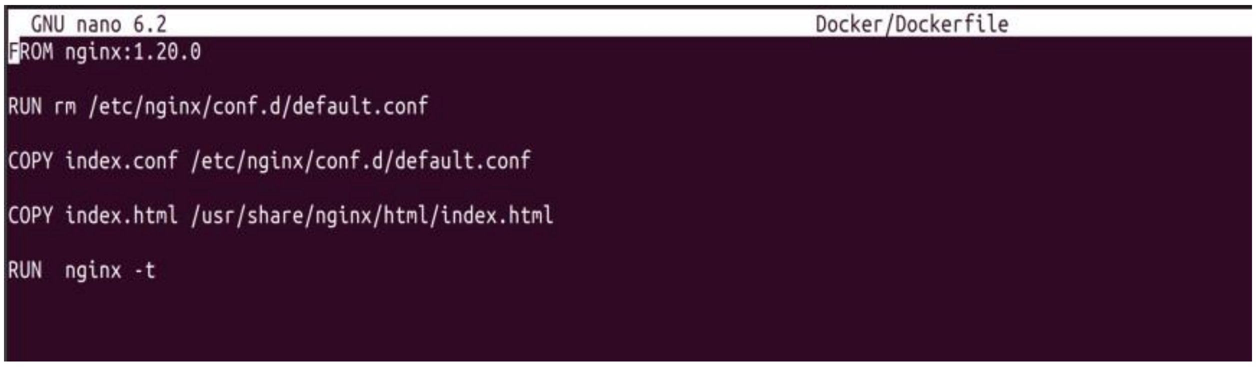
Index.html contains the contents of the landing page, which basically displays message “hi Sharams did docker assignment”.



```
  GNU nano 6.2                                                    Docker/index.conf
server{
        listen 80;
        server_name localhost;

        location / {
                root /usr/share/nginx/html;
                index index.html;

        }


}
```

Figure 3 index.conf

Index.conf file contains the server block configuration for the index.html page that is to be displayed. It specifies the root directory, the landing page, i.e., index.html, the name of the server and the port to listen to.



```
  GNU nano 6.2                                                   Docker/Dockerfile
FROM nginx:1.20.0

RUN rm /etc/nginx/conf.d/default.conf

COPY index.conf /etc/nginx/conf.d/default.conf

COPY index.html /usr/share/nginx/html/index.html

RUN  nginx -t
```

Figure 4 Dockerfile

Here, Dockerfile basically does the following:

- Uses Nginx image as a base.
- Runs the command in the bash of the container to remove default config of nginx.
- Copies index.conf file from the Docker directory and uses it as default.conf instead.
- Copies index.html from the directory to the Nginx default HTML directory.
- Runs ‘nginx -t’ command to verify if the configuration is ok.

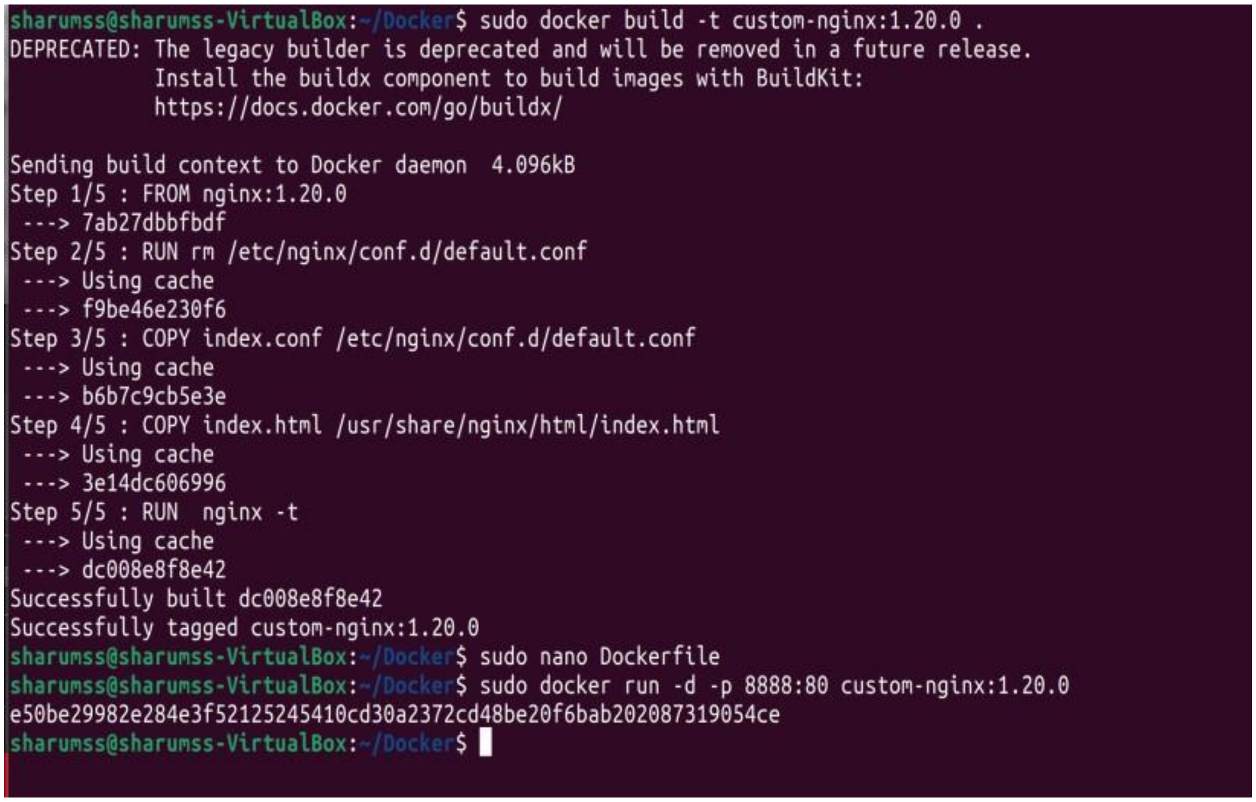
Then, a Docker image named ‘custom-nginx’ of version 1.20.0 was built using the Dockerfile and index.html and index.conf page in the ‘Docker’ directory.

Command: sudo docker build -t custom-nginx:1.20.0 .

Also, after building the image, a container was run from the image in daemon mode. Port 80 inside the container was mapped to port 8888 of the host.

Command: sudo docker run -d -p 8888:80 custom-nginx:1.20.0

Screenshot:



```
sharumss@sharumss-VirtualBox:~/Docker$ sudo docker build -t custom-nginx:1.20.0 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  4.096kB
Step 1/5 : FROM nginx:1.20.0
 ---> 7ab27dbbfbdf
Step 2/5 : RUN rm /etc/nginx/conf.d/default.conf
 ---> Using cache
 ---> f9be46e230f6
Step 3/5 : COPY index.conf /etc/nginx/conf.d/default.conf
 ---> Using cache
 ---> b6b7c9cb5e3e
Step 4/5 : COPY index.html /usr/share/nginx/html/index.html
 ---> Using cache
 ---> 3e14dc606996
Step 5/5 : RUN  nginx -t
 ---> Using cache
 ---> dc008e8f8e42
Successfully built dc008e8f8e42
Successfully tagged custom-nginx:1.20.0
sharumss@sharumss-VirtualBox:~/Docker$ sudo nano Dockerfile
sharumss@sharumss-VirtualBox:~/Docker$ sudo docker run -d -p 8888:80 custom-nginx:1.20.0
e50be29982e284e3f52125245410cd30a2372cd48be20f6bab202087319054ce
sharumss@sharumss-VirtualBox:~/Docker$ 
```

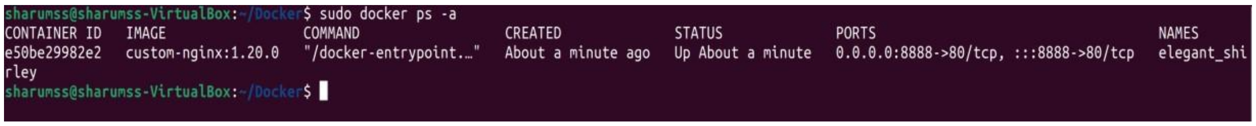
Figure 5 Building Image and Running Container from it

Then, the status of container was checked to see if it was really running in the background as specified earlier.

Command: sudo docker ps -a

The container had been successfully created and it was running too.

Screenshot:

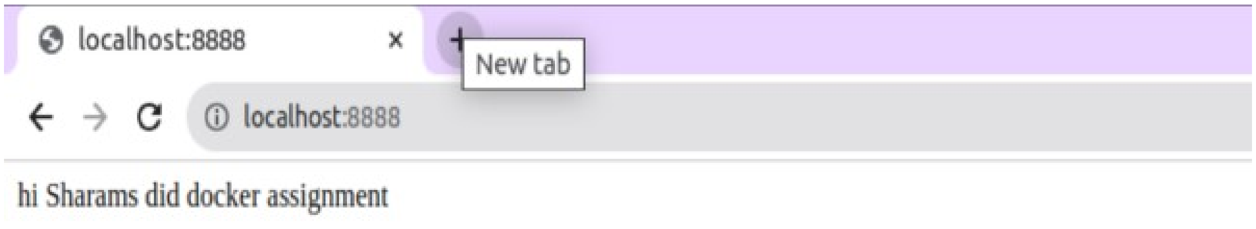


```
sharumss@sharumss-VirtualBox:~/Docker$ sudo docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED              STATUS              PORTS                                   NAMES
e50be29982e2   custom-nginx:1.20.0   "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:8888->80/tcp, :::8888->80/tcp   elegant_shi
rley
sharumss@sharumss-VirtualBox:~/Docker$ 
```

Figure 6 Status of Container

Then, the custom NginX container was accessed in the web browser of the host by navigating to http://localhost:8888.

Screenshot:



hi Sharams did docker assignment

Figure 7 Final Result

Key Concepts

- **Docker Hub**

Docker Hub is a central remote repository hosted on a cloud-based platform for container images. It's basically a GitHub for Docker Images allowing developers to store, share, and distribute Docker Container Images. It provides similar features to that of GitHub. Few of them are listed below:

- Image Repo
- Public and Private Repo
- Version Control
- Allows Collaboration
- Automated Builds
- Webhooks
- Integration with Docker Compose/ Docker Swarm
- Official Images

- **Basic Docker Commands**

- Inspecting a container
  Command: docker inspect container_name_or_id
  Function: This command provides detailed information about a Docker container, including its configuration, network settings, environment variables, and more.
  Use: To troubleshoot container issues and examine container's metadata.

- Removing Docker Images
  Command: docker rmi image_name_or_id
  Function: This command is used to delete Docker images from your local image repository.
  Use: To manage disk space by removing outdated/ unused images.

- Managing Docker Services
  Commands: docker service ls / docker service scale service_name=desired_replicas
  Function: To create, list, scale, update, and remove services within a Docker Swarm cluster.
  Use: To manage services in a Swarm cluster. Services are used to define and manage long-running, scalable applications.

- Tagging Docker Images
  Command: docker tag source_image target_image:tag
  Function: This command is used to assign a name and optionally a version (tag) to a Docker image.
  Use: Tagging images is essential for versioning and naming images in a way that's more meaningful and identifiable.

- Building Docker Images:
  Command: docker build -t image_name:tag .
  Function: The docker build command is used to create a Docker image from a Dockerfile.
  Use: To package applications and their dependencies into a container image that can be run consistently across different environments.