

Name: Om Sanjay Bhavsar

Roll No: 331069

GR No: 22120084

Div: A

Batch: A3

Assignment No. 7

Aim:

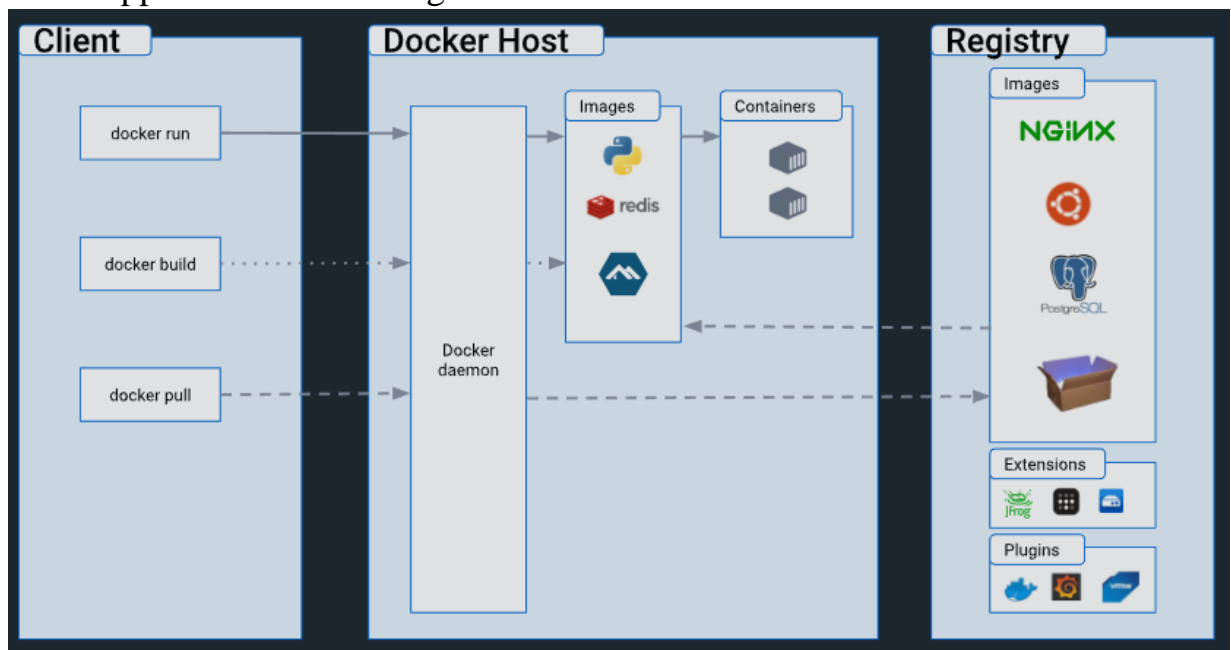
Deploy a web application using Docker.

Theory:

- What is Docker :
 1. Docker is an open platform for developing, shipping, and running applications.
 2. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
 3. With Docker, you can manage your infrastructure in the same ways you manage your applications.
 4. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.
 5. Docker provides the ability to package and run an application in a loosely isolated environment called a container.
 6. The isolation and security allows you to run many containers simultaneously on a given host.

- Docker Architecture:

1. Docker uses a client-server architecture.
2. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.
3. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.
4. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.
5. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.



- Difference between Docker and Virtual machine:

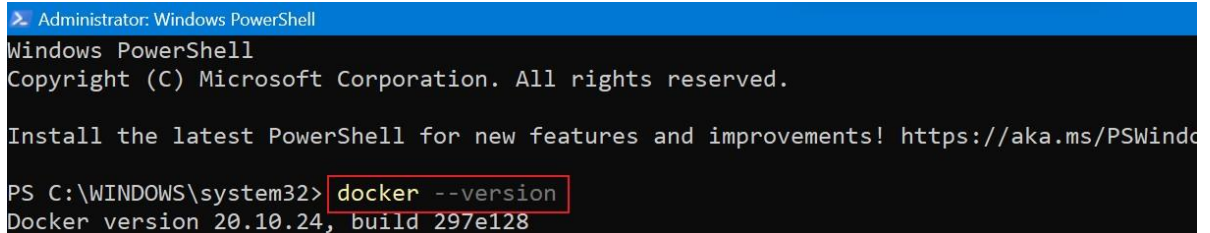
	Docker	Virtual Machine[VM]
Boot-Time	Boots in a few seconds.	It takes a few minutes for VMs to boot.
Runs on	Dockers make use of the execution engine.	VMs make use of the hypervisor.
Memory Efficiency	No space is needed to virtualize, hence less memory.	Requires entire OS to be loaded before starting the surface, so less efficient.

Isolation	Prone to adversities as no provisions for isolation systems.	Interference possibility is minimum because of the efficient isolation mechanism.
Deployment	Deploying is easy as only a single image, containerized can be used across all platforms.	Deployment is comparatively lengthy as separate instances are responsible for execution.
Usage	Docker has a complex usage mechanism consisting of both third party and docker managed tools.	Tools are easy to use and simpler to work with.

- Docker Commands:

1. `docker --version:`

This command is used to get the currently installed version of docker.



```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

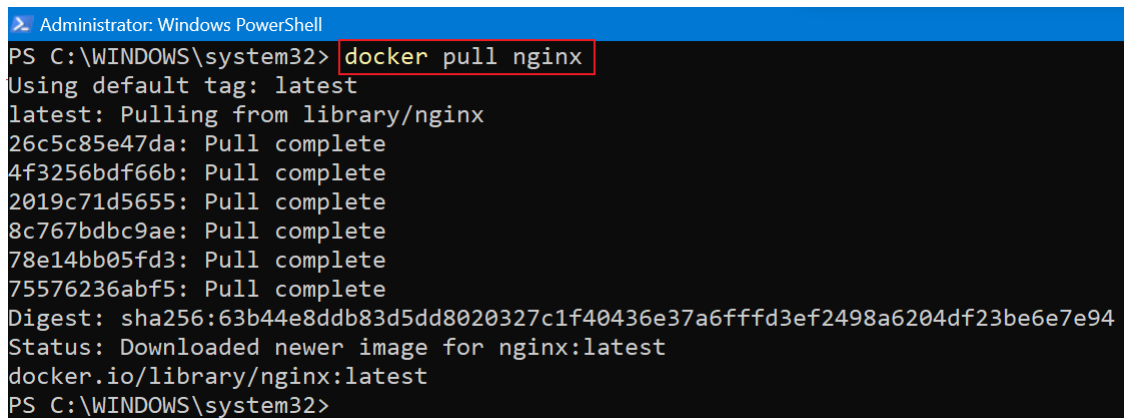
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> docker --version
Docker version 20.10.24, build 297e128

```

2. `docker pull:`

This command is used to pull images from the docker repository (hub.docker.com)



```

Administrator: Windows PowerShell
PS C:\WINDOWS\system32> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
26c5c85e47da: Pull complete
4f3256bdf66b: Pull complete
2019c71d5655: Pull complete
8c767bdbc9ae: Pull complete
78e14bb05fd3: Pull complete
75576236abf5: Pull complete
Digest: sha256:63b44e8ddb83d5dd8020327c1f40436e37a6fffd3ef2498a6204df23be6e7e94
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
PS C:\WINDOWS\system32>

```

3. docker run:

Usage: docker run -it -d <image name>

This command is used to create a container from an image

```
PS C:\app> docker run -it -d nginx
fd4cacddd918c4a3c53188039c8469d3dc1d3d92d70f911245e4795e47972
PS C:\app>
```

4. docker ps:

This command is used to list the running containers

```
PS C:\app> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                    NAMES
fd4cacddd918   nginx     "/docker-entrypoint..." 57 seconds ago Up 55 seconds  80/tcp                festive_ellis
3a44a3020044   nginx     "/docker-entrypoint..." About an hour ago Up About an hour  0.0.0.0:8001->80/tcp    web-site
```

5. docker exec:

Usage: docker exec -it <container id> bash

This command is used to access the running container

```
PS C:\app> docker exec -it fd4cacddd918 bash
root@fd4cacddd918:/#
```

6. docker stop:

Usage: docker stop <container id>

This command stops a running container

```
PS C:\app> docker stop fd4cacddd918
fd4cacddd918
PS C:\app>
```

• Dockerfile:

1. Docker can build images automatically by reading the instructions from a Dockerfile.
2. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
3. Format:
Comment
INSTRUCTION arguments
4. The instruction is not case-sensitive.
5. However, convention is for them to be UPPERCASE to distinguish them from arguments more easily.

7. Docker runs instructions in a Dockerfile in order.
8. A Dockerfile must begin with a FROM instruction. This may be after parser directives, comments, and globally scoped ARGs.
9. The FROM instruction specifies the Parent Image from which you are building.

- Docker-Compose:

1. Compose is a tool for defining and running multi-container Docker applications.
2. With Compose, you use a YAML file to configure your application's services.
3. Then, with a single command, you create and start all the services from your configuration.
4. Compose works in all environments: production, staging, development, testing, as well as CI workflows.
5. It also has commands for managing the whole lifecycle of your application:
 - Start, stop, and rebuild services
 - View the status of running services
 - Stream the log output of running services
 - Run a one-off command on a service

- Docker-Swarm:

1. A Docker Swarm is a group of either physical or virtual machines that are running the Docker application and that have been configured to join together in a cluster.
2. The activities of the cluster are controlled by a swarm manager, and machines that have joined the cluster are referred to as nodes.
3. One of the key benefits associated with the operation of a docker swarm is the high level of availability offered for applications.
4. Docker Swarm lets you connect containers to multiple hosts similar to Kubernetes.
5. Docker Swarm has two types of services: replicated and global.

Implementation:

Step 1: Install nginx on windows follow the link:

<http://nginx.org/en/docs/windows.html>

unzip the downloaded file in the drive you want.

Goto cmd and perform foll commands:

```
Command Prompt
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shrav>cd c:/

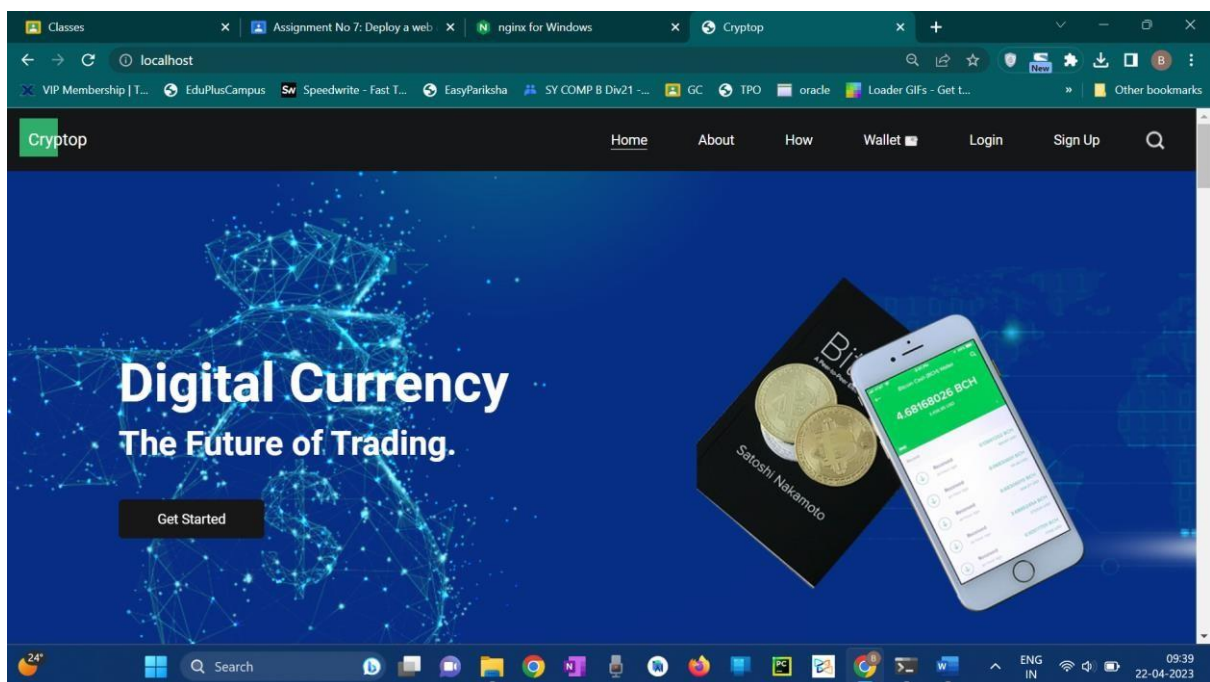
c:\>cd nginx-1.24.0

c:\nginx-1.24.0>start nginx

c:\nginx-1.24.0>
```

Step 2: Copy the sample-website in “C:\nginx\html\” folder. Step

3: open browser and run “localhost:80”



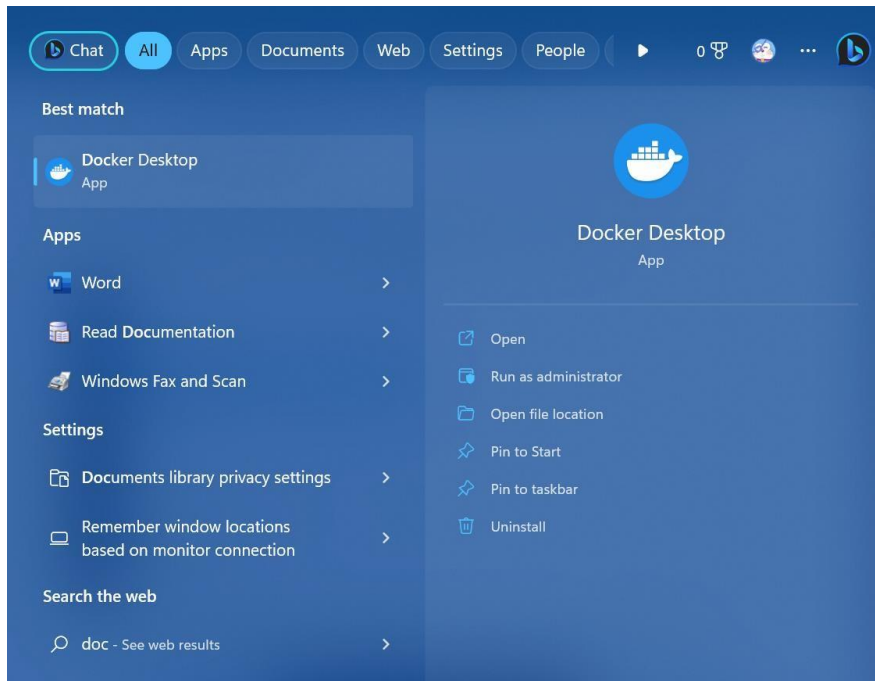
Step 4: Download Docker for windows, follow the link :

<https://docs.docker.com/desktop/install/windows-install/>

Step 5: Start Docker Desktop

Docker Desktop does not start automatically after installation. To start Docker Desktop:

1. Search for Docker, and select Docker Desktop in the search results.



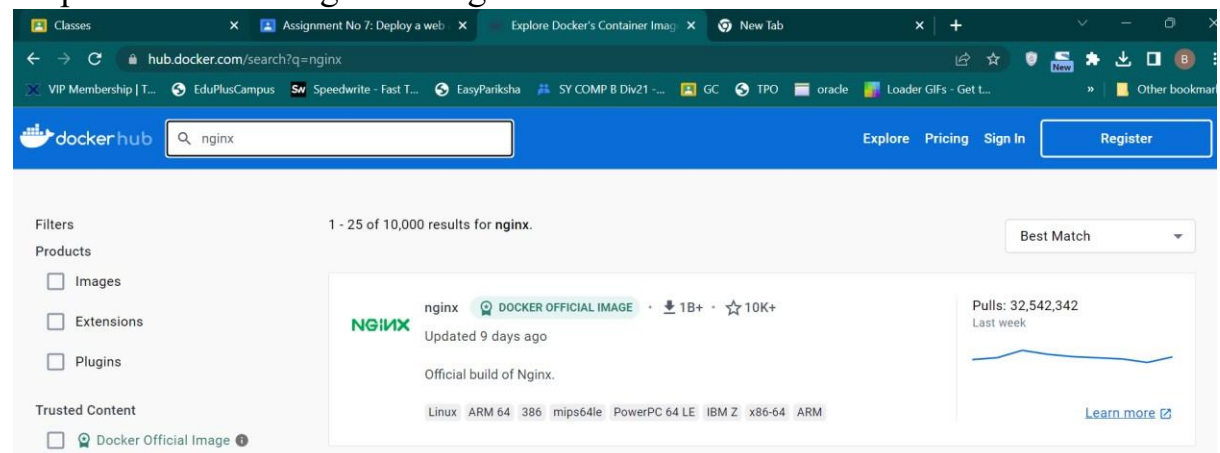
Step 6: Open Powershell and check Docker installation using commands:

a. `docker --version`

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> docker --version
Docker version 20.10.24, build 297e128
```



Step 3: pull the latest image of nginx using command:
docker pull nginx

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
26c5c85e47da: Pull complete
4f3256bdf66b: Pull complete
2019c71d5655: Pull complete
8c767bdbc9ae: Pull complete
78e14bb05fd3: Pull complete
75576236abf5: Pull complete
Digest: sha256:63b44e8ddb83d5dd8020327c1f40436e37a6fffd3ef2498a6204df23be6e7e94
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
PS C:\WINDOWS\system32>
```

Step 4: check the docker images on your desktop by using
command:
docker images

```
PS C:\WINDOWS\system32> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx         latest    6efc10a0510f   9 days ago    142MB
PS C:\WINDOWS\system32>
```

Step 5: go in the “SampleWebsite” folder and then Create a container using the docker command and sync the “SampleWebsite” folder with folder inside the container folder. (This is called Mount Bind”) “docker run -d -p 8001:80 -v \${PWD}:/usr/share/nginx/html --name web-site nginx”

```
PS C:\sampleWebsite> docker run -d -p 8001:80 -v ${PWD}:/usr/share/nginx/html --name web-site nginx
3a44a3020044a92c87350710e0f359fdec145afa21d20b6d3f83e458dbf65a71
PS C:\sampleWebsite> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                   NAMES
3a44a3020044   nginx    "/docker-entrypoint...." 2 minutes ago Up 2 minutes   0.0.0.0:8001->80/tcp    web-site
PS C:\sampleWebsite>
```

Step 6: verify the website open browser and check “localhost:8001”.
Now this website is running inside your container.



□ DockerFile:

Step 1: Create a Directory structure like

App :

1. sampleWebsite
2. Dockerfile

Step 2: Write a following script into “Dockerfile”

```
FROM nginx:latest
```

```
COPY ./sampleWebSite/ /usr/share/nginx/html/
```

```
EXPOSE 80
```

Step 3: build image from docker file using command:

```
docker build -t my-app:v1 .
```

```
PS C:\app> docker build -t my-app:v1 .
[+] Building 0.5s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 107B
=> [internal] load .dockerignore
=> => transferring context: 28
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load build context
=> => transferring context: 30B
=> CACHED [1/2] FROM docker.io/library/nginx:latest
=> [2/2] COPY ./sampleWeb/ /usr/share/nginx/html/
=> exporting to image
=> => exporting layers
=> => writing image sha256:7815fc1083b74bb900e4280bd4c5db364e307fa3773f0872494b245661ac5f03
=> => naming to docker.io/library/my-app:v1
```

Step 4: check images using command:

docker images

```
PS C:\app> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-app	v1	7815fc1083b7	14 seconds ago	142MB
nginx	latest	6efc10a0510f	9 days ago	142MB

☐ PUSH Image to “DockerHub”

Step 1: if you do not have account first go to the website and sign up.

login to docker hub using command :

```
PS C:\app> docker login -u shravanigadre
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/
```

Step 2: docker images

```
PS C:\app> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-app	v1	7815fc1083b7	14 seconds ago	142MB
nginx	latest	6efc10a0510f	9 days ago	142MB

Step 3: docker tag (old image name) ombhavsar/newname docker tag

my-web:v1 ombhavsar/newapp

```
PS C:\app> docker tag my-app:v1 shravanigadre/my-app:v1
```

```
PS C:\app> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-app	v1	7815fc1083b7	5 minutes ago	142MB
shravanigadre/my-app	v1	7815fc1083b7	5 minutes ago	142MB
nginx	latest	6efc10a0510f	9 days ago	142MB

Step 4: docker push shravanigadre/newapp:

```
PS C:\app> docker push shravanigadre/my-app:v1
The push refers to repository [docker.io/shravanigadre/my-app]
789ac392b2f3: Pushed
9d907f11dc74: Mounted from library/nginx
79974a1a12aa: Mounted from library/nginx
f12d4345b7f3: Mounted from library/nginx
935b5bd454e1: Mounted from library/nginx
fb6d57d46ad5: Mounted from library/nginx
ed7b0ef3bf5b: Mounted from library/nginx
v1: digest: sha256:e391d575b68b1849db3b5d4e5ead7b8c927e2ebe9b7aaf698f11121575827c2a size: 1777
PS C:\app>
```

Step 5: Login to Docker Hub and check the repository

