

CLOUD COMPUTING AND DEVOPS

Rohan Tikotekar

VIIT IT C C3

RollNo.333056

PRN.22010060

Assignment 7: Deploy a web application using Docker

Theory:

1. What is docker
 - Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
 - With Docker, you can manage your infrastructure in the same ways you manage your applications. 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.
 - Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allows you to run many containers simultaneously on a given host.
 - Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.
2. Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.
3. Docker client uses commands and REST APIs to communicate with the Docker Daemon(Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

Docker Registry: Docker Registry manages and stores the Docker images. There are two types of registries in the Docker - Public Registry – Public Registry is also called as Docker hub. Private Registry - It is used to share images within the enterprise.

Docker Objects: There are the following Docker Objects –

Docker Images: Docker images are the read-only binary templates used to create Docker Containers. It uses a private container registry to share container images within the enterprise and also uses public container registry to share container images within the whole world. Metadata is also used by Docker images to describe the container's abilities

VM vs Docker

Features	VM (Virtual Machines)	Docker
Boot-Time	VM boots in a few minutes.	Docker takes a few seconds to boot.
Runs on	A virtual machine uses a hypervisor.	Dockers use an execution engine.
Memory Efficiency	It is less efficient because it requires the whole operating system to be loaded before beginning the surface.	No space will be required for virtualization, so less memory.
Isolation	Interference possibility will be minimum because of its isolation mechanism.	Dockers are prone to adversities. No provisions for many isolation systems.
Deployment	VM contains lengthy deployment because its isolated instances are liable for execution.	Docker contains easy deployment because of an individual image. It is containerized and can be applied beyond each platform.
Usage	A virtual machine has tools that are simpler and easy-to-use to implement.	Docker has a convoluted usage mechanism. It consists of Docker managed tools and third party both.
OS support	All virtual machines have an isolated OS.	All the containers can distribute OS.
Storage	It requires a few GBs.	Its container is lightweight (MBs/KBs).
Availability	Ready-made virtual machines are available but complex to find.	Pre-built containers of Docker are available.
Resource Usage	More usage of resources.	Less usage of resources.
Creation Time	Creating a virtual machine will take a longer time relatively.	The container of the Docker can be made in seconds.

Dockerfile uses DSL (Domain Specific Language) and contains instructions for generating a Docker image. Dockerfile will define the processes to quickly produce an image. While creating your application, you should create a Dockerfile in order since the Docker daemon runs all of the instructions from top to bottom.

Steps To Create a Dockerfile

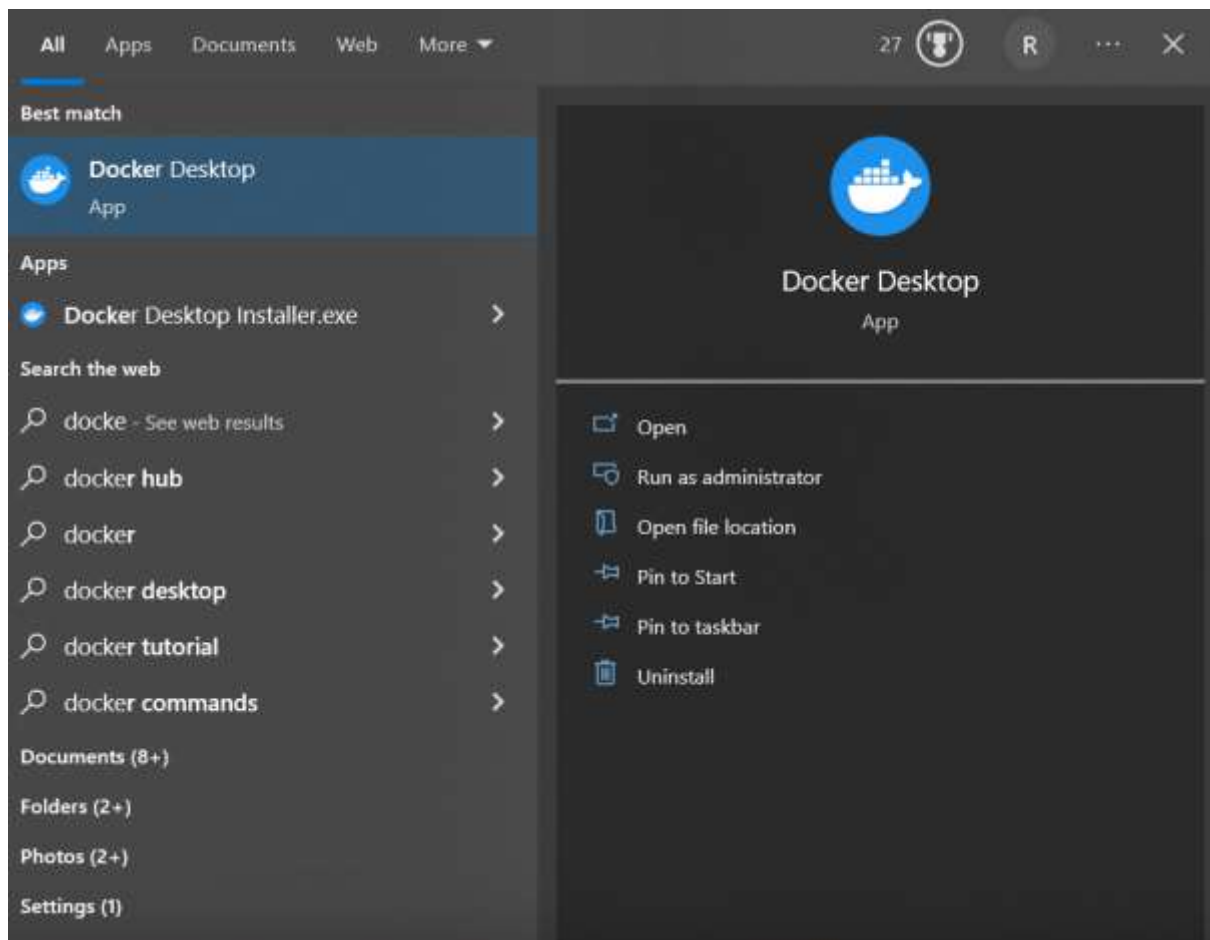
- Create a file named Dockerfile.
- Add instructions in Dockerfile.
- Build Dockerfile to create an image.
- Run the image to create a container.

Docker file instructions:

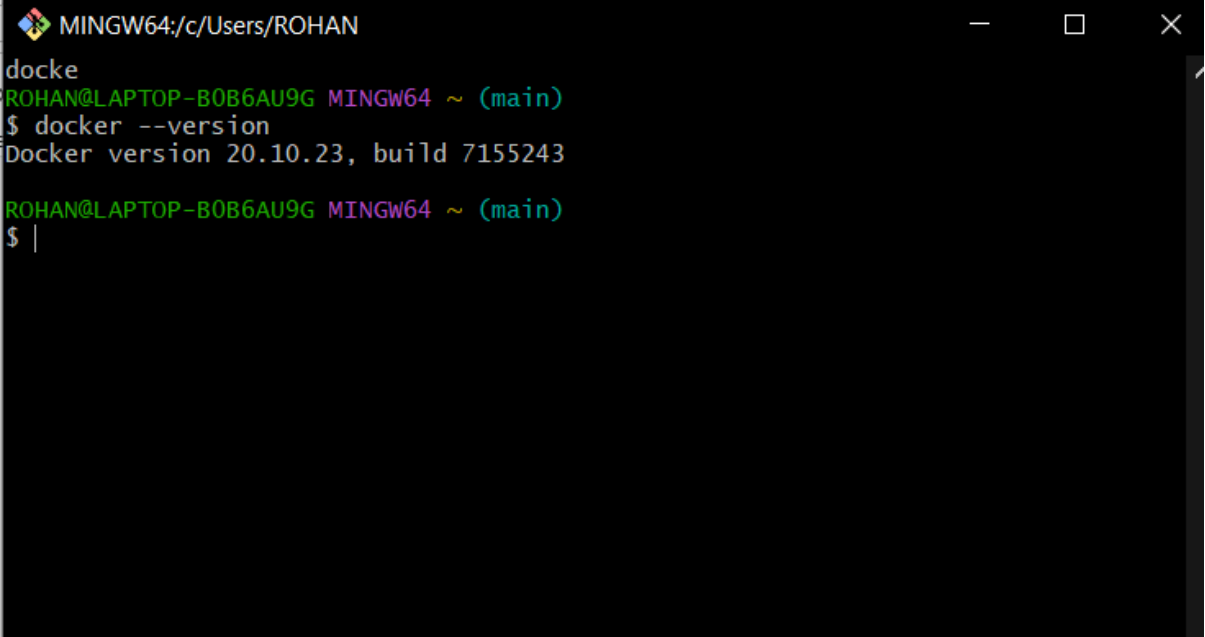
- FROM
- COPY
- ADD
- RUN
- CMD
- ENTRYPOINT
- MAINTAINER

Implementation:

1. Install nginx
<http://nginx.org/en/docs/windows.html> cd c:\ unzip nginx-1.23.4.zip
cd nginx-1.23.4
start nginx
2. Step 2: Copy the sample-website in "C:\nginx\html\" folderStep
3. Start docker desktop

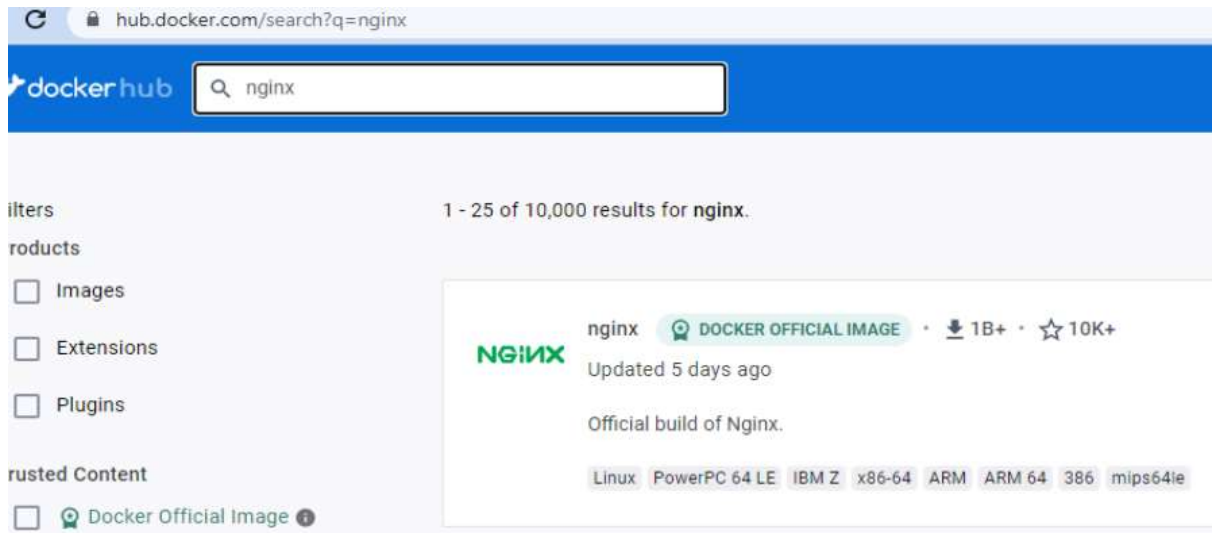


4. Check docker version

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/ROHAN'. The terminal shows a user named ROHAN@LAPTOP-B0B6AU9G in a MINGW64 environment. The user has entered the command 'docker --version' and the output is 'Docker version 20.10.23, build 7155243'. The prompt '\$ |' is visible at the bottom.

```
MINGW64:/c/Users/ROHAN
docks
ROHAN@LAPTOP-B0B6AU9G MINGW64 ~ (main)
$ docker --version
Docker version 20.10.23, build 7155243
ROHAN@LAPTOP-B0B6AU9G MINGW64 ~ (main)
$ |
```

5. Steps to run the "Sample website" in Docker container
- Step 1) visit to Docker hub web site: <https://hub.docker.com/>
- Step 2) search for "nginx" image on site



Step 3) pull the latest image of nginx using command

"docker pull nginx"

DockerFile

Step 1) Create a Directory structure like App SampleWebSite eDockerfile

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:\Users\ROHAN\OneDrive\Desktop\django-app>
* History restored

PS C:\Users\ROHAN\OneDrive\Desktop\django-app> docker build -t trohan37/myapp21:latest .
```

Step 4: Docker images

```
PS C:\Users\ROHAN\OneDrive\Desktop\django-app> docker images
```

REPOSITORY	IMAGE ID	CREATED	SIZE	TAG
trohan37/myapp21	3cc2c99d46e5	28 hours ago	215MB	latest
trohan37/myapp20	dbd5e08df90b	28 hours ago	215MB	latest
trohan37/myapp17	416ea1656265	28 hours ago	215MB	latest
trohan37/myapp18	416ea1656265	28 hours ago	215MB	latest
trohan37/myapp19	416ea1656265	28 hours ago	215MB	latest

Step 5) docker tag (old image name) username/newname ie

Docker -t myapp trohan37/myapp2

Step 6) docker push

```
PS C:\Users\ROHAN\OneDrive\Desktop\django-app> docker push trohan37/myapp
Using default tag: latest
The push refers to repository [docker.io/trohan37/myapp]
62b0f1268f41: Preparing
80a876b0c8f3: Preparing
2c22c685b034: Preparing
0a38cdb6aa81: Preparing
b873ccfbfbf6: Preparing
8bed4fbf969e: Waiting
696d3caf4765: Waiting
8e90761c511e: Waiting
75b6abbcdfde: Waiting
559114740b17: Waiting
3ae97ebd75ca: Waiting
```