Experiment 09 - Docker Setup

Roll No.	24
Name	Iyer Sreekesh Subramanian
Class	D15-A
Subject	DevOps Lab
LO Mapped	LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements LO5: To understand the concept of containerization and Analyze the Containerization of OS images and deployment of applications over Docker

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

Introduction:

(Introduction to Docker)

What is Docker?

Docker is a Linux-based, open-source containerization platform that developers use to build, run, and package applications for deployment using containers. Unlike virtual machines, Docker containers offer:

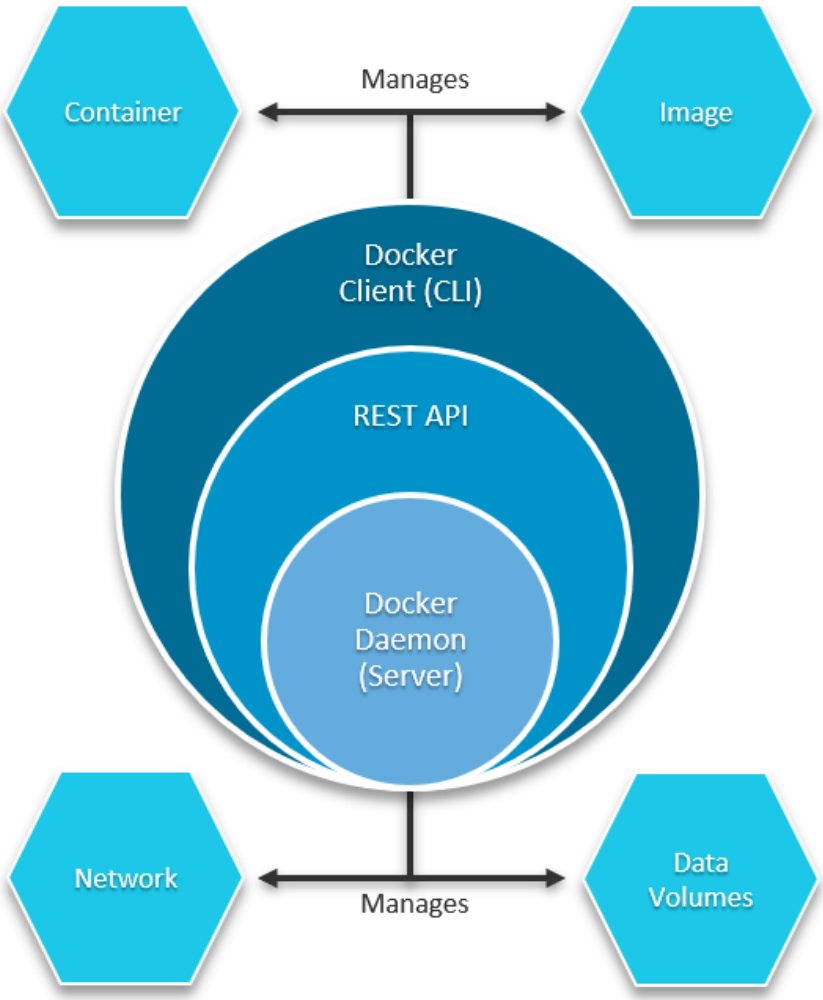
- ➔ OS-level abstraction with optimum resource utilization
- ➔ Interoperability
- ➔ Efficient build and test
- ➔ Faster application execution
- ➔ Fundamentally, Docker containers modularize an application's functionality into multiple components that allow deploying, testing, or scaling them independently when needed.

Take, for instance, a Docker containerized database of an application. With such a framework, you can scale or maintain the database independently from other modules/components of the application without impacting the workloads of other critical systems.

Components of a Docker architecture

Docker comprises the following different components within its core architecture:

- Images
- Containers
- Registries
- Docker Engine



Images

Images are like blueprints containing instructions for creating a Docker container. Images define:

- ➔ Application dependencies
- ➔ The processes that should run when the application launches

You can get images from DockerHub or create your own images by including specific instructions within a file called Dockerfile.

Containers

Containers are live instances of images on which an application or its independent modules are run.

In an object-oriented programming analogy, an image is a class and the container is an instance of that class. This allows operational efficiency by allowing you to multiple containers from a single image.

Registries

A Docker registry is like a repository of images.

The default registry is the Docker Hub, a public registry that stores public and official images for different languages and platforms. By default, a request for an image from Docker is searched within the Docker Hub registry.

You can also own a private registry and configure it to be the default source of images for your custom requirements.

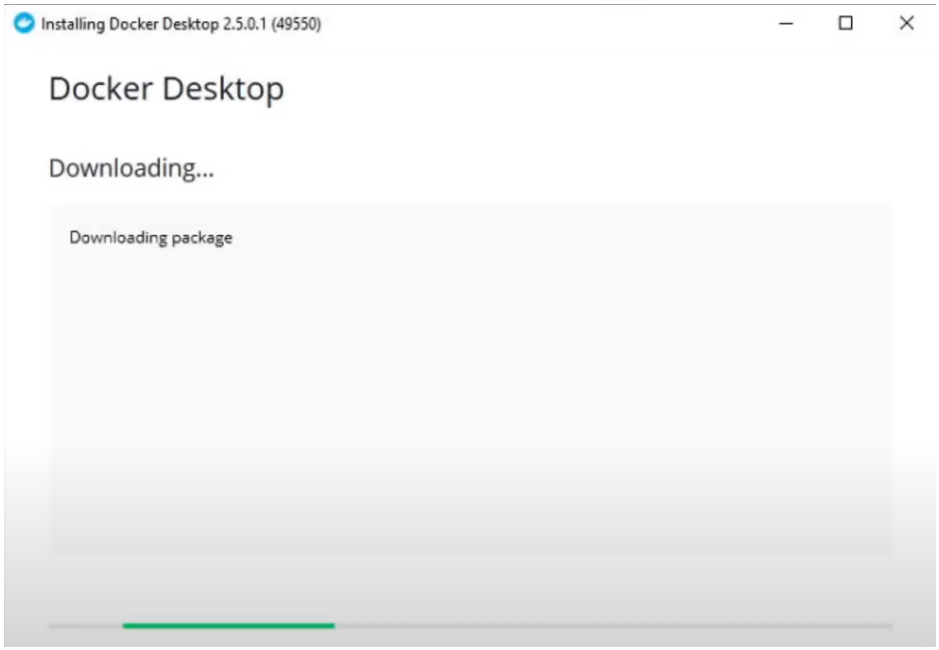
Installation:

(Explain the steps used with screenshots)

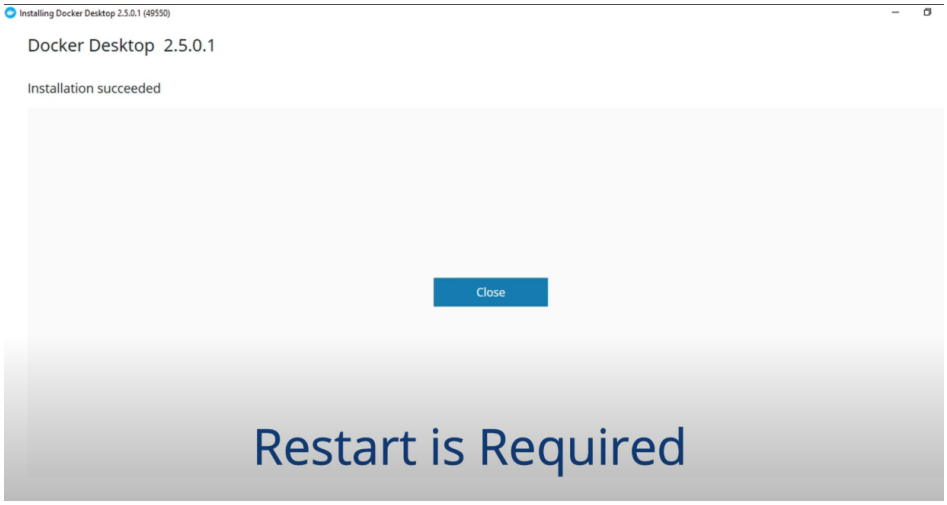
Steps:

1. Open up the Docker website and click on the Download button to download Docker Desktop for your Operating System.

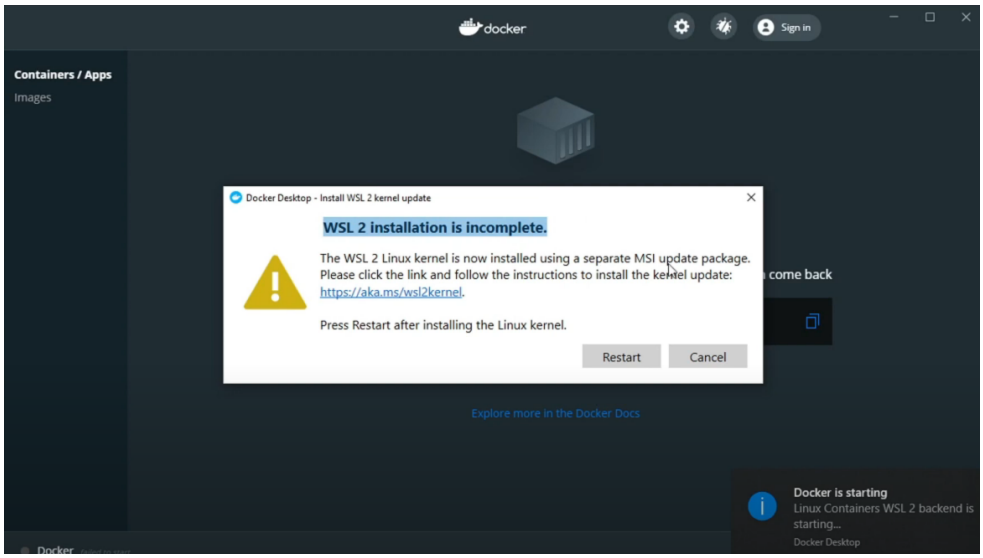
2. Open the Installer and wait for Docker to download its prerequisites.



3. Restart your machine once the install is complete.

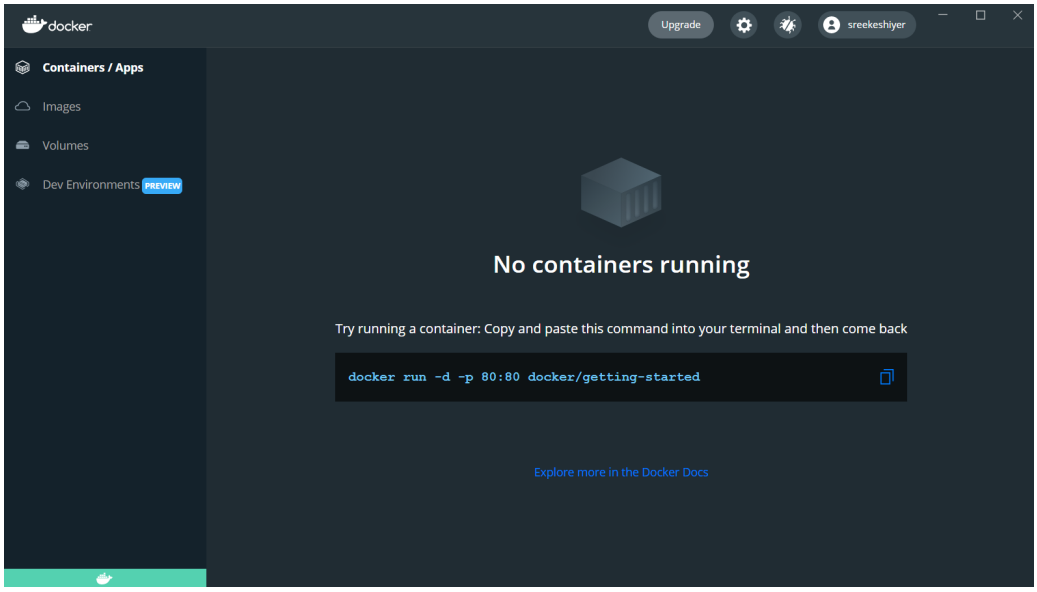


4. Start Docker Desktop.



Download the required WSL2 kernel and restart your machine to complete the installation if prompted.

5. With that, Docker Desktop is ready.



Docker Commands:

(Explain commands and execute them. Display the output with screenshots)

1. Checking Docker Version

```
docker version
docker info
```

```
→ ~  docker version
Client:
 Cloud integration: 1.0.17
 Version:           20.10.8
 API version:       1.41
 Go version:        go1.16.6
 Git commit:        3967b7d
 Built:             Fri Jul 30 19:58:50 2021
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.8
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.16.6
  Git commit:       75249d8
  Built:            Fri Jul 30 19:52:31 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.9
  GitCommit:        e25210fe30a0a703442421b0f60afac609f950a3
 runc:
  Version:          1.0.1
  GitCommit:        v1.0.1-0-g4144b63
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
→ ~
```

```
→ ~  docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Build with BuildKit (Docker Inc., v0.6.3)
  compose: Docker Compose (Docker Inc., v2.0.0)
  scan: Docker Scan (Docker Inc., v0.8.0)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 20.10.8
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentri
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
 Default Runtime: runc
 Init Binary: docker-init
```

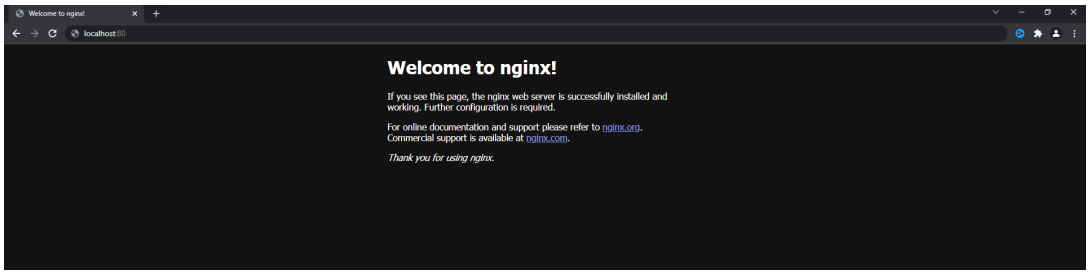
2. Starting a Container from an Image

We can run Docker Containers from a pre-existing image or docker will pull the specified image from the Docker hub.
For this example, we will run an nginx server in a docker container on port 80.

```
docker container run --publish 80:80 -d nginx
```

```
→ ~  docker container run --publish 80:80 -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
b380bbd43752: Pull complete
fca7e12d1754: Pull complete
745ab57616cb: Pull complete
a4723e260b6f: Pull complete
1c84ebdff681: Pull complete
858292fd2e56: Pull complete
Digest: sha256:644a70516a26004c97d0d85c7fe1d0c3a67ea8ab7ddf4aff193d9f301670cf36
Status: Downloaded newer image for nginx:latest
9c6e865aabfa62eaa3c4b1087b5abb80156267d89c5a736c1f1e0c3e70d75c1d
→ ~
```

On your browser, open up localhost:80 and check to see if the nginx server is up.



3. Listing out Containers

We can find a list of running containers on our machine.

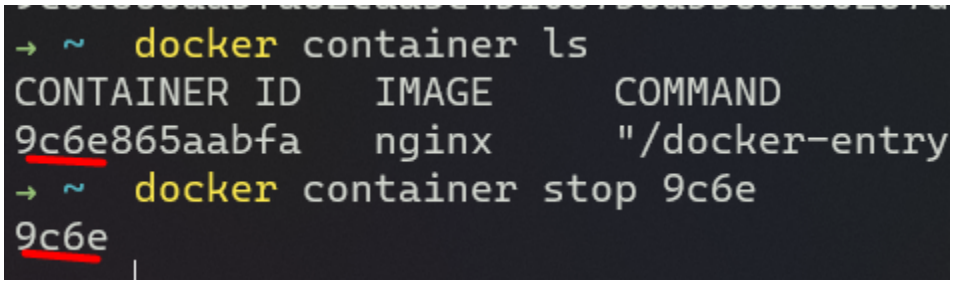
```
docker container ls
```

```
→ ~  docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                NAMES
9c6e865aabfa   nginx     "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   0.0.0.0:80->80/tcp   eager_sammet
```

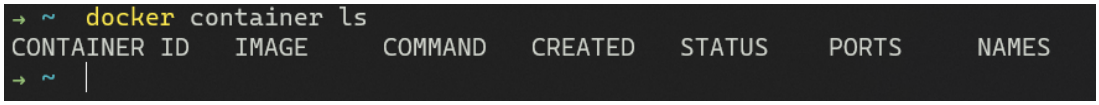
4. Stopping a Container

   We can stop a running container using the docker container stop command by providing the id we found above.

   We only need to provide the initial few letters of the id, until it's totally unique.
   `docker container stop <id>`



   If we now run docker container ls we get an empty response.



5. Listing out All Containers

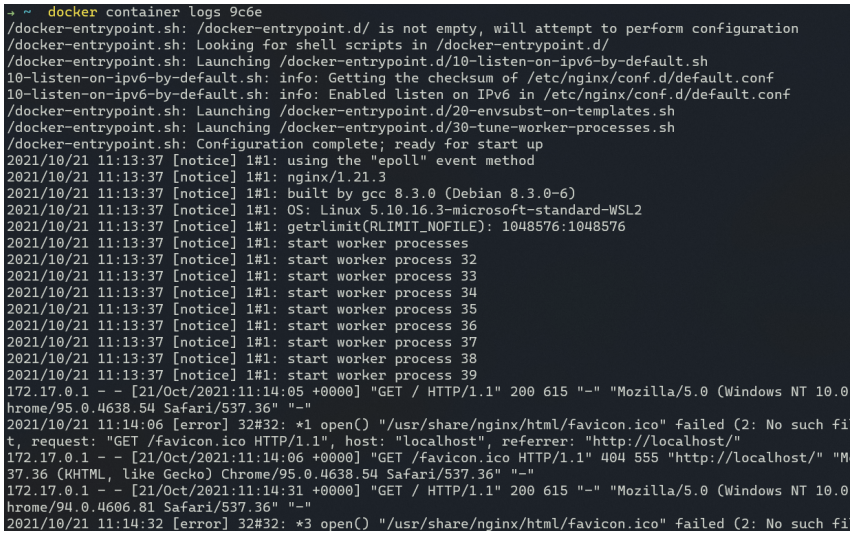
   We can use the -a flag to the previous list command to find a list of all containers, even those which have stopped.

   `docker container ls -a`

6. Show container logs

   We can use the command logs to show logs for a specified container -

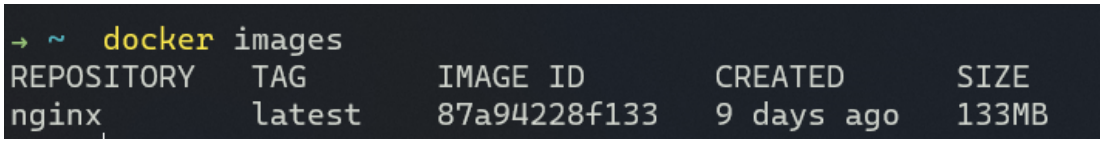
   ```
   docker container logs <id>
   ```



7. Listing out Images

   We can use the docker images command to show a list of docker images we locally have.

   ```
   docker images
   ```



Conclusion

Thus, we learned how to install Docker on our machines and use basic Docker commands using the CLI to create, run and stop Docker Containers.