**Implementation steps**:

## DOCKER BASIC COMMANDS:

# Create an image from Dockerfile in the current Dir:

"$ docker build -t nihdemo1 . "

# run a container from image:

"$ docker run --name nihdemo1 -p 8080:5000 nihdemo1 "

\*\* Port (docker) 5000 forwarded to port (local machine) 8080

\*\* “ host\_post:container\_port “

# Port forwading / which local port it opens at.

# remove a container:

"$ docker rm /nihdemo1 "

## Docker COMPOSE BASIC COMMANDS:

Check if docker compose installed:

“$ docker-compose version ”

Run compose. We’ll add the -d flag to run everything in the background.

“$ docker-compose up -d ”

Provide the yml file name using -f:

“$ docker-compose -f Docker-compose\_old\_q.ymlup -d ”

Ref: <https://docs.docker.com/engine/reference/commandline/compose/#description>

Ref: <https://docs.docker.com/get-started/08_using_compose/>

**Theory concepts**:

Docker: cuz they (containers) are better, lighter, faster, more portable, efficient, lightweight, and secure.

- Containers and VMs are similar in their goals:

to isolate an application and its dependencies into a self-contained unit that can run anywhere.

- containers and VMs remove the need for physical hardware.

- Virtual machine (guest machine): carries an entire virtualized hardware stack of its own, including

virtualized network adapters, storage, and CPU — which means it also has its own full-fledged guest

operating system.

- one difference between containers and VMs is: containers \*share\* the host system’s kernel with other containers.

-# Docker: is an open-source project based on Linux containers.

It uses Linux Kernel features like namespaces and control groups to create containers on top of an OS.

- \*\*docker allows to package an application (even) on laptop, which in turn can run unmodified on any cloud, or even bare metal.

- \*ADVANTAGE: the local environment setup is minimal and consistent across Windows, macOS, and Linux systems.

runs same environment always, Sandbox Projects, just works without all install dependencies.

- No pre allocation of RAM in docker. and have very less runtime.

===================

Docker Fundamentals:

===================

- Docker Engine: is the layer on which Docker runs. It’s a lightweight runtime and tooling that manages containers, images, builds, and more.

- It's made of: A Docker Daemon, A Docker Client, and A REST API.

- Docker COMPOSE is a tool for defining and running multi-container Docker applications.

if you have a need to define a service that requires more than one Docker container to function.

- The docker-compose.yml file is broken into sections, each section represents a single container

which, when combined with the other containers, create the service.

Ref: - https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b/

Ref: - https://stackify.com/guide-docker-java/

## DOCKERFILE: the Cooking Recipes of Docker Containers.

A Dockerfile starts with the FROM statement which tells Docker which base image you want to start with.

You can imagine this as using a canvas with the background already drawn and only the central part (your program) missing.

Most of the time the image you want to pull is the base image of your programming language, which you can

find at the before mentioned Docker Hub.

REF: https://medium.com/free-code-camp/a-beginners-guide-to-docker-how-to-create-your-first-docker-application-cc03de9b639f

'''

# A dockerfile must always start by importing the base image.

# We use the keyword 'FROM' to do that.

# In our example, we want import the python image.

# So we write 'python' for the image name and 'latest' for the version.

FROM python:latest

# In order to launch our python code, we must import it into our image.

# We use the keyword 'ADD' to do that.

# The first parameter 'main.py' is the name of the file on the host.

# The second parameter '/' is the path where to put the file on the image.

# Here we put the file at the image root folder.

ADD main.py /

# We need to define the command to launch when we are going to run the image.

# We use the keyword 'CMD' to do that.

# The following command will execute "python ./main.py".

CMD [ "python", "./main.py" ]

'''

=================================================

DOCKER

REF: https://youtu.be/YFl2mCHdv24

1. What is Docker

2. Virtual Machines vs. Docker

3. Introduction to Dockerfiles, images and containers

4. The Docker Hub

5. Writing a Dockerfile

6. Building an image

7. Running a container

8. Mounting volumes

9. One process per container

CONTAINER: is a running instance of an IMAGE

IMAGE - template for creating environment you wanted to run (its snapshot)

image: has OS, software, application code.

images are 'defined' using DOCKERFILE (a Text File), with a list of steps to perform, to create that image.

Use 'Dockerfile' to build the 'Image', which is run to get 'Container'.

\*\*\*

-p for forward port 80 on host (the computer) to port 80 in container ().

-- " docker run -p 80:80 <container\_name> "

\*\*\*\*

- to reflect changes made: rebuild the image and spin up new container from updated image

volumes: 2 types

1) to persist and share data between containers.

2) share folders b/w host and the container. mount a local directory as a volume inside the container. (to mount use'-v' while running in command line)

-- " docker run -p 80:80 -v /full/path/to/local/source/folder/:/var/www/html <container\_name> "

volume gives container the ability to see only files on the host, it do not change the image, image have to be rebuilt.

one process per container, life of container is tied directly to a single process. main process terminates, n container stops.

===============================================================

DOCKER COMPOSE:

REF: https://youtu.be/Qw9zlE3t8Ko