Docker containerize the application available.

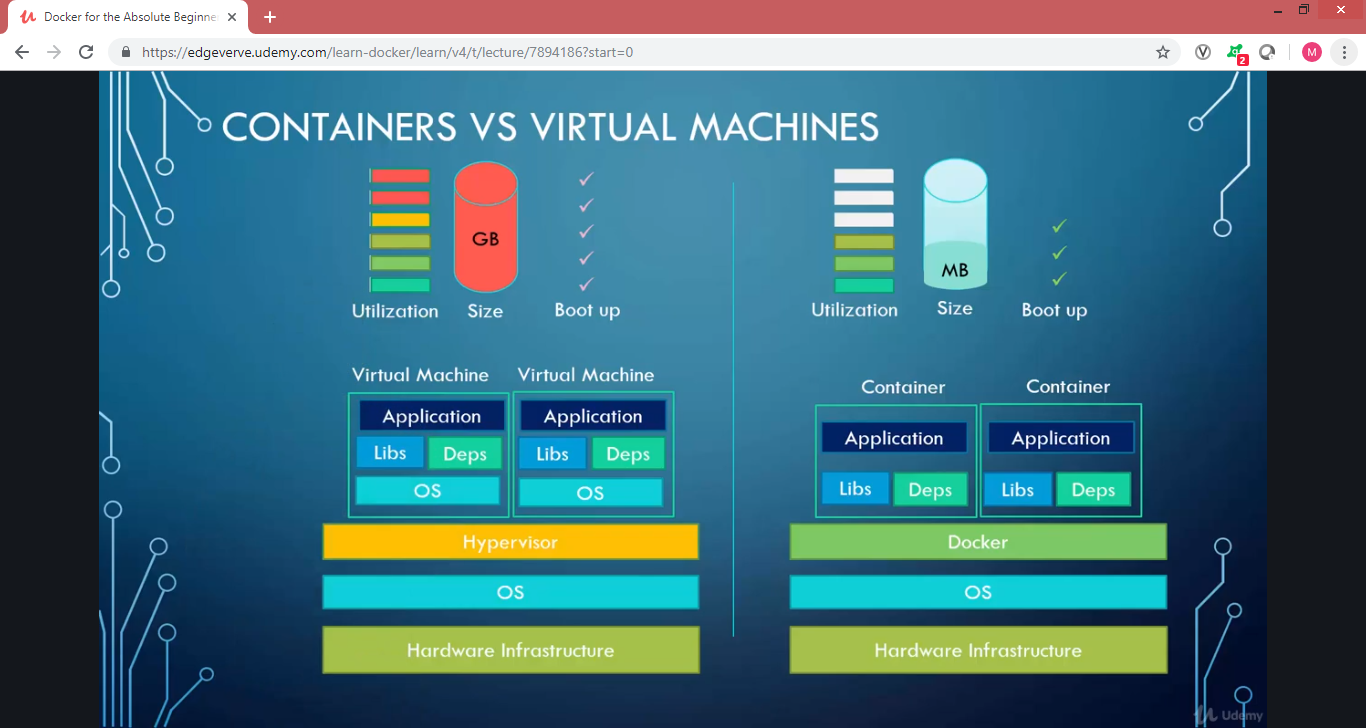
These docker containers share an OS Kernel as in there will be diff containers on the same OS kernel

Ex: If the OS is Ubuntu, then the docker can create containers with any OS like Fedora, etc which are LINUX based OS.

Windows based container cannot be hosted on a linux OS or linux based kernel

Containers vs VM

Architectural difference – each VM will have a separate OS while for all docker containers OS will be shared. PFB.



Containers vs Images:

Images are like usual class (in OOP) / template and Containers are like objects there can be multiple containers which are developed from the same image.

Basic commands

Run – this is used to create the docker containers from the image. If the docker image is not available in then it will be pulled from docker hub.

Ex: docker run <image name> -- docker run Ubuntu

-it option to be used to with run command to login to the container

-d tomake run command in the background.

If there are no services running in the containers, it will be auto stopped.

Ps – basic container details, with option –a it will give previous container details too.

Stop – to stop a container. Ex: docker stop docker\_name. But this doesn’t release the disk space.

Rm – this is used to remove the container.

Stopped but existing dockers will be shown with –a option in …docker ps command

Images – list of images which are downloaded

Rmi – remove images from the docker, we need to ensure that there are no containers running in that image

Pull - to pull the image from docker hub and save locally

Exec – to execute a command in the container.

Ex: docker exec docker\_name cat /etc/hosts

**TAG**:

RUN command for a specific version.

Docker run Ubuntu:17.04 – this is for running this specific version and is called a tag(part from :) if no tag specified by default it would tag the latest

**Attach and detach:**

-d option is used to run a command in detached mode

If needed to attach back then:

Docker attach docker\_name

**STDIN**

Docker uses “ -i ” as option for taking user inputs

**PORT MAPPING:**

Docker run –p 80:5000 docker\_name  
This will map port 80 of the docker host to port 5000 of the docker container

So we can run same instance from multiple ports in the docker host

**Volume Mapping:**

If we need the data in the container safe before removing a container, below command to be used:

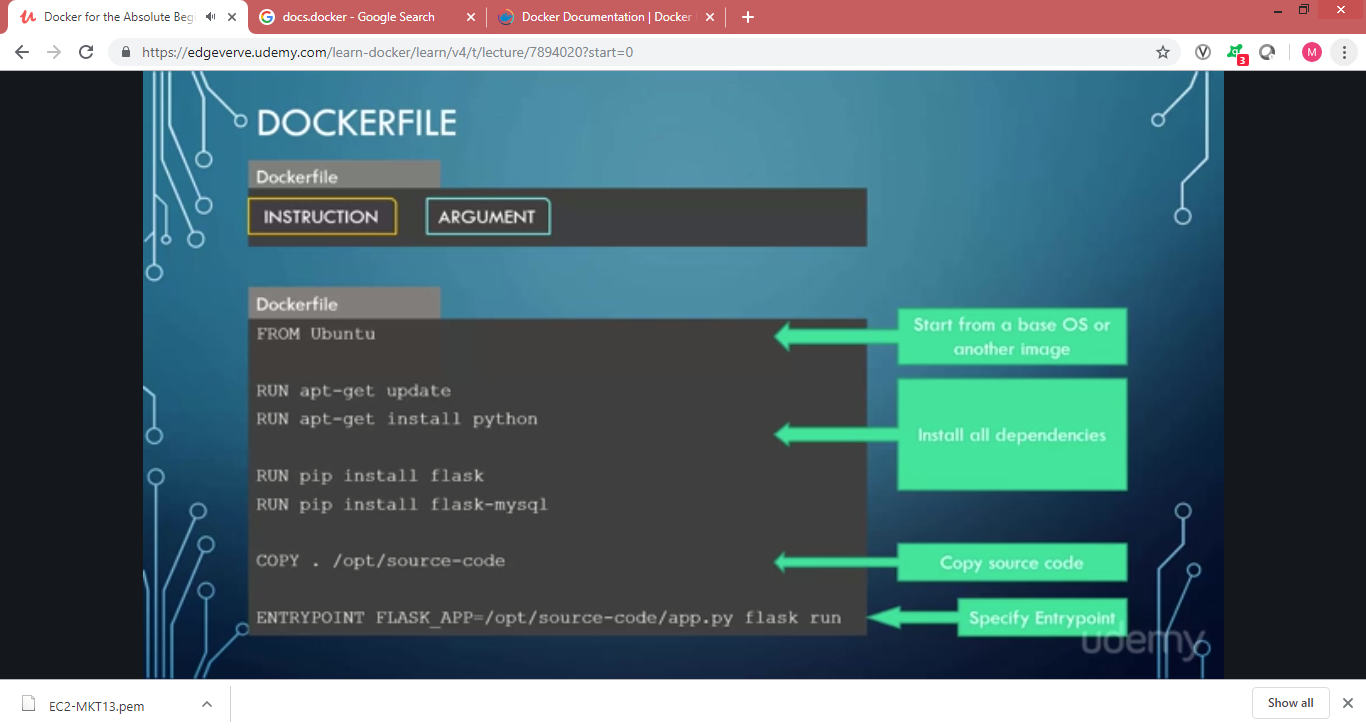
Docker run –v <new path – outside the docker container >:<old\_path> docker\_name

**Image**

Create a docker file – a file with instructions to the docker and arguments.

Docker build –t Dockerfile –t image\_name

Docker push image\_name



**Commands vs entry point**

The container will be available only as long as the webservices are available.

To make a certain command as permanent we can create a new image with the command included.

Ex of a sample docker file is as below:

FROM Ubuntu

CMD sleep 5 – the command can be given in this way or as CMD[“sleep”,”5”]

Entrypoint is like parameterized cmd command

Ex:

FROM Ubuntu

ENTYRPOINT [“sleep”]

Run command would be – docker run Ubuntu-sleeper 10 – the 10 mentioned as argument will be considered as argument for sleep command.

But if no argument specified that would cause an error inorder to give a default value during the docker creation we need to use CMD and ENTRYPOINT together as below:

FROM Ubuntu

ENTRYPOINT [“sleep”]

CMD [“5’]

In this above docker file the default value will be considered as 5 if no arguments passed in RUN, if argument is passed then the passed argument would be considered.

If the ENTRYPOINT command to be changed then below can be used:

Docker run –entrypoint sleep1.0 ubuntu-sleeper 10

Instead of using multiple docker run commands to set up a container, we can use a docker compose file.

This will be a .yaml file which will have separate services needed for the appln.

Docker-compose up – commands to bring up docker compose

\*\*Above command doesn’t rebuild the image, so to make any changes in the dockerfile to reflect in the creation of the image – docker-compose build

Docker-compose stop – to stop

Docker-compose down – to bring down

Docker

YAML file is used to represent data in key value pair

Ex: Key-value pair : in below example the values fruit, vegetable, liquid, water are the keys and apple, carrot, water and ice are values. The space after the : is mandatory to distinguish a value.

Fruit: Apple

Vegetable: Carrot

Liquid: Water

Solid: Ice

**Array/List**: - indicates that the values are part of arrays, the spacing should be common in all the elements

Fruit:

* Orange
* Apple
* Banana

Vegetable:

* Carrot
* Tomato
* Potato

**Dictionary:** Below is an example for dictionary, the spaces before the elements should be unique to distinguish between an element and dictionary name.

Banana:

Calories: 10

Fat: 10

Guava:

Calories: 30

Fat: 1

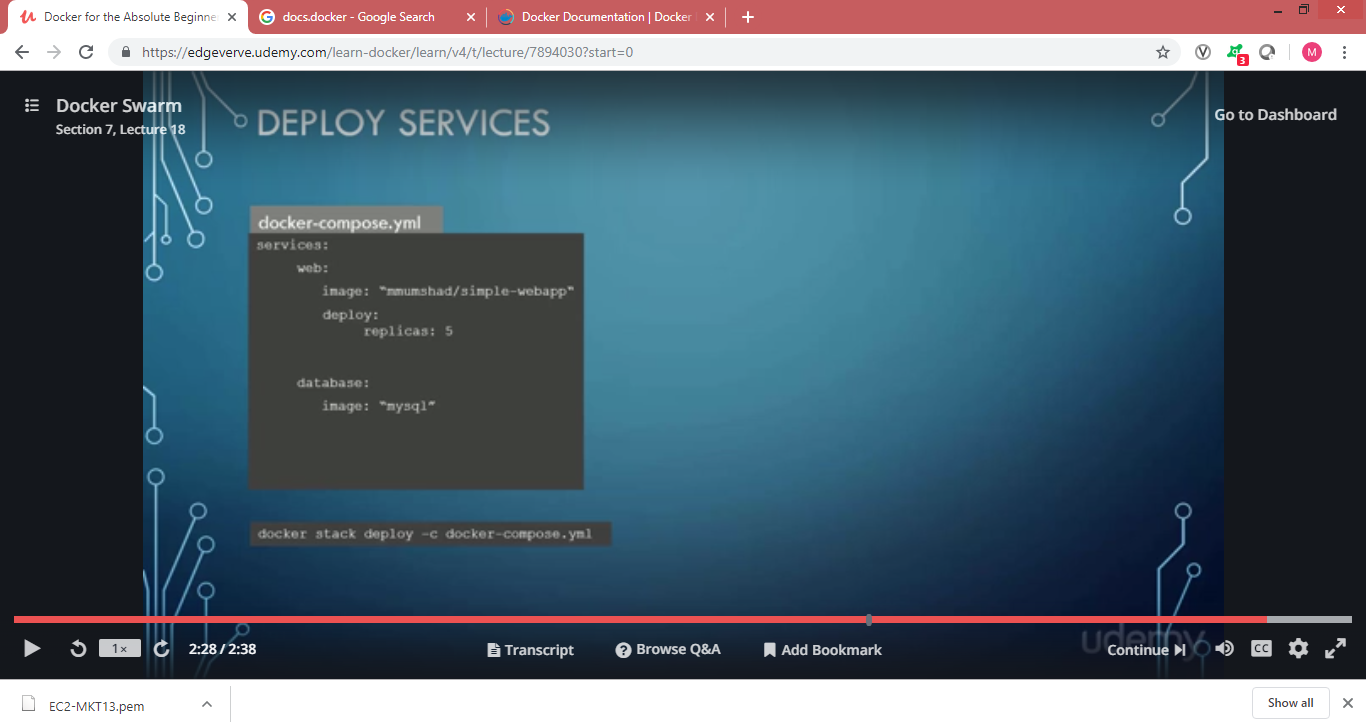
* Arrays are ordered and Dictionary is unordered

Docker Swarm: Instead of single docker host there will be multiple docker hosts that will be used to avoid any host issue

When there are multiple docker hosts one if it would be docker manager and the rest will be workers

Docker swarm init - to initialize manager

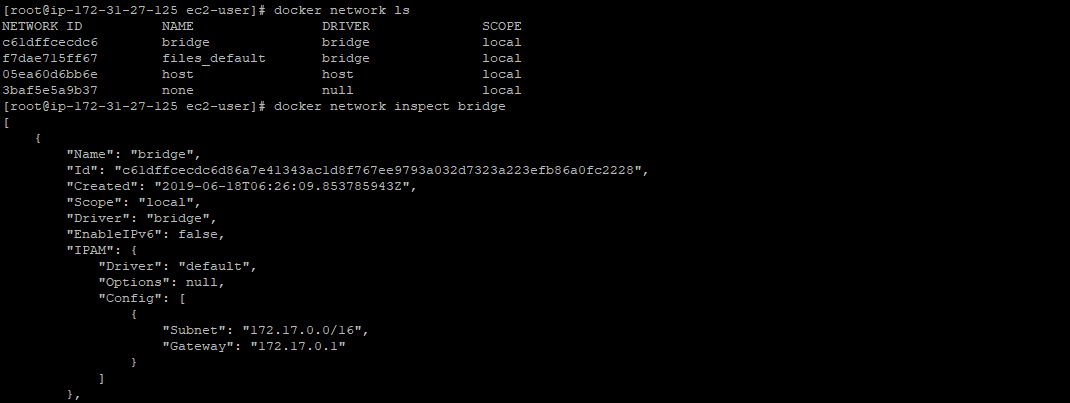
Docker swarm join –token <token > -- to join the manager



The deploy key added will create the web app in 5 instances across the docker hosts.

Default Networks

1. Bridge



Creating a new network

Docker network create –driver bridge <network name>

Containers in one bridge network cannot be able to connect to container in another bridge network. Below command to be used to enable the same

Docket network connect bridge <container name>

1. None – Isolated containers

* Creation will have option as “–net none”
* Ex: docker run -dit --net none busybox sleep 1000

1. Host – will have parameter as –net host

Will have access to all the ips available in the host.

We can create our own network with below commands:

Docker network create \

* - driver bridge \
* -subnet 182.18.0.0/16 (no space between hypens)

Custom-isolated-network

Docker network ls -- to list all the networks created

**Docker Images push:**

To push the image we need to have the image named with the docker hub (or any public repository) name.

Ex: docker tag <image id> <registry user name>/<imagename>:<version>

Note: Images with latest tag will not be auto updated when pushed into the registry

* Above command is like cp command the actual image will not be overwritten but the image will be copied

Ex: docker login –username=mabin

Docker push <registry user name>/<imagename>:<version>