**Docker Zero to Hero**

**Docker**:

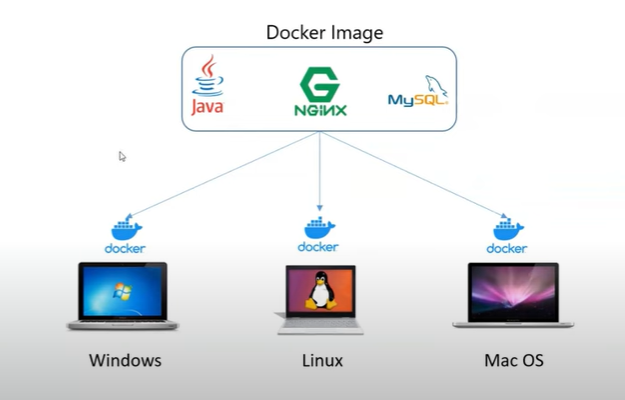
A docker is a set of platform as a service(PaaS) products that uses OS level virtualization to deliver software in packages called Comtainers.

Problem:

In development of an application different developers uses various OS types like Windows, Linux and Mac OS as per their feasibility. For the development of the application we need different supporting applications like JAVA, Nginx, MY sql, etc.

All the developers install these applications in their Systems to build the code and Post completion it has to be deployed into the production server of any one OS. Here the compatibility issues may arise due to the development of each part of the code in different OS. The code may not run properly.

Solution:

To mitigate this problem, we can use the docker. The docker engine is installed in all the developer’s systems irrespective of the OS. Now the supporting applications are installed on the docker. This creates a docker container and the same container is used by all the developers. This docker container is deployed in the production server also.

First, we have to create a docker image then we have to use the image in the docker engine to obtain a container.

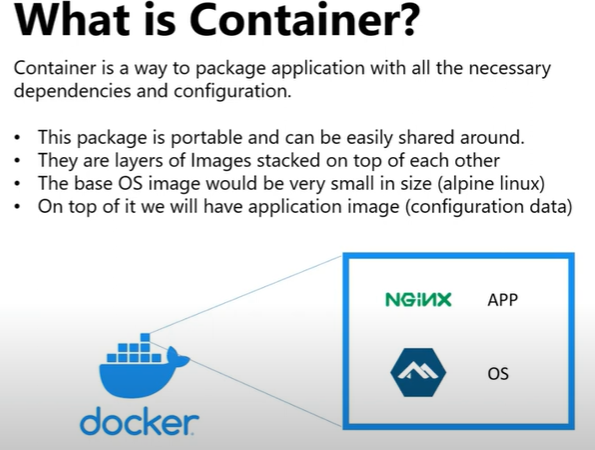
The developed code is packaged in the container and deployed in the server.

**Docker Container:**

A container is a way to package applications with all the necessary dependencies and configurations.

**Alpine Linux:**

It is an light weight OS of Linux. It is the smallest OS in the world.



A container is a package of Application, Libraries and system dependencies. This container has a minimal OS compared to the main OS. The containers use the resources from main OS.

Each container has a minimal OS to support the applications. This containers can be shipped and deployed in other machines by creating a Docker Image.

Docker File

Docker Image

Docker Container

First we write a docker file and with the help of docker engine we create a Docker Image and we use that image to create a container.

**Commands:**

Docker build - To build a docker image using the docker file.

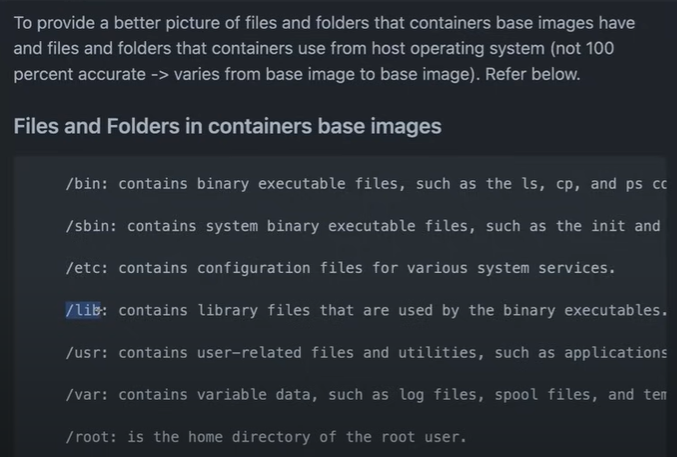
Docker run - To create a container using the docker image.

Docker push:- To push the image into the docker repository.

**Disadvantages of Docker:**

* The docker depends on the docker engine and the engine has a single point failure i.e if the engine goes down then all the containers stop working.
* Docker images has layers that takes more space.

To mitigate all these problems we have a tool called **BUILDAH**.

These are the basic system dependencies that isolates one container from another.

All the mentioned below folders are a part of the image and these are not shared between other containers.

**Docker Daemon:**

A docker daemon is a process and it is the heart of the docker.

Whenever we use a docker cli commands, that command is received by the docker daemon and gets executed to create an image or a container.

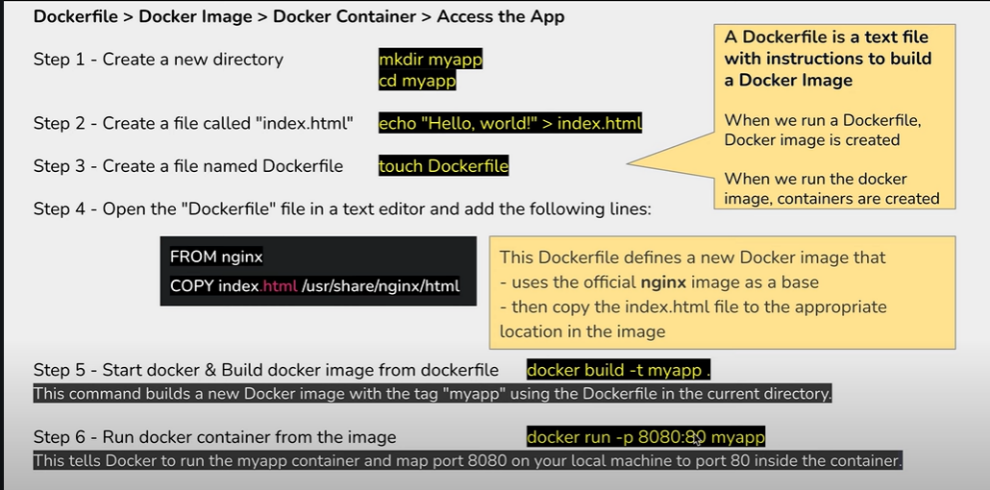
All the commands like docker build, docker run and docker push will gets executed by the daemon.

**Docker Hub** : It is a public registry for storing docker images.

**Git Hub** : It is a registry for story source codes and version control.

Docker username: chsaikuma7

Password: Shanvika@2021

**Creating a docker image from a docker file:**

**Important**:

* The docker file name should be Dockerfile.
* The docker file and the application code (myapp) file should be in the same folder

**Code explanation:**

* FROM means the mentioned application gets installed
* COPY means the file gets copied into the mentioned folder of the application.
* -t tags to differentiate the docker images
* -p maps the port of the VM to the port of the container
* -d detached mode (this helps in executing multiple containers at same time, the containers run in the background)

**We can push the image into our dockerhub account using the command docker push imagename.**

**docker run -p -d 8080:80 name**

**DOCKER COMMANDS:**

* **Docker version –** To know the docker version
* **Docker info -** To know the docker details
* **Docker login -** To login to docker with username and password of docker HUB
* **Docker images -** To know the docker images with image IDs
* **Docker build . –** To create a docker image from the docker file.
* **Docker pull -** To pull the docker images from docker HUB.
* **Docker push -** To push the docker images into the repositories of docker Hub
* **Docker rmi -** To remove the docker images or delete the images
* **Docker ps -** To know the docker containers currently running.
* **Docker run -** To create a docker container from docker image.
* **Docker start -** To start the container
* **Docker stop -** To stop the container
* **Docker stats -** Gives the memory, cpu details of the containers.
* **Docker system df-** Gives the dick usage of Docker.
* **Docker system prune -** This deletes the docker containers and images that are in ideal state and have no work.

**MULTI STAGE BUILD:**

Improve the efficiency of the build.

In multi stage build we can have any no. of stages but should have only one final stage.

This helps in reducing the size of the image.

**Distroless image:**

A distroless image is a light weight docker image that will have only the run time environments.

Golang helps in creating the distroless images. This reduces the image size to 10 mb or 15 mb.

**Along with this it provides better security.**

This has very less packages and only the runtime. As no vulnerabilities are present so it will be impossible for the hackers to enter your application.

Using golang at the final stage of the images will increase the security as it doesn’t require the run time environments.

Need in depth knowledge to write a multi stage or distroless image.

**Docker Volumes and Bind Mounts**

* A docker container directly cannot access the files in the local machine.
* A docker container cannot store the data and cannot retrieve the previous files data.

To over come these challenges we have Bind mounts and docker volumes.

**Bind Mounts:**

A bind mount will bind a folder in the container and a folder from the main machine. The folder in the container will access the folder in the host machine as there are bind together.

Through this even if the container goes down the binded folder in the host will have the information and we can bind it with another new container also.

**Volumes:**

It provides better life cycle. The volume is created as a logical partitioning of the host and can mount it with a container.

We can connect same volume to multiple containers. We can create a volume with the docker commands.

This creates a special drive volume space for the containers and the information of the containers is stored in the volumes.

We can create and destroy a volume. It has a flexibility to attach the container to an external storage service like S3, another ec2 instances, etc.

* The bind mounts should be on the same

**Commands:**

* Docker volume ls – to list the volumes in the machine host
* Docker volume create name – To create a volume( a logical partitioning)
* Docker volume inspect name – To get details of the volume
* Docker volume rm name - deletes the docker volume
* Docker volume rm name1 name2 - To delete multiple volumes at a time
* Docker run -d --mount source=volumename, target=/app imagename
* Docker inspect containerID – gives details of container

**Docker Networking:**

* Networking allows communication between containers and host machine.

1. Container to container communication
2. Isolation between containers.
3. Container to host communication.

**Bridge Networking :**

C2

C1

HOST

C1- 172.59.6.3

C2- 10.1.0.1

Host- 192.168.3.6

* Here the host and containers are of different subnets. We cannot ping the containers through host as they are of different subnets.
* Without communication the containers cannot use the resource from the host.
* So to solve this problem an virtual ethernet is created for the containers called Docker 0.
* The containers hosted in the host machine will use this docker 0 (v eth) to communicate with the host. Only one v eth is created in the host by docker. All containers use the same to communicate.
* This causes a security problem because of a single path communication.
* This communication is called Bridge Networking. This is an isolation network.

178.18.3.5

V eth

192.178.3.5

V eth

Docker-0

V eth

10.18.3.5

Host

**Host Networking:**

* In this way the docker will directly bind your container with the host. Here the container and host will be in the same subnet.
* The problem is that the people having access to host can easily access the containers and vice versa.
* Due to security reasons this is not used.

**Overlay Networking:**

This is used when we have multiple hosts and multiple containers. This is used in clusters.

All these networking is used to isolate the containers from each other so that hackers cannot access other containers using one container and at the same time the containers should also communicate with host.

**Custom Bridge Networking:**

* Earlier all the containers of the host will communicate through the docker-0 only, which is not a secure way.
* But using the custom bridge network we can create a custom bridge network so that the containers which needs isolation can use this and communicate with the host.

V eth

178.18.3.5

192.178.3.5

V eth

V eth

Custom net.

Docker-0

10.18.3.5

Host

Commands:

* **Docker network ls** – gives the networks of the containers
* **Docker network rm networkname** – deletes the network.
* **Docker network create name -**  creates a custom bridge network

**Containerizing a MERN Stack Application and Deploying using Docker Compose**

**Three Tier Architecture:**

A three tier architecture has 3 tiers.

1. Presentation (Front end/UI) (React JS, HTML & CSS)
2. Business Logic (backend/server) (JAVA,Python)
3. Data (Database) (SQL)

**MERN Stack:**

A mern stack is a three tier architecture that is popularly used.

**M- Mongo DB (database layer)**

**E- Express js ( business logic)**

**R- React js (sever/business logic)**

**N- Node js (front end UI)**

* Here all these are open source
* Has java script in three of them which makes it homogenous
* Simplicity of creating the code