MAHENDRAKUMAR V

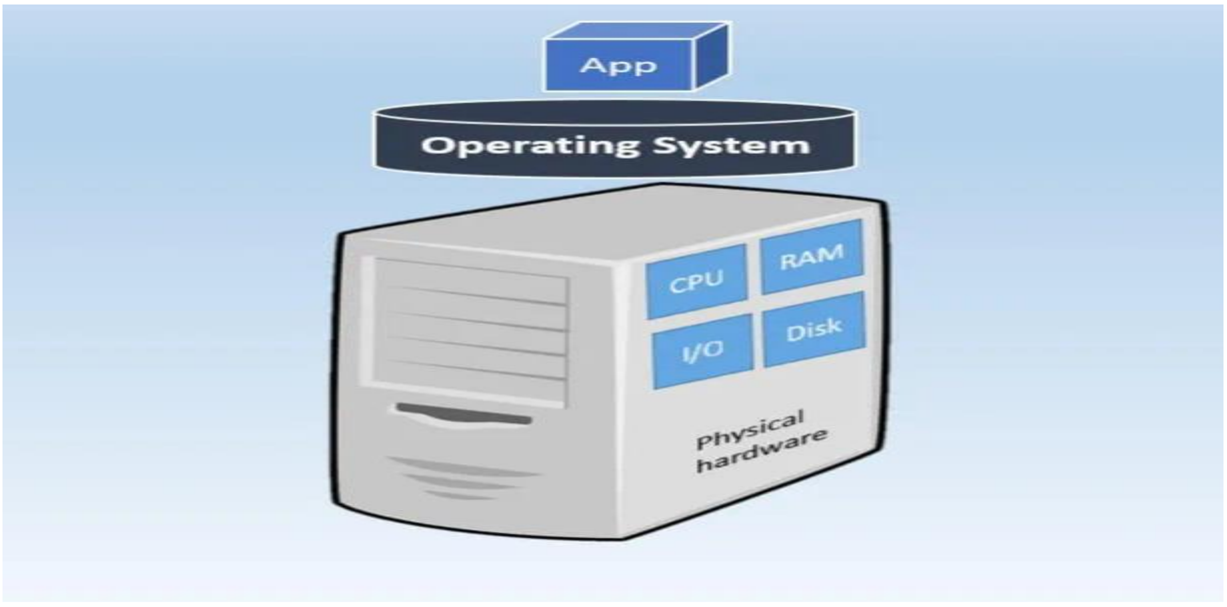
IT | CLOUD ENGINEER

Author: MahendraKumar V
LinkedIn: https://www.linkedin.com/in/mahendra-kumar-v/
Role: Cloud Engineer

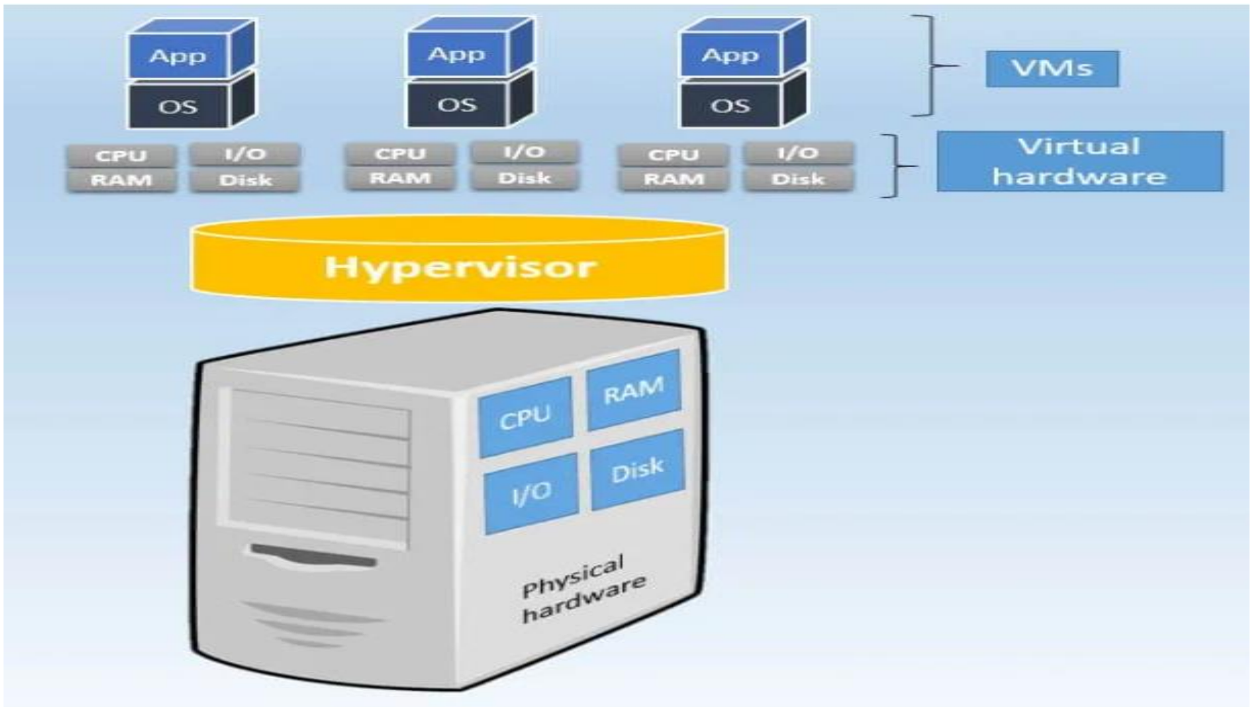
Docker

1. What is difference between VM vs Physical Server vs Container?

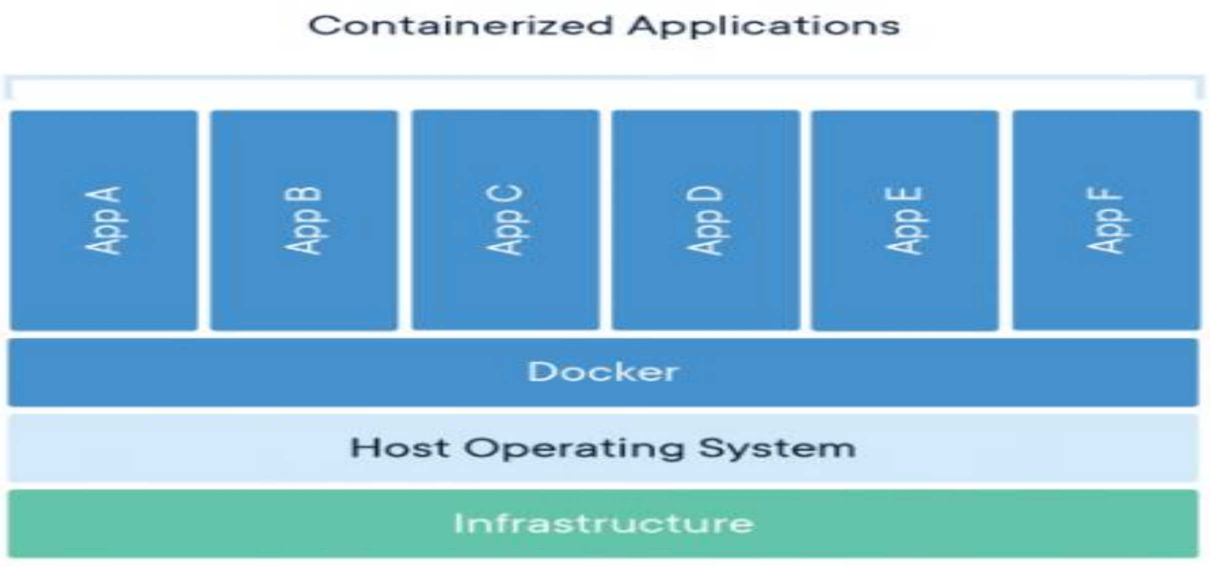
Physical server: Also know as bare metal sever's or dedicated server's.
Physical server is hardware device that store and manage data and run applications and operating system.
Physical server only design for sigle user, Its like single tenant computer server.
The resources and components of physical servers are not shared between multiple user's.
Each Physical server's are include Memory, Storage, Hardware, processor, Network connection and operating systems for running program and applictions.



Virtual Machine: This is software bashed computer that act as a physical server.
Virtual Machine is a multi tenant environment. Means multiple vm's run on the same physical hardware.
Virtual Machine can run program and store data and connect to network.
Virtial Machine created using virtualization technology, Which uses software to simulate hardware.
VM run on the virtual machine called host.
The hypervisor like vmware vSphere or microsoft VM's



What is Containers: Its run time environment and its isolated from the host machine
A light weight standalone executable package of software.
Isolate software from its environment.
Ensure that software works uniformly accross different environment.
Use containers to build, run and share your applications.
A container is a standard unit of software that package up code and all its dependancies.
So the application run quicly and reliable from our computing environment to another.



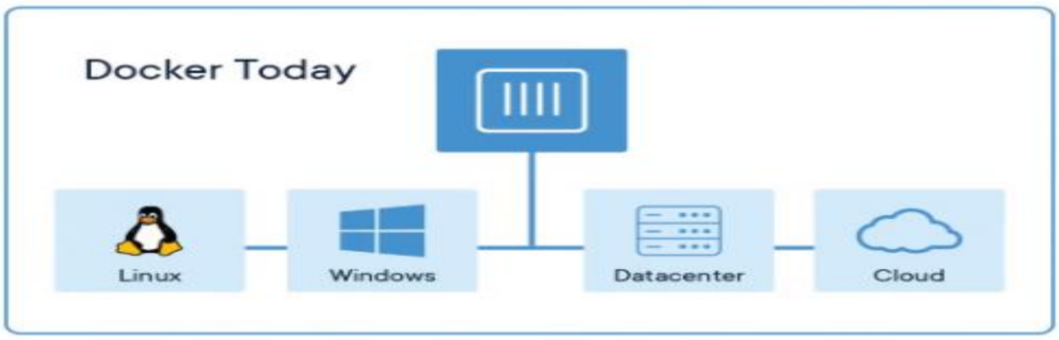
2> What is docker container and why we need it?

Container images become containers at run time and in the case of Docker containers – Images become containers when they run on Docker engine

Container images available for both windows and linux bashed applications.

Containers isolated software from its environment and ensure that it works uniformly desprite difference for instance between deployment and stagging.

Dockers today:



Docker containers are everywhere like linux, Windows, Datacenter, Cloud, Serverless...etc

Docker containers that are run in the docker Engine:

1> Standard
2> Light Weight
3> Secure

1> Standard:

Docker created industrial standard for containers, So they could be portable anywhere

2> Light Weight

Container share the machine os and kernal, Do not require os per application, Driving higher server efficiency and reducing servers and licence cost.

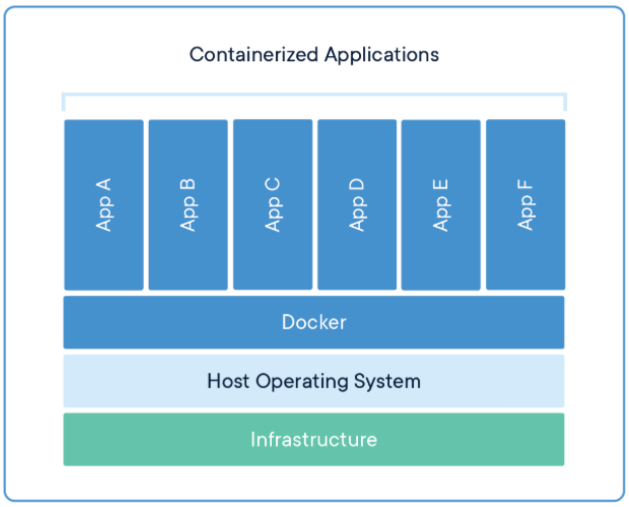
3> Secure

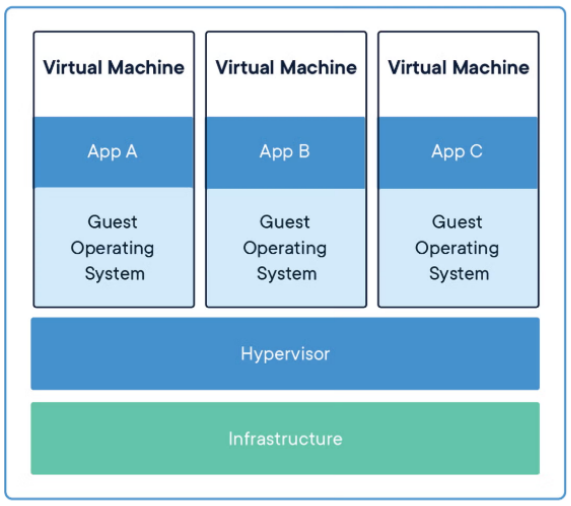
Container applications are safe and secure and Docker provides the strongest default isolation capablities in the induatry

3> Cocmparing Containers and Virtual Machines

Conainers: Containers are abstractions at the app layer that package code and dependencies together.
Multiple containers run on the same machine and share os kernal with other containers.
Each conatainers running on the isolated processes in use space.
Containers take less space then vm (Container imnages takes 10s of MB in image size.
Can handle more applications.

Vitual Machines: Virtual machine abstraction of physical hardware turning one server in to many servers
The Hypervisor allow multiple vm in to a single host machine.
Each vm included full copy of the OS, The application, Necssary binaries and libraries.

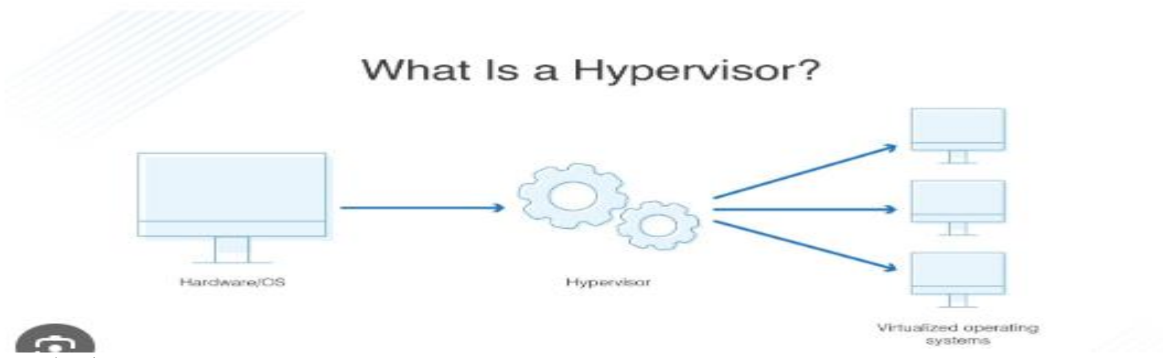




4. What is hypervisor?

Hypervisor is a software that allow **multiple virtual machines** to run on a **single physical server.**

The hypervisor allocates physical resources like memory and cpu to Vm's as needed



Each Vm's has its own operating system and application.
The hypervisor enables Vm's to operate independantly, Which provide better security and isolation between different application and os instances.

Benifits of hypervisor:
Reduce hardware cost
Maximize resource utilization.
Scalablity
Advance fetures like live migration.

5. Advantage and Disadvantages of Container?

Containers are having both advantages and disaddvantages including better application deployment, Security risk and storage challenges.

Advantages:
Portablity – Containers can be moved to different system and devices.
Scalability - Containers can scale up and down to meet challenges.
Resource Efficiency – Container helps make better resources.
Cross platform durablity – Container can run in different operating system, Like linux, Windows, mac
Application developement- Container can help speedup application development, testing and production.

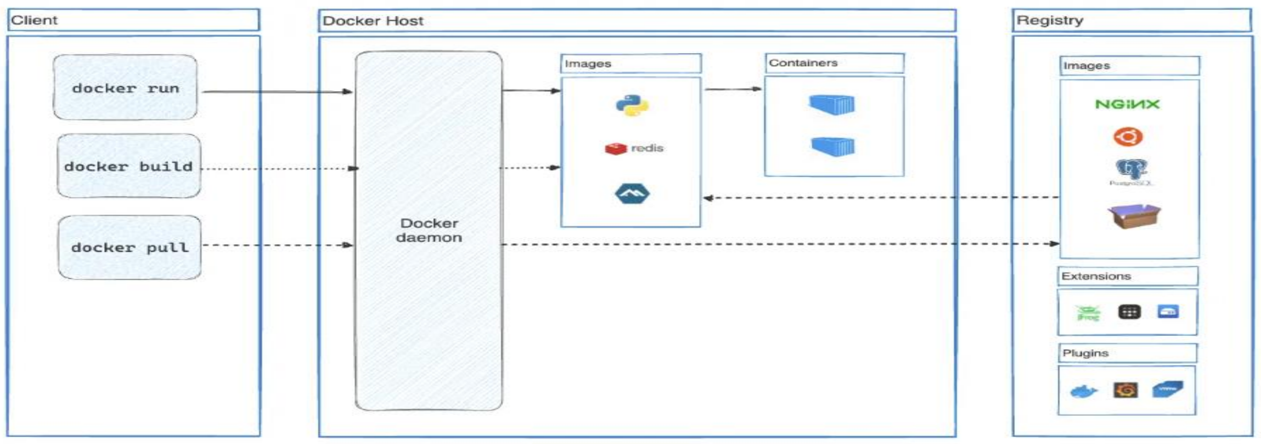
Disadvantages:
Security risk – Open top containers can be more vulnariblities to theft and tampering.
Storage and networking challenges – Container can only exist when they needed.
Complexity – Containers can be complex to manage.
Not all application benifits – Container may not be benifits for all applications, Especially those that are not designed to run microservice application.

Storage- containers are desiged to be stateless, They dont store data or state.

This can managing persistant data, Such as databases difficult within a container environment.
However there are solutions, such as persistant volumes, that can be used to manage data stored in the docker container environment.

6. How Docker Works? Explain in Detail?

1. Docker works as a client-server architecture.
2. The Docker client talks to the docker daemon.
3. Docker daemon does the heavy lifting of building, running and distributing your docker container.
4. Docker Daemon and Docker client run in the same system.



Or

if you want your docker client need to commicate with remote docker daemon, The docker client and docker daemon using REST api , over unix sockets or a network interface.

Another docker client is docker compose – That let you work with application consisting of a set of containers.
Docker Daemon -> Listen api request from client -> Manage image / Containers and network, Volume
Docker Client -> Docker user intract with docker using cli interface
Docker registry – Store docker images.

7. What are the problems with Docker?

Data Persistant : By default data with in a docker container is temporarly and lost when the container is stopped or removed. Users need to implement the solutions like volumes or bind mounts persist data externally

Resource Management : Running multiple containers can lead to high resource consumption on the host machine, Purticularly its not manage resource limit.

Scalablity Limitation: Docker alone might not support for large scalable deployment without any additional orchestration tool like kubernetes to manage container scaling and distribution across multiple host.

Complexity for complex application : Simple application can be containerized very fast and easy way, but some complex application with intricate dependencies might require significant efforts to properly containerize within docker.

Security concern – Containers are isolated between environment, Potential vulnariblity within the container image or the host system can still exploited.

8.Explain Container lifecycle?

Created: A container that has been created but not started

docker create --name <container name> <image name>

Running: A container running with all its processes

docker start <container name>

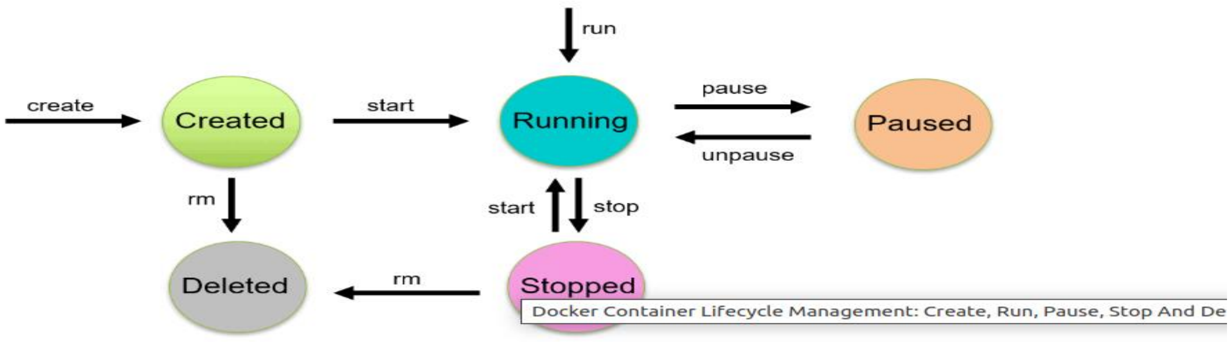
docker run -it --name <container name> <image name>
Paused: A container whose processes have been pause

docker pause <container name>
Stopped: A container whose processes have been stopped

docker stop <container name>
docker stop $(docker container ls –aq)

Deleted: A container in a dead state

$ docker stop <container name>
$ docker rm <container name>



8. What is DockerFile?

Docker file constrain set of instructions used to build docker image, It automate the process of packaging application with their dependencies
Docker file is text file.
Docker file instructions define that how to build a docker image step by step.

Steps for docker file creation:
1> Create a file named dockerfile
2> Add instructions in the Docker file
3> Build a docker file to crete a image
4> Run the image to creare a container.

Dockerfile are used to create a automate the process of creating docker image.(Which are blueprint of how to run docker container)

9. List the most commonly used instructions in Dockerfile?

Below is the commonly used instructions in the docker file

FROM : It is the 1st intructions of the docker file, Its specify the base image for the build process
COPY: Copy files and directories from Docker host in to Docker container.
RUN: During the image build execute run command.
WORKDIR: Set work directory for instructions and continer
ENV: Set environment variable that the container will use.
CMD: Default command that the container execute when its start.
ENTRYPOINT: Specify Default Execution
EXPOSE: Describe which port your application is listern on.
USER: Set User and Group ID.

10. What is Docker Compose? Diff btw Docker & Docker-Compose?

Describe which port your application is listern on.
Docker compose is a tool.
Docker compose tool is used to manage no of (Multiple) containers using yaml file.
Docker compose.yaml file define the configuratuions about network, Volumes allowing you to create and manage in single command.
Docker compose manage multiple containers such as Application, Web servers and Database and other server’s.
Insted of manually running single command to cretae docker container we can write yaml file to manage multiple container.

Dockercompose.yaml
Service Definition: You can define your applications service (Eg, App , Database)

Network Management: Allow to create netwoek communication between your containers. Enabling them to communicate between each of them.
Volume Management: You can specify persistent volume to srtore data, That need to survive containers restart.
Single command orchestration: docker command up – You can build start and manage all the containers.
Lifecyle Management: Provide commands to start, stop, restart and rebuild your application's services.

What is docker:
Docker is a containerization engin that provides a cli for building, running and managing individual containers on your host.

11. How will you lauch multiple application at a time in docker?

To launch multiple application at a time in docker by using docker compose,

Describe which port your application is listern on.
Docker compose is a tool.
Docker compose tool is used to manage no of (Multiple) containers using yaml file.
Docker compose.yaml file define the configuratuions about network, Volumes allowing you to create and manage in single command.
Docker compose manage multiple containers such as Application, Web servers and Database and other server's.
Insted of manually running single command to cretae docker container we can write yaml file to manage multiple container.

Dockercompose.yaml
Service Definition: You can define your applications service (Eg, App , Database)
Network Management: Allow to create netwoek communication between your containers. Enabling them to communicate between each of them.
Volume Management: You can specify persistent volume to srtore data, That need to survive containers restart.
Single command orchestration: docker command up – You can build start and manage all the containers.
Lifecyle Management: Provide commands to start, stop, restart and rebuild your application's services.

12. What is Multi Stage docker build?

Multistage docker build define that the way you run your build process in various stage.
Every stage have its own base image.
This allow your to create smaller, more optimized docker image.

Use multiple from instructions for download different images for evey stages.
Each from instructions perform each stage
Each stage perform sperate tasks.
Copy artifact from one stage in to another stage.
In Final stage copy only the necessary components to the final image.

13. What is Docker image?

From Docker file became docker image -> From Docker image became docker container.
Docker images is a lightweight, Standalone packages.
Packages that contain code, libraries, System tools to run a software application.
Its blueprint of Creating docker container
Docker images providing consistancy and portablity across different environment.
Images are created using docker files.
Docker images are stored in the registry like docker hub and can easily pulled, pushed and shared.

Docker images that are read only binary templates used to creare docker container.

Single file with all dependencies and configuration thar are required to run a program.
There is three different way to create docker images:
1> Create images from the docker file.
2> Create images from existing Docker hub.
3> Create images from existing docker or container.

14. Diff Btw Docker image & Docker layers?

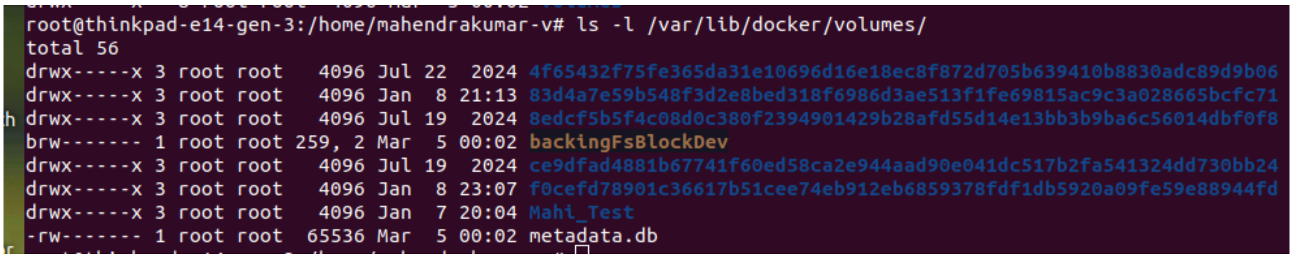
Docker images defines that read only template containing instructions and the necessary files to run an application.
Docker layer is building blocks of an image, each representing a specific file system changes (Adding and modifying)
Once the layer will created we can't be changed.
Layers are read only we can't change
Layers can be shared between different images, which helps to reduce size.

15. Explain Docker volume and its types?

Docker volume like a special folder (Directory inside our container)
Docker container can use to store data and share data.
Docker volume is a shared space where the container files are stored and share.
If once the container got destroyed then the data is still alive.

If you want the volume in to different container that need to work to gather.

If you created volume the we can easily share and use the same volume togather



```
root@thinkpad-e14-gen-3:/home/mahendrakumar-v# ls -l /var/lib/docker/volumes/
total 56
drwx-----x 3 root root   4096 Jul 22  2024 4f65432f75fe365da31e10696d16e18ec8f872d705b639410b8830adc89d9b06
drwx-----x 3 root root   4096 Jan  8 21:13 83d4a7e59b548f3d2e8bed318f6986d3ae513f1fe69815ac9c3a028665bcfc71
drwx-----x 3 root root   4096 Jul 19  2024 8edcf5b5f4c08d0c380f2394901429b28afd55d14e13bb3b9ba6c56014dbf0f8
brw------- 1 root root 259, 2 Mar  5 00:02 backingFsBlockDev
drwx-----x 3 root root   4096 Jul 19  2024 ce9dfad4881b67741f60ed58ca2e944aad90e041dc517b2fa541324dd730bb24
drwx-----x 3 root root   4096 Jan  8 23:07 f0cefd78901c36617b51cee74eb912eb6859378fdf1db5920a09fe59e88944fd
drwx-----x 3 root root   4096 Jan  7 20:04 Mahi_Test
-rw------- 1 root root  65536 Mar  5 00:02 metadata.db
```

By Default storage of Docker is *var*lib/docker/volumes
Docker stores images containers and volume in to *var*lib/docker path.

By default all the files created inside the container are stored on a writable container layer that sit top of the read only layer. Data written to the container layer does not persist when the docker container destroyed.

The writable layer is unique per container.
You can't easily extract the data from the writable later to the host or another container.

Volume is a directory inside our container -> 1st We have create a volume as a directory -> Then share volume -> Even if stop container still we can use volume because its shared volume

You can declare a directory as volume while creating a container

You can share volume across any number of container

but we can't change the existing container volume

You can mapped volume in two different way:
Host to Container
Container to Container

Type of Volume:
Named Volumes: Docker managed volume means docker create directory in host machine and store date, Share with multiple container
Binded Mounts: Map a directory or file on the host machine in to directory or file within the container.
Anonymous Volumes: Common volume that when the docker create automatically create anonymous volume when you dont provide name or host path.

16. Explain Docker Networks and its types?

Containers can connect to communicate with each other and external services.
Containers have network enabled by default, and they can make outgoing connection.
Container has no information about what kind of network its attached.
A containers only sees a network interface with an ip address, a gateway, a route table, DNS service and other network details.

Docker there is five different buit in network drive

Bridge : Create a software bashed bridge between your host and a container.
Containers can connect to the network Can communicate with each other, But they are isolate from outside network.
Each container in the network is assigned its own IP address. Because the network's bridged to your host, containers are also able to communicate on your LAN and the internet. They will not appear as physical devices on your LAN, however.

Host - Containers that use the host network mode share your host's network stack without any isolation. They aren't allocated their own IP addresses, and port binds will be published directly to your host's network interface. This means a container process that listens on port 80 will bind to <your_host_ip>:80.

None – The none network mode, Canot connect to the network and can't establish communication between containers or the external network.
Completly isolation between host and containers

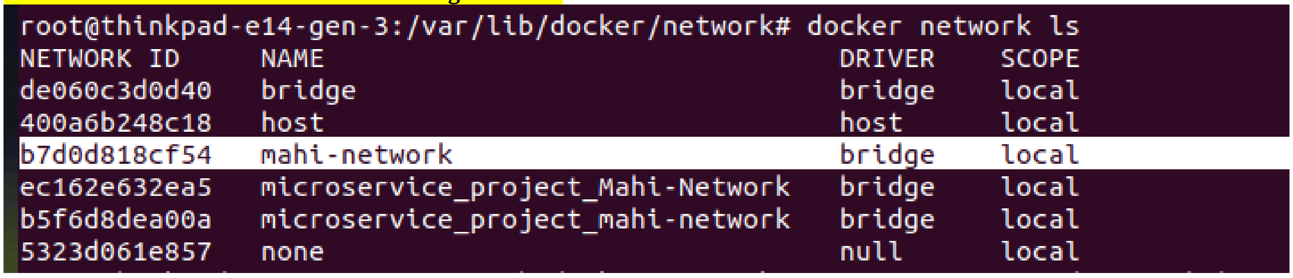
Overlays: networks are required when containers on different Docker hosts need to communicate directly with each other. These networks let you set up your own distributed environments for high availability.

IpVlan : IpvLAN networks are an advanced option for when you have specific requirements around container IP addresses, tags, and routing.

MacVlan : networks are useful in situations where containers must appear as a physical device on your host's network, such as when they run an application that monitors network traffic.

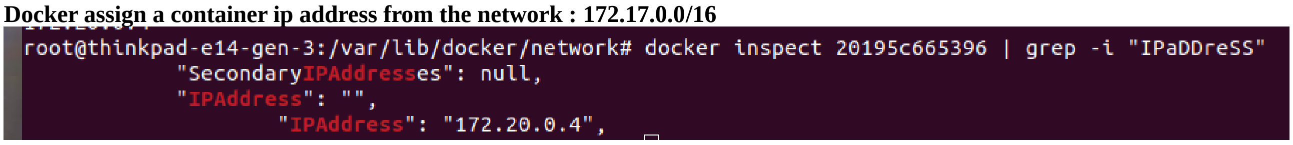
17. What is default docker container ip?

Docker uses a default network mode is bridge network



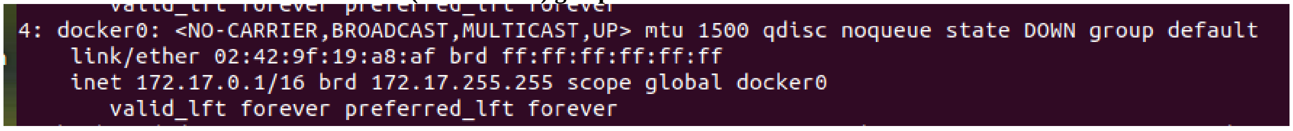
```
root@thinkpad-e14-gen-3:/var/lib/docker/network# docker network ls
NETWORK ID     NAME                                DRIVER    SCOPE
de060c3d0d40   bridge                              bridge    local
400a6b248c18   host                                host      local
b7d0d818cf54   mahi-network                        bridge    local
ec162e632ea5   microservice_project_Mahi-Network   bridge    local
b5f6d8dea00a   microservice_project_mahi-network   bridge    local
5323d061e857   none                                null      local
```

Docker assign a container ip address from the network : 172.17.0.0/16



```
root@thinkpad-e14-gen-3:/var/lib/docker/network# docker inspect 20195c665396 | grep -i "IPaDDreSS"
            "SecondaryIPAddresses": null,
            "IPAddress": "",
                    "IPAddress": "172.20.0.4",
```

The machine where the docker can run (Docker host) get ip address 172.17.0.1/16 on this network

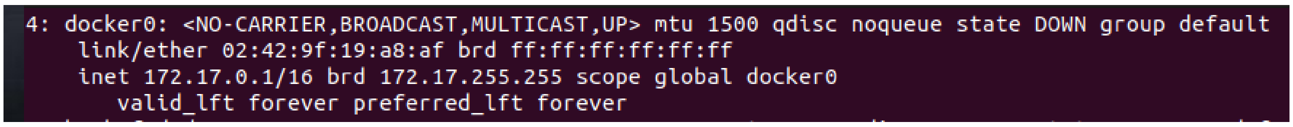


```
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:9f:19:a8:af brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

18. How will you assign static ip to docker container?

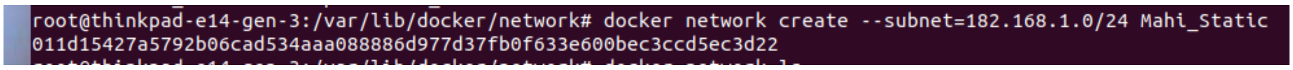
1st step we need to create a new own docker network drive (Need to create user defined network)

The default docker network

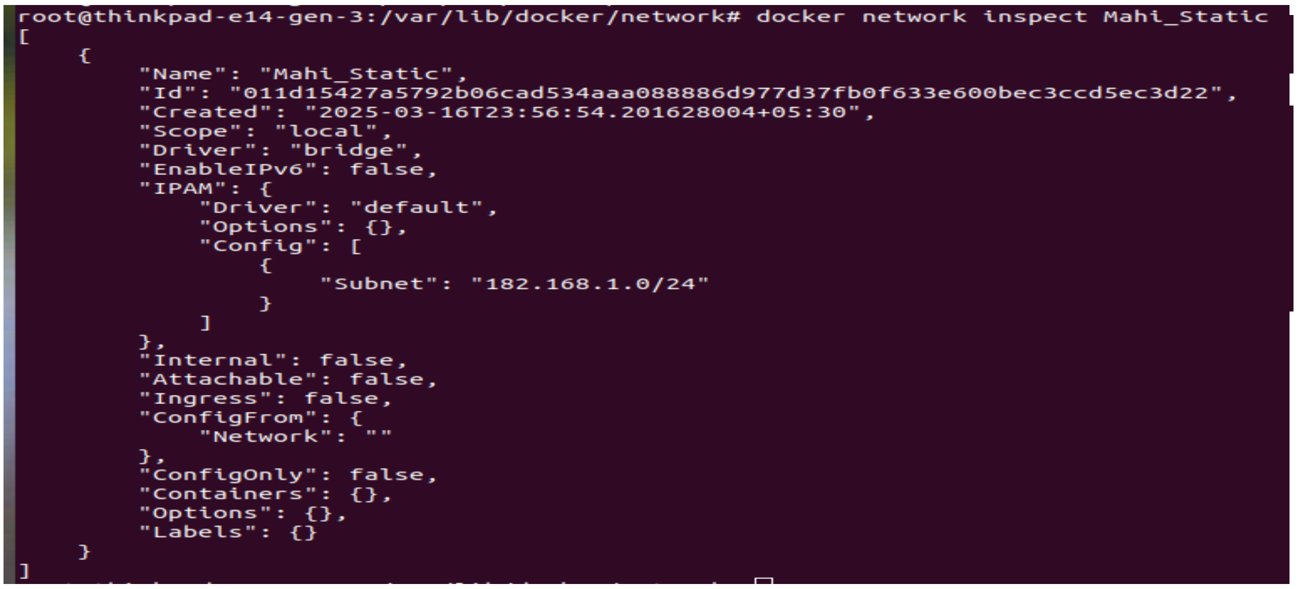


```
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:9f:19:a8:af brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

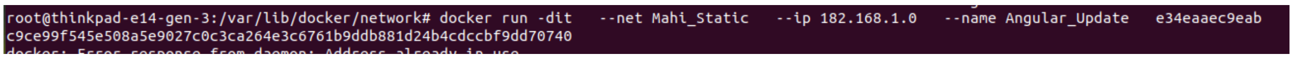
Create a new docker network with non overlapping CIDR value



```
root@thinkpad-e14-gen-3:/var/lib/docker/network# docker network create --subnet=182.168.1.0/24 Mahi_Static
011d15427a5792b06cad534aaa088886d977d37fb0f633e600bec3ccd5ec3d22
```



```
root@thinkpad-e14-gen-3:/var/lib/docker/network# docker network inspect Mahi_Static
[
    {
        "Name": "Mahi_Static",
        "Id": "011d15427a5792b06cad534aaa088886d977d37fb0f633e600bec3ccd5ec3d22",
        "Created": "2025-03-16T23:56:54.201628004+05:30",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "182.168.1.0/24"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```



```
root@thinkpad-e14-gen-3:/var/lib/docker/network# docker run -dit   --net Mahi_Static   --ip 182.168.1.0   --name Angular_Update   e34eaaec9eab
c9ce99f545e508a5e9027c0c3ca264e3c6761b9ddb881d24b4cdccbf9dd70740
docker: Error response from daemon: Address already in use
```

19. How will you Change the Default Docker Subnet IP Range

By default, Docker assigns the `172.17.0.0/16` subnet to the `docker0` bridge network. You can change this default subnet by modifying **Docker’s daemon configuration.**

There is two way to chage default docker subnet ip range
1> Edit the docker deamon configuration or creare *etc/docker/daemon.json*
vim etc/docker/daemon.json
```
{
  "bip": "192.168.1.1/24" // Bridge ip define the subnet for docker0
}
```

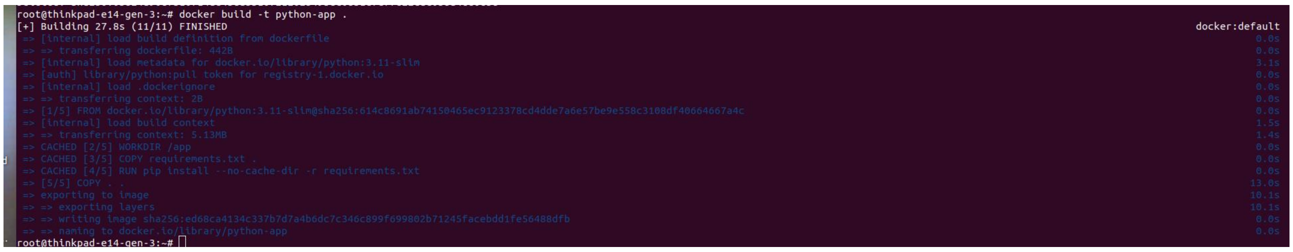
2> Creare a custom bridge network
Instead of modifying the default network, you can create a **custom bridge network** and use it for your containers:
```
docker network create \
  --subnet=192.168.100.0/24 \
  my_custom_network

docker run -dit --net=my_custom_network --name=my_container nginx
```

20. Will you write one simple dockerfile for java application or python app or nodeJs app?

```
# Use Python base image
FROM python:3.11-slim

# Set the working directory
WORKDIR /app

# Copy application files
COPY . .

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Expose the application port
EXPOSE 5000

# Run the application
CMD ["python", "app.py"]
```

Build and Run using the below command

```
docker build -t my-python-app .
docker run -p 5000:5000 my-python-app
```



```
root@thinkpad-e14-gen-3:~# docker build -t python-app .
[+] Building 27.8s (11/11) FINISHED                                                                    docker:default
 => [internal] load build definition from dockerfile                                                             0.0s
 => => transferring dockerfile: 442B                                                                             0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                              3.1s
 => [auth] library/python:pull token for registry-1.docker.io                                                    0.0s
 => [internal] load .dockerignore                                                                                0.0s
 => => transferring context: 2B                                                                                  0.0s
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:614c8691ab74150465ec9123378cd4dde7a6e57be9e558c3108df40664667a4c   0.0s
 => [internal] load build context                                                                                1.5s
 => => transferring context: 5.13MB                                                                              1.4s
 => CACHED [2/5] WORKDIR /app                                                                                    0.0s
 => CACHED [3/5] COPY requirements.txt .                                                                         0.0s
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt                                              0.0s
 => [5/5] COPY . .                                                                                              13.0s
 => exporting to image                                                                                          10.1s
 => => exporting layers                                                                                         10.1s
 => => writing image sha256:ed68ca4134c337b7d7a4b6dc7c346c899f699802b71245facebdd1fe56488dfb                     0.0s
 => => naming to docker.io/library/python-app                                                                    0.0s
root@thinkpad-e14-gen-3:~# 
```

21. Will you write one docker compose file and explain in detail?

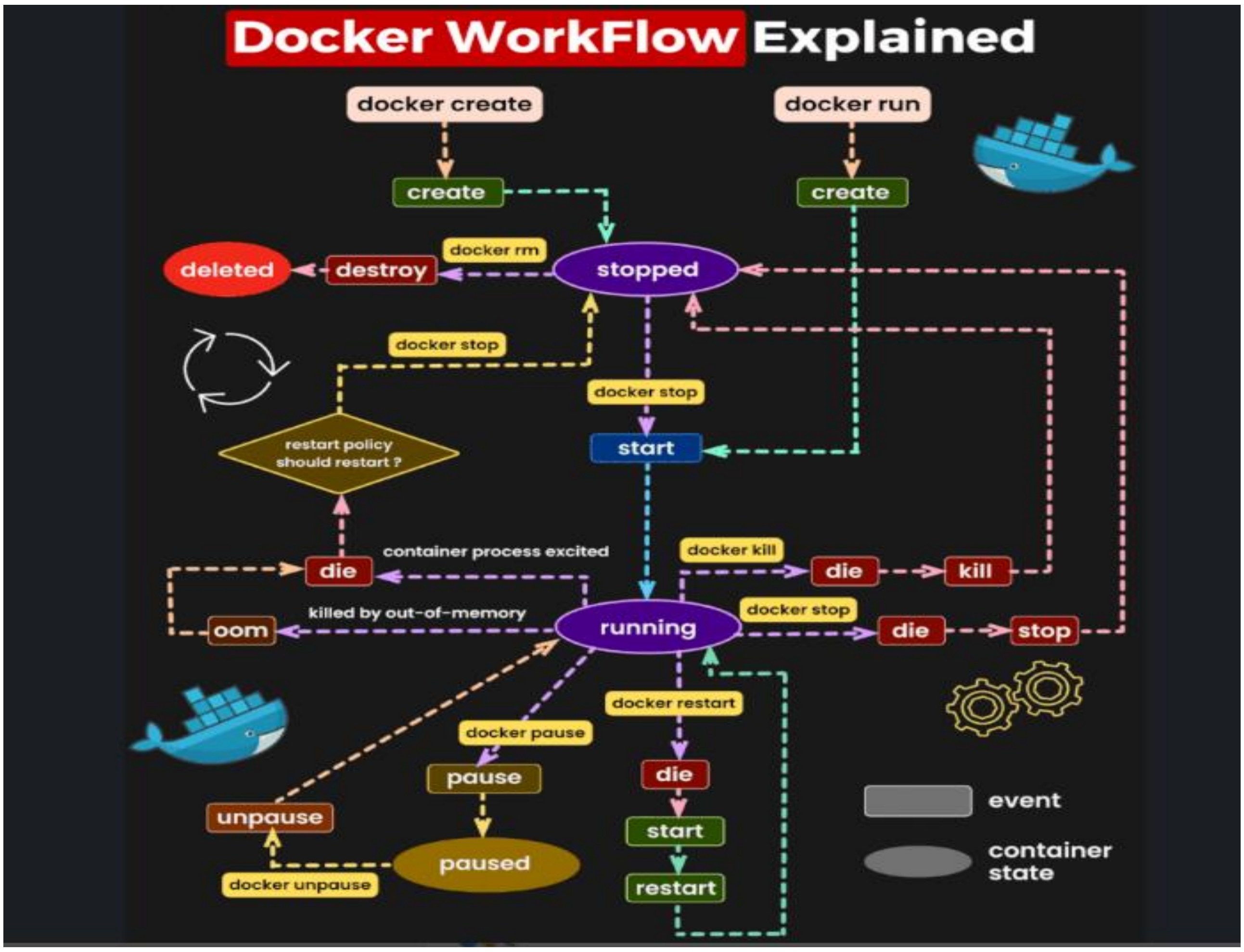
Version: "3.8" // Specify the docker composer version
services: **// Defne the specific service**
  web: **// Service name that specify the container for the flask application**
    build: . **// Specify the docker to build image using docker file . Which is current working directory**

```
    container_name: flask_app // Specify the container custom name
    ports:  // Specift the host port and inside container port
      - "5000:5000"
    volumes: // Map the current directory inside the container with /app
      - .:/app
    environment: // Set Environment Variable
      - FLASK_ENV=development
    command: python app.py // Over ride the default command in the docker file.
```

22. Will you write one multi stage docker file?

A **multi-stage Dockerfile** helps reduce image size by separating the build and runtime environments. Below is a **multi-stage Dockerfile** for a **Python Flask application**:

```
# 💥 Stage 1: Build Stage
FROM python:3.11-slim AS builder

# Set working directory
WORKDIR /app

# Copy only requirements file first (for caching)
COPY requirements.txt .

# Install dependencies in a temporary layer
RUN pip install --no-cache-dir -r requirements.txt


# 💥 Stage 2: Final Production Stage
FROM python:3.11-slim

# Set working directory
WORKDIR /app

# Copy installed dependencies from the builder stage
COPY --from=builder /usr/local/lib/python3.11/site-packages /usr/local/lib/python3.11/site-packages
COPY --from=builder /usr/local/bin /usr/local/bin

# Copy the application source code
COPY . .

# Expose the application port
EXPOSE 5000

# Run the application
CMD ["python", "app.py"]
```


Docker WorkFlow Explained
docker create
docker run
create
create
docker rm
deleted
destroy
stopped
docker stop
docker stop
restart policy should restart ?
start
container process excited
docker kill
die
die
kill
killed by out-of-memory
oom
running
docker stop
die
stop
docker restart
docker pause
pause
die
unpause
start
paused
restart
docker unpause
event
container state

All The Best !!