7/24/2025
7/04/2025

Docker

1) What is a Docker ?

Docker is a containerization platform that allows me to package applications with all dependencies in to light weight portable containers that can run anywhere.

2) What is the difference between a Docker image and a container ?

An image is a blueprint or template. A container is a running instance of that image. Images are read-only; containers are live & interactive.

3) What is the difference between Docker & virtual Machines !

Docker uses OS-level virtualization (shares host kernel), while VMs virtualize the entire OS. Docker is faster and more lightweight than VMs.

4) What is a Dockerfile ?

A Dockerfile is a Script with a Set of instructions used to build Docker images automatically (ex: installing packages, copying files, exposing ports)

5) What is Docker compose?

Docker compose is a tool to define and manage multi-container docker applications using a docker-compose.yaml file.

6) What are Docker volumes? why are they important?

Volumes are used for persistent storage in Docker. They allow data to persist even if the container is deleted.

7) What is Docker network?

Docker networks allows containers to communicate with each other. Types includes bridges, host, overlay and none.

8) What is a dangling image?

A dangling image is an image without a name or tag. They are usually intermediate images left after a failed build.

9) How do you monitor & debug docker containers?

- docker logs <container>
- docker stats
- docker inspect.

10) How do you handle container persistence?

By using Docker volumes or bind mounts to store data outside the container file system.

11) What are the best practices for writing a dockerfile?

- using a minimal base image (Alpine) reduces potential vulnerabilities.
- Running as a non-root user helps prevent previlege Escalation.
- Including sensitive information in the docker file is a security risk.
- Avoid latest tags.
- combine commands to reduce layers.
- use .dockerignore
- use multi-stage builds.

12) How do you secure docker containers?

- use non-root users
- Regularly scan images for vulnerabilities.
- use signed images (Docker content Trust)
- use official images
- Enable resource limits
- Network isolation.

13) What is the difference between -d and -it in docker run ?

-d : detached mode (runs in background)

-it : Interactive terminal.

14) What is a Docker Registry ?

A storage for Docker images (Ex: Docker Hub, ECR)

15) What is .dockerignore ?

Specifies files/folders to exclude when building images.

16) What are multi-stage builds in Docker ?

Technique to reduce final image size by using multiple FROM instructions.

17) What is the difference between bind mount & Volume ?

Volume : Managed by Docker.

Bind mount : points to a host path.

} Main difference

Docker Volumes :

1. Managed by Docker (/var/lib/docker/volumes/) → on Linux
2. Best for sharing data between containers.
3. Can be backed up and migrated (→ portable) Easily.
4. Created using Docker CLI (docker volume create)

Ex: docker volume create myvolume → volumename

docker run -it -v myvolume:/data --name ct1 ubuntu → image, → containername

volumename ↓ mount point inside the container

Bind mounts :

- Directly maps a host machine directory to a container.
- More flexibility but relies on the file system.
- Requires absolute path.
- Not as portable as volumes.

Ex: docker run -it -v /home/user/app:/data --name ct2 ubuntu

→ Maps the host folder to the container folder

/home/user/app → Host folder (directory on your Host machine)

/data → container folder (directory inside the container)

18) What is the host network mode ?

shares host's network stack; reduces isolation

19) Why should we avoid host network in production?

No isolation, security risks, port conflicts.

20) How do you reduce docker image size ?

- use minimal base images (Ex: Alpine)
- Multi-stage builds
- clean up unnecessary files.

21) What is the role of cgroups in docker ?

Used to limit resources (cpu, memory) to containers.

22) What are namespaces in Docker ?

Provide isolation of processes, network, and filesystems.

23) What is a docker commit ?

Docker commit is a command used to create a new Docker image from an Existing container.

It captures the current state of a running or stopped container and saves it as a new image.

docker commit <container_name/container_Id>

Ex: docker commit ct1
↳ container name

24) What is Docker attach vs Docker exec ?

Docker attach:

- connects to the main process of a running container, Exiting (ctrl+c) may stop the container if it's running in the foreground.

docker attach <cont_name/cont_Id>

Docker exec: Runs a new process inside an existing container, Does not affect the main process.

- Just adds another process & safer for debugging because you can Exit without stopping the container.

docker exec -it <cont_id/cont_name> /bin/bash

25) What is docker tag ?

The docker tag command assigns a new name (tag) to an existing docker image.

docker tag <Existing_image:tag> <new image_name:tag>

26) What are main Dockerfile instructions ?

a) FROM: Every Dockerfile Must start with this
Tells Docker which base image to start with.

Ex: FROM amazonlinux

b) WORKDIR: Sets the working directory inside the container.

Ex: WORKDIR /app

"Go into the /app folder inside the container & do the next steps there.

c) COPY: copy files from your host machine into the container.

Ex: COPY . /app

Copy Everything from current folder (host) to /app in the container.

d) ADD : Similar to copy, but can also download files from URLs or Extract tar files.

Ex: ADD https://example.com/file.zip /app/

Download and add file.zip into /app/.

e) RUN : Runs a command while building the image (not when the container runs)

RUN Yum install -git -y

Update packages and install Git during the image build.

d) CMD : sets the default command to run when the container starts.

CMD ["/bin/bash"]

Start /bin/bash when the container runs.

e) ENTRYPOINT : Also defines the startup command like CMD, but more strict. Used when you always want the command to run. (cannot be overridden)

Ex: ENTRYPOINT ["python3"]

Always run python when the container starts.

f) EXPOSE : Tells Docker which port the container will use.

Ex: EXPOSE 80

This container will listen on port 80.

27) what is the default Docker Network ?

bridge network.

28) what is docker0 ?

docker0 is the default virtual bridge network created by Docker on the host when Docker is installed.

It acts like a virtual switch that connects containers to each other and to the host via NAT (Network Address Translation).

IP range : usually 172.17.0.0/16

29) Explain Dockerfile instructions ?

FROM : Sets the base image

RUN : Executes commands in a new image layer

COPY : copy files

ADD :

CMD : provides default Execution Arguments

ENTRYPOINT : Sets the main process

EXPOSE : indicates the containers listens on a port

ENV : Set Environment Variables inside the container.

ARG : Defines a variable only at build time.

VOLUME : Creates a mount point & declares a volume for persistence.

Differences

30) Write the differences between a Docker and VM ?

Aspect	VMs	Docker
Isolation	completely isolated, Each VM has its own OS.	Lightweight isolation, shares the host OS kernel.
Resource Usage	Requires more resources (CPU, RAM, storage)	uses fewer resources, runs faster & lighter.
Boot time	Slower to start as it loads the Entire OS.	starts quickly, as it runs within the host OS.
Portability	Less portable	High portable
Use case	Ideal for running multiple OS and legacy apps	perfect for microservices & cloud native apps.

31) Difference between container & Image ?

Aspect	Image	Container
Definition	A static blueprint or template for applications.	A running instance created from an image.
State	Immutable (does not change after creation)	Mutable (can change & interact while running.
Purpose	used to create containers	Executes tasks & runs applications.
Life cycle	Exists permanently until deleted	Temporary & exists while in use.

32) what is the difference between Docker attach & docker exec ?

docker attach : connects to the main process of a running container.

- Exiting (ctrl+c) may stop the container if it is running in the foreground.

docker exec : - Runs a new process inside an existing container. Does not affect the main process.

- Just adds another process & safer for debugging because you can Exit without stopping the container.

Note : 1) use "docker attach" when you want to connect to the main process (but be careful with ctrl+c)

2) use "docker exec" when you want to run an Extra command inside the container without stopping it.

33) Write the difference between Docker commit & dockerfile ?

Aspect :	docker commit	Dockerfile
Purpose	creates a new image from a container's current state.	used to define instructions for building docker images.
Usage	Manual process : captures changes made to a running container	Automated process : builds images based on specific configurations.
Flexibility	Limited, stores the Exact state of the container at a point in time.	Highly customizable, allows defining layers, commands and environments.

34) Write the difference between Docker Hub and Docker registry ?

Docker Hub:
- public website to store docker images
- you can push and pull images
- Like a "playstore" for dockers
- Managed by docker company

Docker registry:
- A private place to store docker images
- you host it yourself
- Used inside companies for security.

35) what is the difference between Bridge & None network?

Bridge Network (default):
- Docker's default network type.
- provides internal communication between containers.
- containers get an IP address (ex: 172.17.xx) and can access each other if in the same network.

None network: (High security)
- Disables all networking for the container.
- Container gets no internet network interface
- No internet, no communication with other containers or host.

36) what is the difference between ARG and ENV ?

a) ARG = Build-time Variable:
- Used only during image build.
- Not available in the running container.
- can be passed using --build-arg

Ex: you tell the chef (Dockerfile) what topping to use while making the pizza (building the image).

Dockerfile
```
ARG TOPPING=cheese
RUN echo "Adding $TOPPING topping"
```

You run:
```
docer build --build-arg TOPPING=pepperoni
```

But.. once pizza is ready (container runs), you can't change the topping.

b) ENV = Runtime Variable: used during build & after container starts. Available inside the running container.
- ENV - can change this anytime without rebuilding

→ Dockerfile:
```
ENV TEMP=hot
CMD echo "serving pizza at $TEMP temperature"
```
→ you run: docker run -e TEMP=cold mypizza

→ During runtime, the container prints
Serving pizza at cold temperature

* you can change this anytime, without rebuilding..

37) How do containers communicate with each other?

Via Docker networks. We can use the default bridge or create a custom network. Containers on the same network can refer to each other by container name.

38) What linux features are essential for containerization?

1. Namespaces: Isolate resources for each container.
2. Control groups (cgroups): Limit & monitor resources (cpu, memory, disk etc.) per container.
3. chroot (change root): Filesystem isolation.

39) Explain image layering in Docker?

- Images are built in layers.
- Each instruction in Dockerfile creates a layer.

40) How do you check the IP of a container?

docker inspect container_id

41) What happens if you don't specify CMD or ENTRYPOINT?

The container runs and exits immediately as there is nothing to execute.

42) What is the difference between docker pause & docker stop?

pause Suspends processes.

stop terminates them gracefully.

43) What is the difference between Dockerfile & docker commit?

Dockerfile is reusable & version-controlled.

Commit is Manual & static.

44) How does docker Ensure isolation?

using Linux namespaces & cgroups.

45) How do you troubleshoot a container that exits immediately?

check logs using docker logs, inspect Dockerfile & Verify Entrypoint & CMD.

46) can two containers Share the same volume?

Yes, Volumes can be shared across containers by mounting them with -v.

47) what is the difference between bridge, host & none network modes?

bridge = default, isolated

host = uses host network directly

none = no networking. (full security)

48) What is the role of labels in Docker?

Add metadata to images & containers for automation or identification.

LABEL maintainer = "sowetha@example.com"

49) What is the difference between docker push and docker pull ?

Push = upload to registry

Pull = download from registry.

50) what is a Docker tag ? why is it important ?

A tag refers to a version of an image, like myapp:v1. It helps manage multiple versions. Avoid using 'latest' in Production.

51) How do you roll back a container to a previous version ?

If previous images are tagged properly, I can stop the current container & start one with the previous tag.

docker run myapp:v1.0

52) What happens if you use CMD and ENTRYPOINT together ?

CMD becomes argument to ENTRYPOINT

53) A container fails to start with " port already in use" what should you check first ?

Host Machine for port conflicts.

Because another process may already be using that port (like Apache, Nginx...)

54) you mounted a volume but the container doesn't see your files. Why ?

Could be wrong path, permissions or an Empty volume that overwrites host content.

55) What is 'docker-compose up' and 'docker-compose up --build'

docker-compose up : starts containers using Existing images (builds only if missing)

docker-compose up --build : Always builds images first, then starts containers.

Docker commands

1. docker --version : To check docker version.
2. docker info : Display Docker system information
3. docker help : Get help on docker commands
4. docker images : List all images
5. docker search <image> : Search for images on Docker Hub
6. docker pull <image> : Download image from Docker Hub
7. docker build -t <name>:<tag> -f dockerfile. : Build image from dockerfile.

8. docker tag <image> <new-name>:<tag> - Tag an image.

This command gives a new name & tag to an existing Docker image.

9. docker rmi <image> : Remove image

10. docker history <image> : View image history

This command shows the layer-by-layer history of a Docker image.

11. docker inspect<image> : Detailed info about image.

12. docker image prune : Remove unused images

13. docker save -o <file>.tar <image> :

It saves Docker image as a .tar file (archive), so we can back it up, share it with others & move it to another system.

-o <file>.tar : output filename (like myapp.tar)

14. docker load -i <file>.tar : This command loads (imports) a docker image from a .tar file that was saved using 'docker save'.

-i <file>.tar : input filename

15. docker ps : List running containers

16. docker ps -a : List all containers (Running + stopped)

17. docker run <image> : Run container from the Docker image.

18. docker run -it <image> : Run a container interactively with terminal.

19. docker run -d <image> : Run a container in detached mode.

20. docker run --name <name> <image> : Name the container.

21. docker run -p 8080:80 <image> : Map port (host : container)

22. docker run -v <host-path>:<container-path> <image> : (volume-name)
Mount volume into container.

23. docker run -v <host path>:<container path> <image> :
Bind mount
We are linking a file or folder from our system (host) to a folder inside the container.

24. docker run --env <key>=<value> <image> :
Set Environment variable inside the container at runtime.

25. docker exec -it <container> /bin/bash/ :
Execute command in container

26. docker attach <container> : Attach to running container.

27. docker logs <container> : Shows logs of a container.

28. docker stop <container> : Stop container

29. docker start <container> : Start stopped container

30. docker restart <container> : restart container

31. docker pause <container> : Pause container

32. docker unpause <container> : unpause container

33. docker rm <container> : Remove container.

34. docker rm -f <container> : forcefully deletes a container, even if it is still running.

35. docker container prune : Remove all stopped containers

36. docker rm $(docker ps -aq) : Delete all containers

37. docker rmi $(docker images -q) : Delete all images.

38. docker cp <source> <destination> : docker copy commands.

39. docker cp <container_name>:<container_path> <path on host> : (container_path: path in container)
Copy from container → host

40. docker cp /path/on/host <container_name>:/path/in/container :
copy from host → container.

41) docker volume create <name> : Create Volume

42) docker volume ls : List volumes.

43) docker volume inspect <name> : show detailed info about a volume.

44) docker volume rm <name> : Remove a volume

45) docker volume prune : Remove all unused volumes.

46) docker network ls : List all docker networks

47) docker network create <name> : create a custom network

48) docker network inspect <name> : View details of a network.

49) docker network rm <name> : Remove a network

50) docker network prune : Remove all unused networks.

51) docker network create --subnet 10.81.0.0/16 <network_name>
Create own network with particular subnet

52) docker run -it --name <container> --network <network_name>
--ip <IP> <Image_name>
Create a container with custom Ip address in particular subnet.

53) docker stats : Show container stats

54) docker top <container> : show running processes

55) docker system prune : cleanup unused data

56) docker login : Login into Docker Hub

57) docker logout : Logoot from Docker Hub

58) docker push <image> : push image to registry

59) docker pull <image> : pull image from registry

60) docker run --rm <image> : Automatically remove the container when it exits.