**Docker Questions**

Q1: What is difference between containerization and virtualization?

* **Containerization** uses shared OS kernel. Each container runs as an isolated process with its own filesystem, libraries, and dependencies. Lightweight and faster.
* Maintain different data in different containers by change port number.
* We can run same application with different version.
* **Virtualization** uses hypervisors to create multiple VMs on a host. Each VM has its own OS, making it heavier and slower than containers.

|  |
| --- |

Q2: Explain Docker architecture

1. Docker Host: where we can install the docker engine, it will be the Docker Host
2. Docker Client: where we can make the requests in docker to interact with docker daemon
3. Daemon: which will work as API in docker host
4. Docker Hub or Registry: where we can find the requested images, if it is not present in the local images

Q3: How can you check docker is installed or not?

Ans: systemctl status docker or docker –version

Q4: Which user by default can run the docker commands

Ans: Only the **root** user or any user in the **docker** group.

Q5: I have a user by name Ubuntu I want Ubuntu user to run the docker commands how can he run the docker commands

Ans: The Ubuntu user need to add to the docker group

: sudo usermod -aG docker ubuntu

Q6: List out the images in docker host

Ans: docker images

Q7: List the running containers in docker host

Ans: docker ps

Q8: What are the different statuses of docker containers

Ans: There are four statuses for docker containers, they are

* **Created** – Container is created but not started.
* **Running** – Actively running.
* **Paused** – Suspended (paused CPU, etc.).
* **Exited** – Stopped with exit code.
* **Dead** – Unusable or unresponsive container.

Q9: Create and run a new container using below details

Image: jenkins/jenkins:jdk17

Containername: Jenkins

Port: 8080

Mode: Detached mode

Ans: docker run --name jenkins -d -p 8080:8080 jenkins/jenkins:jdk17

Q10: User wants to install git inside a container by name ‘jenkins’ how can he install?

Ans: docker exec -it jenkins apt update

docker exec -it jenkins apt install git -y

Q11: How can I list the images, running, stopped and paused containers using single command?

Ans: docker ps -a or docker images && docker ps -a

Q12: How can I troubleshoot a container when it is not running?

1. **Check logs:**

docker logs <container\_name>

1. **Inspect for detailed info:**

docker inspect <container\_name>

1. **Check status or error codes:**

docker ps -a

1. **Run in interactive mode to test manually:**

docker run -it <image\_name> /bin/bash

Q13: What is persistent volume in docker.

Ans: A persistent volume is a **data storage mechanism** used to **store data outside of the container's lifecycle**,, which can store the data even if the container is deleted.

Q14: What are different volumes in docker.

Ans: In Docker we have three types of volumes, they are

1. **Named Volumes** – Created using Docker CLI (docker volume create)
2. **Anonymous Volumes** – Auto-created when you use -v /data
3. **Bind Mounts** – Maps host path to container path

|  |
| --- |

Q15: What is the use of docker-compose. Can you explain about docker-compose.yml file.

* **Docker Compose** is used to **define and run multi-container Docker applications**.
* Uses a **YAML file** to configure application services, networks, and volumes.

**Sample docker-compose.yml:**

version: "3"

services:

web:

image: nginx

ports:

- "80:80"

db:

image: mysql

environment:

MYSQL\_ROOT\_PASSWORD: example

**Run with:**

docker-compose up -d

Q16: How can we use docker commands:

| **Command** | **Purpose** |
| --- | --- |
| docker run---- | Create and start a new container |
| docker exec--- | Run commands inside a running container |
| docker ps---- | List running containers |
| docker build-- | Build an image from a Dockerfile |
| docker pull-- | Download image from Docker Hub |
| docker push-- | Upload image to Docker Hub |
| docker images- | List all images on the system |
| docker login--- | Log into Docker Hub |
| docker logout-- | Log out from Docker Hub |
| docker search-- | Search for images on Docker Hub |
| docker version | Show Docker version info |
| docker info | Show system-wide Docker info |

Q17: Write down a command to connect and disconnect a container to a host network

Ans: we cannot connect or disconnect to a host network

Q18: What is the default network in docker. What is inspect command used

Ans: Bridge Network is the default network in docker, inspect command is used to details of the container like network, ip address, …

Q19: What is the none network in docker. how can we disconnect from bridge network and connect to none network

Ans: none network is used to move the container to resting state, in none network it didn’t contain the ip address

-we can disconnect from bridge network, by using below command

: docker network disconnect bridge container\_name

-we can connect to none network, by using below command

: docker network connect host container\_name

Q20: Default location od docker, Images and containers

Ans: default location to store images in docker is: /var/lib/docker/images

default location to store containers in docker is: /var/lib/docker/containers

Q21: What is a docker image?

Ans: A Docker **image** is lightweight, standalone package containing everything needed to run an application.

It includes:

* Application code
* System tools
* Libraries
* Environment variables
* Configuration files

Q22: Why do we use following in Dockerfile

|  |  |
| --- | --- |
| FROM | Specifies the **base image** (e.g. FROM ubuntu) |
| LABEL | Metadata (e.g., version, maintainer info) |
| WORKDIR | Sets the working directory inside the container |
| USER | Specifies which user to run commands as |
| COPY | Copies files/directories from host to container |
| CMD | Sets the **default command** to run when the container starts |

Q23: I want to delete the unused networks and volumes

Ans: to delete unused networks, we use below command

: docker networks prune

to delete unused volumes, we use below command

: docker volumes prune

Q24: How can I check the resource usage of running containers.

Ans: we can check the resource usage of running containers, we can use below command

: docker stats

Q25: What is ENV in a docker file and how is it used?

* ENV sets environment variables in the container.

dockerfile

ENV APP\_ENV=production

Use case: You can access this variable inside the container or in your app logic.

Check it inside a container:

echo $APP\_ENV

Q26: Write a docker file for nodejs and java application

**Dockerfile for Node.js App:**

Dockerfile

# Base image

FROM node:18

# Set working directory

WORKDIR /app

# Copy files

COPY . .

# Install dependencies

RUN npm install

# Expose port

EXPOSE 3000

# Start the app

CMD ["npm", "start"]

✅ **Dockerfile for Java App (using JAR file):**

Dockerfile

# Base image

FROM openjdk:17

# Set working directory

WORKDIR /app

# Copy JAR file

COPY target/myapp.jar myapp.jar

# Expose port (optional)

EXPOSE 8080

# Run the JAR

CMD ["java", "-jar", "myapp.jar"]

Q27: What is dockerfile optimization and how do you implement it?

Optimization means writing **efficient Dockerfiles** to:

* Reduce image size
* Improve build time
* Avoid unnecessary layers

🔹 **Best practices:**

* Use lightweight base images (like alpine)
* Combine commands using && to reduce layers
* Use .dockerignore to exclude unnecessary files
* Only copy what's needed
* Minimize layers (combine RUN commands)

RUN apt-get update && apt-get install -y curl && rm -rf /var/lib/apt/lists/\*

Q28: Forcefully delete all the running containers and images

Ans: To delete all the images forcefully, we can use below command

: docker rmi -f $(docker images -q)

To delete all the containers forcefully, we can use below command

: docker rm -f $(docker ps -aq)

**29.Explain Docker daemon.**

Docker daemon (dockerd) is a background service that manages Docker containers. It listens for Docker API requests, manages images, containers, networks, and volumes, and interacts with the operating system.

**30 Explain about your Dockerfile.**

A **Dockerfile** is a script containing instructions to build a Docker image. A simple example:

FROM node:14

WORKDIR /app

COPY . .

RUN npm install

CMD ["node", "server.js"]

This creates an image with Node.js, copies the application files, installs dependencies, and sets the container to run server.js.

**31. How did you optimize your Dockerfile?**

Best practices for Dockerfile optimization:

* Use **multi-stage builds** to reduce image size.
* Use **alpine** images for minimal footprint (e.g., FROM node:14-alpine).
* Minimize RUN layers by combining commands.
* Use .dockerignore to exclude unnecessary files.

Example:

FROM node:14-alpine AS builder

WORKDIR /app

COPY . .

RUN npm install && npm run build

FROM node:14-alpine

WORKDIR /app

COPY --from=builder /app .

CMD ["node", "server.js"]

**32. What is the difference between Docker, Swarm, and Kubernetes?**

| **Feature** | **Docker** | **Swarm** | **Kubernetes** |
| --- | --- | --- | --- |
| Type | Container runtime | Container orchestrator | Advanced container orchestrator |
| Scalability | Limited | Moderate | High |
| Load balancing | No | Yes | Yes |
| Self-healing | No | Limited | Yes |

* **Docker**: Manages single containers.
* **Swarm**: Manages container clusters (native to Docker).
* **Kubernetes**: Advanced orchestration with automatic scaling and healing.

**33. How can you enter into a container? How can you clean unused volumes, networks, images, and containers?**

**Entering a container:**

docker exec -it <container\_id> /bin/bash

**Cleaning unused resources:**

docker system prune -a

docker volume prune

docker network prune

docker image prune

docker container prune

**34. What is the difference between CMD and ENTRYPOINT?**

* CMD sets the default command to run inside a container.
* ENTRYPOINT defines a fixed command that cannot be overridden unless specified differently at runtime.

Example:

CMD ["nginx", "-g", "daemon off;"]

ENTRYPOINT ["nginx"]

If you run:

docker run myimage /bin/sh

* If using CMD, /bin/sh runs instead of nginx.
* If using ENTRYPOINT, nginx still runs.

**35. Difference between COPY and ADD?**

| **Feature** | **COPY** | **ADD** |
| --- | --- | --- |
| Function | Copies files from host to container | Copies files + extracts archives |
| Remote URL support | No | No |
| Extracts tar files | No | Yes |

Use COPY unless you need ADD to extract compressed files.

Example:

COPY file.txt /app/ # Simple copy

ADD archive.tar.gz /app/ # Extracts files

**36. What is Docker Compose?**

Docker Compose is a tool to define and run multi-container applications using a YAML file (docker-compose.yml).

Example:

version: "3"

services:

web:

image: nginx

ports:

- "8080:80"

db:

image: postgres

environment:

POSTGRES\_USER: user

POSTGRES\_PASSWORD: pass

Start with:

docker-compose up -d

**37. How did you use Trivy and Docker Scout to scan Docker images?**

* **Trivy** is an open-source vulnerability scanner:
* trivy image myimage
* **Docker Scout** provides insights and policy compliance checks:
* docker scout quickview myimage

Both tools help identify security vulnerabilities in Docker images.