DEVOPS
Interview Questions and Answers

1. What is the difference between Virtualization and containerization?

Virtualization	Containerization
Each virtual machine (VM) runs on its own operating system, providing strong isolation.	Containers share the host OS kernel, providing process-level isolation.
VMs include the entire operating system, resulting in higher resource overhead.	Containers are lightweight, only packaging the application and its dependencies.
VMs take longer to start because they need to boot up an entire OS.	Containers start almost instantly as they don't need to boot an entire OS.
VMs have fixed resources allocated, leading to less efficient resource utilization.	Containers can dynamically allocate resources, leading to more efficient resource utilization.
Requires a hypervisor (e.g., VMware, Hyper-V, KVM) to manage and run VMs.	Requires a container engine (e.g., Docker, Podman) to manage and run containers.

2. How to do the port mapping to the Container?

To map ports between your host machine and a container, you use port mapping or port forwarding. The format is -p hostPort:containerPort.

Eg - docker run -p 8080:80 myimage

Multiple Port Mapping - docker run -p 8080:80 -p 443:443 myimage

3. How can you get shell access to a running container?

Use `docker exec` to open a shell session inside the container. Replace `container_id` or `container_name` with your container's ID or name.

Eg - docker exec -it container_id_or_name /bin/bash

4. What is the difference between RUN and CMD in Docker file?

RUN - Executes commands during the image build process. ypically used to install software or set up the environment.

Eg -
```
FROM ubuntu:20.04
RUN apt-get update && apt-get install -y curl
```

CMD - Specifies the default command to run when a container is started from the image. It is runtime commands, affecting how the container runs.

Eg -
```
FROM ubuntu:20.04
CMD ["echo", "Hello, World!"]
```

5. What is a Dockerfile?

A Dockerfile is a text file that contains a set of instructions for building a Docker image. It defines how an image is constructed, specifying the base image to use, the software to install, and any configuration needed. Docker reads the Dockerfile and executes its commands to create a custom Docker image.

6. How did you create docker compose?

Creating a Docker Compose file involves defining a `docker-compose.yml` file that specifies the services, networks, and volumes for your multi-container Docker application.

—> Install Docker Compose
—>Create a `docker-compose.yml` File
—>Define Services

Example `docker-compose.yml`

```
version: '3.8'
services:
  web:
    image: nginx:latest
```

```
    ports:
      - "8080:80"
    volumes:
      - ./html:/usr/share/nginx/html
    networks:
      - webnet
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: mydb
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - webnet
networks:
  webnet:
    driver: bridge
volumes:
  db_data:
```

7. What is the Docker namespace?

Docker namespaces are a fundamental part of Docker's containerization technology, providing isolation for containers. Each namespace provides a layer of isolation for certain aspects of the container's environment, ensuring that processes within one container do not interfere with those in another. Here are the main types of Docker namespaces:

PID Namespace: - Isolates process IDs.

Network Namespace: Isolates network interfaces, IP addresses, and routing tables.

Mount Namespace: Isolates filesystem mounts.

8. What is the lifecycle of a docker container?

The lifecycle of a Docker container involves several stages, from creation to termination.

—>Created → Starting → Running

—>Running → Stopping → Exited

—>Exited → Removing → Removed

Create: `docker create <image>` or `docker run <image>`
Start: `docker start <container>`
Stop: `docker stop <container>`
Restart: `docker restart <container>`
Remove: `docker rm <container>`

9. Why is the docker system prune used? What does it do?

The `docker system prune` command is used to clean up unused Docker resources on your system. It helps free up disk space by removing objects that are not currently in use.

Here's what it does:

—> Removes Unused Containers
—> Removes Unused Networks
—> Removes Unused Volumes
—> Removes Dangling Images/untagged
—> Removes Dangling Build Cache

Basic Command : docker system prune

Eg : docker system prune -f //force option

docker system prune --volumes

10. What is the purpose of dockerhost?

The term "dockerhost" generally refers to the host machine or server where

Docker is installed and running. It plays a crucial role in managing and executing Docker containers.

Purpose of DockerHost:

—> Running Docker Daemon

—> It provides the operating system resources (CPU, memory, disk) required to run Docker containers.

—> Image Storage:

—> Network Management

—> Volume Management

The Docker host is the physical or virtual machine where Docker is installed and operates. It provides the environment and resources needed for Docker containers to run and function. The Docker host handles container management, networking, storage, and security, enabling effective containerized application deployment and management**.**

Thank you!....