Q1 What is the primary purpose of Docker containers?

The primary purpose of Docker containers is to provide a lightweight and portable way to package, distribute, and run applications and their dependencies. Containers encapsulate an application and its dependencies, including libraries, runtime, and system tools, into a single unit. This ensures that applications run consistently across environments, from development to testing to production.

Q2 Explain the basic architecture of Docker and how it differs from traditional virtualization.

Client-server:

Docker Client: Issues commands and

communicates with the server via the REST API. It can run on the same system as the server or connect remotely.

Docker Daemon Server: Handles the heavy lifting - building, running and distributing containers. It manages images, containers, networks, and storage.

Docker objects:
Images: Templates containing instructions for creating containers applications + dependencies. Stored in registries public or private.

Containers: Isolated instances of running images. Share a host core, but have separate user space, processes, and resources.

Networks: Define how containers communicate with each other and the outside world.

Docker offers faster, lighter and more efficient containers compared to traditional virtual machines. Virtual machines provide excellent isolation, but are heavier and slower.
The choice depends on our specific needs and priorities. We can use VMs for strong isolation requirements, while Docker excels for microservices, continuous integration/continuous deployment CI/CD, and cloud deployments.

Q3. How do Docker components such as images and containers interact with each other?

1. Docker images:

Build: Docker images are built using a Dockerfile, which is a text file containing instructions for building the image. A Dockerfile specifies the base image, application code, dependencies, and configuration settings.

Docker Build: The docker build command is used to build an image based on the instructions in the Dockerfile. It creates a layered file system that represents each step of the build process. Each layer is cached and can be reused by Docker for subsequent builds as long as the instructions haven't changed.

2. Docker registry:

Push and Pull: Docker images can be stored and shared in Docker registries such as Docker Hub. Once the image is created, it can be pushed to the registry using the docker push command. Other users can then pull the image from the registry using the docker pull command.

3. Docker containers:

Run: The docker run command is used to create and run a container based on a specific image. When a container is started, it is an instance of that image and inherits the file system and configuration settings from the image.

Container lifecycle: Containers have a lifecycle that includes creation, startup, stopping, and deletion. They can also be paused and resumed. The docker start, docker stop, docker pause, and docker rm commands are used to manage the container lifecycle.

4. Interaction with containers:

Isolation: Containers provide process and file system isolation. Each container encapsulates the application and its dependencies, ensuring that it runs independently of other containers.

Network communication: Containers can communicate with each other over the network, either through exposed ports or by connecting to a shared network. Docker supports linking containers and creating custom networks for better control over communication.

5. Volume holders:

Data Persistence: Docker containers are ephemeral by default, meaning they lose data when stopped or deleted. However, Docker allows you to use volume mounts to keep data outside the container. This allows data to be shared between a host computer and a container or between containers.

6. Docker Compose:

multi-container applications: Docker Compose is a tool for defining and running multi-container Docker applications. It uses the docker-compose.yml file to specify services, networks, and bundles, allowing users to define and manage complex application stacks.

Q4. Show the steps required to install Docker on Linux.

To install docker in linux, follow these steps

1. Check the system requirements

2. update your package list with sudo apt update

3. Install docker using the following command sudo apt install docker.io

4. install dependency packages using command sudo snap install docker

5. Check the docker version by typing docker --version in the terminal

6. pull the desired docker image, eg ubuntu docker pull ubuntu -name

Q5 Explain Docker Swarm architecture, advantages and disadvantages.

Docker Swarm is a container orchestration tool that allows you to deploy and manage containerized applications within a cluster of Docker hosts.

architecture:

Manager nodes: These manage the swarm cluster, responsible for scheduling jobs, allocating resources and maintaining service availability. Manager nodes communicate with each other and worker nodes using the Raft consensus algorithm.

Worker nodes: These run containers assigned by manager nodes.

Services: A service defines a set of running jobs on a swarm that represent the components of your application. Services can use load

balancing to distribute traffic
between tasks.

Networks: Define how containers
communicate with each other
within a swarm.
Volumes: Provide persistent storage
for your containers.

advantages:

Simple and easy to use Easier to
set up and manage compared to
Kubernetes.

Lightweight and uses the same
Docker CLI and daemon, minimizing
resource overhead.

It works seamlessly with other
Docker tools like Compose.

Services can be replicated across
the swarm for fault tolerance.

Scale services up or down as needed by adding or removing worker nodes.

Distributes traffic evenly among service instances.

Disadvantages:

Compared to Kubernetes, it lacks advanced features such as self-healing deployment and rollback.

It does not offer comprehensive security features such as role-based access control RBAC.

It may not be as scalable as Kubernetes for extremely large deployments.