Scenario-based Docker interview questions Part 1☺️

1. **You need to deploy an application using Docker, but it requires different environment variables in different environments (dev, staging, prod). How would you handle this?**

**Ans. Follow below steps**

1. Create .env files for each environment like

**.env.dev**

APP\_ENV = development

DB\_HOST = dev\_db.example.com

DB\_USER = dev\_user

DB\_PASS = dev\_password

**.env.prod**

APP\_ENV = production

DB\_HOST = prod\_db.example.com

DB\_USER = prod\_user

DB\_PASS = prod\_password

1. Modify docker-compose.yaml

version: '3.8'

services:

app:

image: my-app:latest

env\_file:

- .env

1. **Load the right .env file when running**

docker compose --env-file .env.dev up -d

docker compose –env-file .env.prod up -d

1. **A developer pushes an image to a private Docker registry, but another developer is unable to pull it. What could be the issue, and how would you resolve it?**

\* Docker registry means url of private docker registry where images are stored. Developers used to pull or push images in the address

**Ans.** The most likely reason is authentication or access issues.

1.Docker login <registry -url> , enter correct login details.

2. Admin should check registry access control and ensure the user has pull access.

3. Developer extracting image that Doesn’t Exist or Wrong Tag

List of docker images : docker image ls -> docker pull <registry\_url>/<image\_name>:<tag>

1. Registry may be unreachable due to network issues or downtime.

docker ps | grep registry -> ping <registry\_url>

1. **You need to ensure that a container restarts automatically if it crashes. What options in Docker can help achieve this?**

**Ans.** Docker alone does not provide full auto-healing, which is why Kubernetes introduced.

But Docker does offer restart policies that help containers restart automatically if they crash.

docker run -d --restart=always my-app

Or we can define restart policies in docker compose with multiple tags

1. **Your application running in a container cannot connect to an external database. How would you troubleshoot and fix this?**

**Ans.** We can probably follow below steps

1. Ensure you're using the correct host and port in your app’s configuration.
2. Run a shell inside the container and check if it can reach the database.
3. If the database is on **AWS RDS, GCP, or Azure**, ensure: The **database allows incoming connections** from your container’s IP.
4. **A team member accidentally deleted a running container. How can you recover its data?**

**Ans.** We can recover its data depending on how it was stored**. There are 3 chances :**

1. Data stored in volume : Docker **volumes** persist even if a container is deleted. We can check volume -> docker volume ls

If volume is there , we can create a new container

1. If data was Stored in a Bind Mount(dir in local host), data still available.
2. If data was inside the container then we cant recover. then **it is gone forever** .
3. **A microservices-based application is running in multiple containers. How would you ensure communication between them?**

**\*** Microservices means an application is built as a collection of small, independent, and loosely coupled services, each handling a specific business function and communicating via APIs.

**Ans.**

1. Establish a custom docker network to allow containers to communicate seamlessly.

docker network create my-microservices-network.

1. When starting each container, connect it to the created network and assign a unique service name.

docker run -d --name service1 --network my-microservices-network service1-image

docker run -d --name service2 --network my-microservices-network service2-image

or we can define the service name in docker compose file

1. Within the containers, services can communicate using the assigned service names.
2. **Your containerized application logs are too verbose, and you need to collect and analyze them efficiently. What logging strategies can you implement?**

**Ans.**

1. Configure your application to emit logs at appropriate levels (e.g., DEBUG, INFO, WARN, ERROR).

2. Set up log rotation to prevent log files from consuming excessive disk space

3. Aggregate logs from all containers into a centralized logging system or service.

4. Format logs in a structured manner, such as JSON. Structured logs are easier to parse and analyze, facilitating more efficient querying and monitoring.

1. **You notice that your Docker image size is too large. How can you reduce it?**

**Ans.**

1. We can choose a Minimal Base Image
2. **Remove unnecessary files** using .dockerignore that can prevent them from being added to the image, thereby reducing its size.
3. **Implement Multi-Stage Builds:** This technique allows you to use multiple stages in your Dockerfile, keeping only the necessary artifacts in the final image.
4. **Your application requires a dependency that is not available in the official Docker image. How would you modify the image to include it?**

**Ans.**

1. Start with the Official Base Image

FROM official-image:tag

1. Install the Required Dependency:

RUN apt-get update && apt-get install -y your-dependency

1. RUN apk add --no-cache your-dependency
2. . You need to enforce resource limits on your Docker containers to prevent them from consuming too much CPU and memory. How would you do this?

Ans.

1. To limit a container's memory usage, use the --memory flag. If its exceeds then Out of Memory error will come

2. To prioritize CPU allocation among containers, use the --cpu-shares flag.

**11. You have two services in separate Docker networks that need to communicate. How would you enable this communication?**

**Ans** Connect both containers to the same network:

docker network connect my-network container1

12. **You need to expose a containerized service on a specific port and make it accessible to external users. How would you achieve this?**

Ans. Use the docker run command with the -p flag to map a port on your host to a port in the container.

For example, if your application inside the container listens on port 80 and you want to expose it on port 8080 of the host:

docker run -d -p 8080:80 your-image

**13. A containerized database application needs persistent storage for its data. How would you configure this?**

Ans. We can store data in /volume or Bind Mounts

14**. You need to share data between multiple containers. What Docker features can help you?**

Ans. We will store data in volume or bind mounts.

**15. Your containerized application is experiencing network latency when communicating with other services. How would you debug and optimize this?**

Ans. Inspect network performance

docker network inspect my-network or

Assign adequate CPU and memory resources to prevent contention that could impact network performance.

**16.** You suspect a container is consuming too many resources. How would you monitor and identify the issue?

Ans Check real-time stats: docker stats

**17. A security team asks you to scan Docker images for vulnerabilities before deployment. How would you do this?**

Ans. docker scan my-image

**18. Your production containers need to run with minimal privileges. What best practices should you follow?**

Ans. Run as non-root user -> USER appuser

* Use **read-only file systems**: docker run --read-only my-app

**19.** An attacker has gained access to a running container. How can you isolate and investigate the breach?

* Pause the container: docker pause container\_id
* Inspect container activity: docker logs container\_id

**20. You need to prevent unauthorized access to the Docker daemon API. How can you secure it?**

Ans. Disable unauthenticated remote access: sudo systemctl edit docker

Or Restrict access with firewall rules.