**Docker Swarm Interview Questions and Answers**

**Basic Questions**

**1. What is Docker Swarm?**

**Answer:**  
Docker Swarm is a **container orchestration tool** that allows multiple Docker hosts to work together as a **single virtual cluster**. It provides **scalability, load balancing, service discovery, rolling updates, and fault tolerance** for containerized applications.

**2. How is Docker Swarm different from Kubernetes?**

**Answer:**

| **Feature** | **Docker Swarm** | **Kubernetes** |
| --- | --- | --- |
| Simplicity | Easy to set up | More complex |
| Scaling | Fast scaling | Advanced scaling |
| Load Balancing | Built-in, automatic | External required |
| Service Discovery | Built-in | Built-in |
| Rolling Updates | Yes | Yes |

Kubernetes is more **feature-rich**, while Swarm is **lightweight and easy to use**.

**3. What are nodes in Docker Swarm?**

**Answer:**  
Nodes are **individual Docker hosts** in a Swarm cluster. There are **two types** of nodes:

* **Manager Node** – Controls the cluster, schedules tasks, and manages services.
* **Worker Node** – Executes tasks assigned by the manager.

**4. How do you initialize a Docker Swarm cluster?**

**Answer:**  
Run the following command on the **manager node**:

docker swarm init --advertise-addr <manager-ip>

**5. What command is used to join a node to a Swarm cluster?**

**Answer:**  
On the **worker node**, run:

docker swarm join --token <worker-token> <manager-ip>:2377

To get the worker token:

docker swarm join-token worker

**Intermediate Questions**

**6. What is a service in Docker Swarm?**

**Answer:**  
A service is a **logical set of containers** running the same application. It defines how a container should run in the Swarm cluster.

Example command to create a service:

docker service create --name web -p 80:80 nginx

**7. What is the difference between docker service and docker stack?**

**Answer:**

* docker service is used for **managing individual services** in Swarm.
* docker stack is used for **deploying multiple services** using a docker-compose.yml file.

**8. How do you scale a service in Docker Swarm?**

**Answer:**  
Use the scale command:

docker service scale web=5

or update the service:

docker service update --replicas 5 web

**9. How does Docker Swarm handle load balancing?**

**Answer:**  
Swarm automatically **distributes traffic** between all replicas of a service using **ingress load balancing**. Requests are routed evenly to all available containers.

**10. How do you remove a node from a Swarm cluster?**

**Answer:**  
First, **drain** the node so no services run on it:

docker node update --availability drain <node-id>

Then, remove the node:

docker node rm <node-id>

**Advanced Questions**

**11. How does Docker Swarm handle rolling updates?**

**Answer:**  
Swarm updates services **gradually** to prevent downtime using:

docker service update --update-delay 10s --update-parallelism 2 --image nginx:latest web

This updates **2 replicas at a time** with a **10-second delay**.

**12. What happens if a manager node fails?**

**Answer:**

* If there are multiple manager nodes, the **Raft consensus algorithm** ensures failover.
* If there's only **one manager**, the cluster becomes **unavailable**.

**13. How can you persist data in Docker Swarm services?**

**Answer:**  
Use **Docker volumes**:

docker service create --name db --mount type=volume,source=mydata,target=/var/lib/mysql mysql

**14. What is the difference between replicated and global services?**

**Answer:**

* **Replicated Service** – Runs **a specific number of replicas** across worker nodes.
* **Global Service** – Runs **one instance per node**.

Example of **replicated service** (3 replicas):

docker service create --replicas 3 nginx

Example of **global service**:

docker service create --mode global nginx

**15. How does Swarm handle secret management?**

**Answer:**  
Swarm allows secure storage of **secrets** like passwords and API keys:

echo "mysecretpassword" | docker secret create db\_password -

docker service create --secret db\_password mysql

**16. How do you monitor a Swarm cluster?**

**Answer:**  
Use:

docker service ps <service-name>

docker node ls

docker service logs <service-name>

For advanced monitoring, tools like **Prometheus**, **Grafana**, and **cAdvisor** can be used.

**17. What is the Raft consensus algorithm in Swarm?**

**Answer:**  
Raft ensures **data consistency** across manager nodes by maintaining a distributed log. If a leader fails, another manager is elected.

**18. How can you rebalance services in Swarm?**

**Answer:**  
Force an update to redistribute tasks:

docker service update --force <service-name>

**19. How do you handle logging in a Swarm cluster?**

**Answer:**  
Use:

docker service logs <service-name>

For centralized logging, use **ELK Stack (Elasticsearch, Logstash, Kibana)** or **Fluentd**.

**20. What are the limitations of Docker Swarm compared to Kubernetes?**

**Answer:**

* **Limited auto-scaling** (Kubernetes has advanced scaling).
* **Fewer network policies**.
* **Less extensibility** (Kubernetes supports more plugins).
* **Weaker fault tolerance**.

**Docker Swarm Commands (Basic to Advanced)**

**1. Basic Commands**

docker swarm init # Initialize swarm

docker swarm join --token <token> <ip>:2377 # Join a node

docker node ls # List all nodes

docker service ls # List running services

docker service ps <service> # Check service tasks

docker swarm leave # Leave the swarm

**2. Service Management**

docker service create --name web -p 80:80 nginx # Create a service

docker service update --replicas 5 web # Scale service

docker service rm <service> # Remove service

**3. Stack Management**

docker stack deploy -c docker-compose.yml my\_stack # Deploy stack

docker stack ls # List stacks

docker stack services my\_stack # List stack services

docker stack rm my\_stack # Remove stack

**4. Advanced Swarm Commands**

docker node promote <worker-node-id> # Promote worker to manager

docker node demote <manager-node-id> # Demote manager to worker

docker node update --availability drain <node-id> # Drain a node

docker service update --update-delay 10s --update-parallelism 2 --image nginx:latest web # Rolling update

docker swarm join-token --rotate worker # Rotate tokens

docker node ps <node-id> # Check tasks on node

**Conclusion**

Mastering **Docker Swarm commands, architecture, and troubleshooting** is essential for managing containerized applications. If you need help with Swarm deployment or CI/CD integration, let me know! 🚀

4o

**You said:**

when to use docker compose and when to use docker swarm

**ChatGPT said:**

**Docker Compose vs. Docker Swarm: When to Use Each?**

| **Feature** | **Docker Compose 🐳** | **Docker Swarm 🐝** |
| --- | --- | --- |
| **Purpose** | Define and run **multi-container applications** on a **single machine** | Manage **container orchestration** across **multiple machines** |
| **Scalability** | Limited to **one machine** | **High scalability** across multiple nodes |
| **Networking** | Uses bridge network by default | Uses **overlay networking** for inter-node communication |
| **Load Balancing** | Not built-in, requires **external** tools like Nginx | **Built-in** load balancing |
| **High Availability** | Not supported (single machine) | **Yes** (automatic failover with multiple manager nodes) |
| **Rolling Updates** | Manual recreation of containers required | **Supported** (automated rolling updates) |
| **Fault Tolerance** | No redundancy, if the machine fails, services stop | **Yes**, services automatically restart on available nodes |
| **Service Discovery** | Containers communicate via container names | **Built-in service discovery** |
| **Use Case** | Local development, small projects | Production, large-scale applications |

**When to Use Docker Compose?**

👉 **Best for local development and testing**  
✅ When you need to **define and run multi-container applications** on a **single machine**  
✅ When working on a **small to medium-sized** application  
✅ When deploying a simple **microservices-based** app that doesn't require clustering  
✅ When testing applications before deploying to a production cluster

💡 **Example Use Case**:  
A local development environment with a web server, database, and caching service:

yaml

version: '3'

services:

web:

image: nginx

ports:

- "80:80"

db:

image: mysql

environment:

MYSQL\_ROOT\_PASSWORD: secret

Run with:

docker-compose up -d

**When to Use Docker Swarm?**

👉 **Best for production and high-availability environments**  
✅ When deploying a **scalable, distributed application** across multiple machines  
✅ When you need **high availability** (auto-failover, redundancy)  
✅ When you want **rolling updates and rollback capabilities**  
✅ When you require **built-in service discovery and load balancing**

💡 **Example Use Case**:  
Deploying an app across multiple nodes with replication:

docker swarm init

docker service create --name web -p 80:80 --replicas 3 nginx

## ****Docker Swarm Commands (Basic to Advanced)****

### ****1. Basic Commands****

#### ****Initialize Docker Swarm****

docker swarm init

or specify an IP for the manager node:

docker swarm init --advertise-addr <manager-ip>

#### ****Check Swarm Status****

docker info

docker node ls

#### ****Get the Join Token for Worker Nodes****

docker swarm join-token worker

#### ****Join a Node to Swarm as a Worker****

docker swarm join --token <worker-token> <manager-ip>:2377

#### ****Leave the Swarm (for a Worker Node)****

docker swarm leave

#### ****Remove a Node from Swarm (Run on Manager)****

docker node rm <node-id>

### ****2. Service Management Commands****

#### ****Create a Service****

docker service create --name web -p 80:80 nginx

#### ****List Services****

docker service ls

#### ****Inspect a Service****

docker service inspect <service-name>

#### ****Check Service Logs****

docker service logs <service-name>

#### ****Scale a Service****

docker service scale web=5

#### ****Remove a Service****

docker service rm <service-name>

### ****3. Stack Management Commands****

#### ****Deploy a Stack****

docker stack deploy -c docker-compose.yml my\_stack

#### ****List Running Stacks****

docker stack ls

#### ****Check Services in a Stack****

docker stack services my\_stack

#### ****Remove a Stack****

docker stack rm my\_stack

### ****4. Advanced Swarm Commands****

#### ****Promote a Worker to a Manager****

docker node promote <worker-node-id>

#### ****Demote a Manager to a Worker****

docker node demote <manager-node-id>

#### ****Drain a Node (for Maintenance)****

docker node update --availability drain <node-id>

#### ****Change a Node’s Availability to Active****

docker node update --availability active <node-id>

#### ****Check Running Tasks on a Node****

docker node ps <node-id>

#### ****Force Remove a Failed Node****

docker node rm --force <node-id>

#### ****Update a Running Service****

docker service update --image nginx:latest web

#### ****Perform a Rolling Update****

docker service update --update-delay 10s --update-parallelism 2 --image nginx:latest web

#### ****Rebalance Swarm Services****

docker service update --force <service-name>

#### ****Check Swarm Raft Logs (For Debugging)****

docker swarm raft log

#### ****Rotate Swarm Join Tokens****

docker swarm join-token --rotate worker

docker swarm join-token --rotate manager