Docker Kubernetes Swarm Guide

Need details guide on the following to crack interview and practice as beginner to expert
Docker Mastery: with Kubernetes+Swarm complete Hands on guide
Course Introduction
The Best Way to Setup Docker for Your OS
Installations ,Commands
Creating and Using Containers Like a Boss
Container Images, Where To Find Them and How To Build Them
Container Lifetime & Persistent Data: Volumes, Volumes, Volumes
Making It Easier with Docker Compose: The Multi-Container Tool
Swarm Intro and Creating a 3-Node Swarm Cluster
Swarm Basic Features and How to Use Them In Your Workflow
Swarm App Lifecycle
Container Registries: Image Storage and Distribution
Docker in Production
The What and Why of Kubernetes
Kubernetes Architecture and Install
Your First Pods
Inspecting Kubernetes Resources
Exposing Kubernetes Ports
Kubernetes Management Techniques
Moving to Declarative Kubernetes YAML
Your Next Steps and The Future of Kubernetes
Automated CI Workflows
GitHub Actions Workflow Examples
Docker Security Good Defaults and Tools
Docker 19.03 Release New Features
DevOps and Docker Clips
Dockerfiles and Docker Images in 2022
Dockerfile and Compose File Reviews
Extra's, Common Questions, and Resources

Below is a **complete beginner → expert → interview-ready roadmap** for **Docker + Kubernetes + Swarm**, designed exactly for **hands-on learning + real interview**

preparation.
This is written assuming **you are a Java / Spring Boot / Microservices developer**, so examples and practice align with real projects.

🚀 Docker Mastery with Kubernetes & Swarm

Beginner → Advanced → Interview Cracking Guide

🧭 How to Use This Guide (Important)

For **each topic**, follow this order:

1. **Concept (Why + What)**
2. **Hands-on Practice (Commands & Labs)**
3. **Real-World Use Case**
4. **Interview Questions (Must-Know)**

Do **daily 1.5–2 hours** → You'll be interview-ready in **30–40 days**.

🐳 PART 1: DOCKER FOUNDATIONS (BEGINNER → INTERMEDIATE)

1️⃣ Course Introduction (Docker Basics)

🔹 What is Docker?

Docker is a **containerization platform** that packages:

- Application
- Dependencies
- Runtime
- Config

➡ Runs **consistently across environments**

🔹 Why Docker?

Problem	Docker Solution
“Works on my machine”	Same container everywhere
Heavy VMs	Lightweight containers
Slow deployments	Fast image-based deployment

🔹 Docker vs VM (Interview Favorite)

Docker	VM
Shares OS kernel	Full OS
Fast startup	Slow
Lightweight	Heavy

2️⃣ Best Way to Setup Docker (Your OS)

🔹 Installation

- **Windows / Mac** → Docker Desktop
- **Linux (Ubuntu)** → Docker Engine

🔹 Verify

```bash
docker --version
docker info
docker run hello-world
```

🎯 Interview Tip

> Docker Desktop includes Docker Engine + CLI + Compose + Kubernetes (optional)

3️⃣ Docker Installations & Core Commands

🔹 Essential Commands (MEMORIZE)

```bash
docker pull nginx
docker images
docker run nginx
docker ps
docker ps -a
docker stop <container>
docker rm <container>
docker rmi <image>
```

🔹 Run with Port

```bash
docker run -d -p 8080:80 nginx
```

➡ Access: `http://localhost:8080`

🎯 Interview Questions

- Difference between `docker run` vs `docker start`
- What does `-d` do?
- What does `-p host:container` mean?

4️⃣ Creating and Using Containers Like a Boss 😎

🔹 Interactive Mode

```bash
docker run -it ubuntu bash
```

🔹 Exec into Running Container

```bash
docker exec -it <container_id> bash
```

🔹 Logs

bash

```
docker logs <container_id>
```

🔹 Resource Monitoring

bash

```
docker stats
```

5️⃣ Container Images – Build Your Own

🔹 Dockerfile (Spring Boot Example)

dockerfile

```
FROM openjdk:17
WORKDIR /app
COPY target/app.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","app.jar"]
```

🔹 Build Image

bash

```
docker build -t springboot-app .
```

🔹 Run

bash

```
docker run -p 8080:8080 springboot-app
```

🎯 Interview Questions

- What is a Docker image?
- Difference between `COPY` vs `ADD`
- What is `ENTRYPOINT` vs `CMD`?

6️⃣ Container Lifetime & Persistent Data (VOLUMES)

🔹 Problem

Containers are **ephemeral** → data lost when container stops.

🔹 Solution → Volumes

```bash
docker volume create myvol
docker run -v myvol:/data ubuntu
```

🔹 Bind Mount

```bash
docker run -v /host/path:/container/path nginx
```

🎯 Interview Questions

- Volume vs Bind Mount
- Why volumes preferred in production?

7️⃣ Docker Compose (Multi-Container Apps)

🔹 Why Compose?

Run **multiple containers together** (App + DB)

🔹 docker-compose.yml

```yaml
version: "3.8"
services:
  app:
    image: springboot-app
    ports:
      - "8080:8080"
    depends_on:
      - db
  db:
```

```
  image: mysql
  environment:
    MYSQL_ROOT_PASSWORD: root
```

🔹 Run

```bash
docker-compose up -d
```

🎯 Interview Questions

- Purpose of `depends_on`
- Difference between Compose & Kubernetes?

🐝 PART 2: DOCKER SWARM (INTERMEDIATE)

8️⃣ Swarm Intro & 3-Node Cluster

🔹 What is Swarm?

Docker's **native orchestration** tool.

🔹 Init Swarm

```bash
docker swarm init
```

🔹 Join Nodes

```bash
docker swarm join --token <token> <manager-ip>
```

🔹 Nodes

```bash
docker node ls
```

9️⃣ Swarm Services & Scaling

```bash
docker service create --name web -p 80:80 nginx
docker service scale web=5
```

🔹 Features

- Load balancing
- Auto rescheduling
- Rolling updates

🎯 Interview Questions

- Docker Swarm vs Kubernetes
- What is a service vs container?

🔟 Swarm App Lifecycle

```bash
docker service update --image nginx:latest web
docker service rm web
```

☸️ PART 3: KUBERNETES (INTERMEDIATE → ADVANCED)

1️⃣1️⃣ What & Why Kubernetes?

🔹 Kubernetes Solves

- Auto scaling
- Self healing
- Load balancing
- Rolling updates

🔹 Kubernetes vs Docker

> Docker = container runtime
> Kubernetes = container orchestrator

1️⃣2️⃣ Kubernetes Architecture

🔹 Master Node

- API Server
- Scheduler
- Controller Manager
- etcd

🔹 Worker Node

- kubelet
- kube-proxy
- Container Runtime

🎯 **VERY IMPORTANT FOR INTERVIEWS**

1️⃣3️⃣ Your First Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx
```

```bash
kubectl apply -f pod.yaml
kubectl get pods
```

1️⃣4️⃣ Inspecting Kubernetes Resources

```bash
kubectl describe pod nginx-pod
kubectl logs nginx-pod
kubectl exec -it nginx-pod -- bash
```

1️⃣5️⃣ Exposing Kubernetes Ports

🔹 Service Types

- ClusterIP
- NodePort
- LoadBalancer

```yaml
kind: Service
type: NodePort
```

1️⃣6️⃣ Kubernetes Management Techniques

- Labels
- Selectors
- Namespaces
- Rolling Updates
- Health Probes

```yaml
livenessProbe
readinessProbe
```

🎯 Interview Favorite Topic

1️⃣7️⃣ Declarative Kubernetes YAML

🔹 Deployment Example

```yaml
kind: Deployment
spec:
  replicas: 3
```

```bash
kubectl rollout status deployment/app
kubectl rollout undo deployment/app
```

1️⃣8️⃣ Kubernetes Next Steps & Future

- Helm
- Operators
- GitOps (ArgoCD)
- Service Mesh (Istio)

🔁 PART 4: CI/CD + SECURITY

1️⃣9️⃣ Automated CI Workflows

🔹 Flow

```css
Code → Build → Docker Image → Push → Deploy
```

2️⃣0️⃣ GitHub Actions (Example)

```yaml
jobs:
  build:
```

```
steps:
  - uses: actions/checkout@v3
  - run: docker build -t app .
```

2️⃣1️⃣ Docker Security (Very Important)

- Non-root user
- Small base images
- Image scanning
- Secrets management

🎯 Interview:

> How do you secure Docker images?

2️⃣2️⃣ Docker 19.03 & Modern Docker

- BuildKit
- Multi-stage builds
- Rootless Docker

2️⃣3️⃣ Dockerfile & Compose Reviews

🔹 Best Practices

- Multi-stage builds
- Avoid latest tag
- Use `.dockerignore`

2️⃣4️⃣ Extras: Common Interview Questions

🔥 Must-Prepare Questions

1. Docker vs Kubernetes
2. Pod vs Deployment
3. Service types
4. Volume vs PVC

5. Swarm vs Kubernetes
6. Rolling update vs recreate
7. Liveness vs Readiness

📅 30-Day Study Plan (Optional)

If you want, I can give you:

✅ **Daily plan**
✅ **Hands-on labs**
✅ **Mock interview questions**
✅ **Real project (Spring Boot + Docker + K8s)**

❤️ Final Advice

You already have **Java + Microservices background** — Docker & Kubernetes will **multiply your value** in interviews.

If you want:

👉 **Interview Q&A**
👉 **Real-time scenario questions**
👉 **Mini projects**
👉 **Cheat sheets**

Just tell me 👍