Docker Swarm

Contents

1. Docker Swarm

Introduction

- **Definition**: Docker Swarm is a native clustering tool for Docker that converts multiple Docker hosts into a single virtual host.
- **Purpose**: Manages and scales containerized applications across multiple hosts.
- **Comparison**: Alternative to Kubernetes but less commonly used in production due to limited features.

Key Features

1. **Clustering**: Groups multiple Docker hosts into a single cluster for easier management.
    - A **cluster** is a group of interconnected systems or nodes (physical or virtual machines) that work together to perform tasks as a single unit. In the context of **Docker Swarm**, a cluster consists of multiple Docker hosts (machines running Docker) that collaborate to manage and scale containerized applications efficiently.
    - **Docker hosts** are machines (physical or virtual) that run **Docker Engine** and are capable of hosting and managing **containers**. These machines are part of a **cluster** in Docker Swarm or other container orchestration platforms.
    - Imagine you have 3 machines:
        i. Machine A (Docker host): Runs Docker Engine and hosts 5 containers for a web application.
        ii. Machine B (Docker host): Hosts 3 containers for a database service.
        iii. Machine C (Docker host): Acts as the manager node in Docker Swarm and oversees the tasks and resources across all Docker hosts.
2. **Service Definition**: Allows defining tasks as single or replicated containers, scalable on demand.
3. **Load Balancing**: Distributes network traffic across available containers.
4. **Service Discovery**: Automatically identifies and integrates nodes joining or leaving the cluster.

5. **Rolling Updates**: Updates services with minimal downtime by gradually updating containers.
6. **High Availability**: Ensures services remain available even if nodes fail.

Architecture

- **Swarm Manager**:
  - Acts as the brain of Docker Swarm.
  - Responsibilities: Scheduling services, monitoring node health, scaling services, and service discovery.
  - Elections are conducted to select a leader among multiple managers.
- **Worker Nodes**:
  - Execute commands received from the Swarm Manager.
  - Run containers without scheduling or orchestration capabilities.
- **Docker Daemon**:
  - Runs on each node and manages container creation and operation.

Cluster Management

- **Node Types**: Manager and Worker.
- **Manager Nodes**:
  - Recommended to have an odd number for quorum maintenance.
  - Can promote/demote nodes between manager and worker roles.
- **Worker Nodes**:
  - Execute tasks but do not participate in scheduling.
- **States of Nodes**:
  - Active: Accepts new tasks.
  - Drain: Prevents new tasks; existing tasks migrate to other nodes.
  - Pause: Halts new tasks without migrating existing tasks.

Advantages

- Simplistic setup and management.
- Basic fault tolerance and scalability.
- Rolling updates for service upgrades without downtime.

Limitations

- Limited scalability and customization compared to Kubernetes.
- Basic fault tolerance and security features.
- Not suited for large-scale or complex applications.

Use Cases

- Ideal for smaller applications, internal projects, or Docker certification preparation.

Key Commands

- **Initialize Swarm**:

```
docker swarm init
```

- **Join Nodes**:

```
docker swarm join --token <token> <manager-IP>:<port>
```

- **List Nodes**:

```
docker node ls
```

- **Create Service**:

```
docker service create --replicas <num> <image>
```

- **Promote Node**:

```
docker node promote <node-ID>
```

- **Demote Node**:

```
docker node demote <node-ID>
```

- **Remove Node**:

```
docker node rm <node-ID>
```

Docker Swarm vs Kubernetes

Feature	Docker Swarm	Kubernetes
Ease of Use	Simple setup	Complex setup
Fault Tolerance	Basic	Advanced (self-healing)
Scalability	Limited	Highly scalable
Load Balancing	Built-in	Built-in + external tools
Security	Basic (TLS)	Advanced (RBAC, policies)
Deployment	Rolling updates	Rolling, Canary, Blue-Green