Docker and Kubernetes

Session 6

Docker Swarm

Agenda

- ➢ Introduction to Docker Swarm
- ➢ Set up docker swarm
- ➢ Deploy a stack to a Swarm

Swarm mode overview

Feature highlights

- Cluster management integrated with Docker Engine
- Decentralized design
- Declarative service model
- Scaling
- Desired state reconciliation
- Multi-host networking
- Service discovery
- Load balancing

What is Swarm

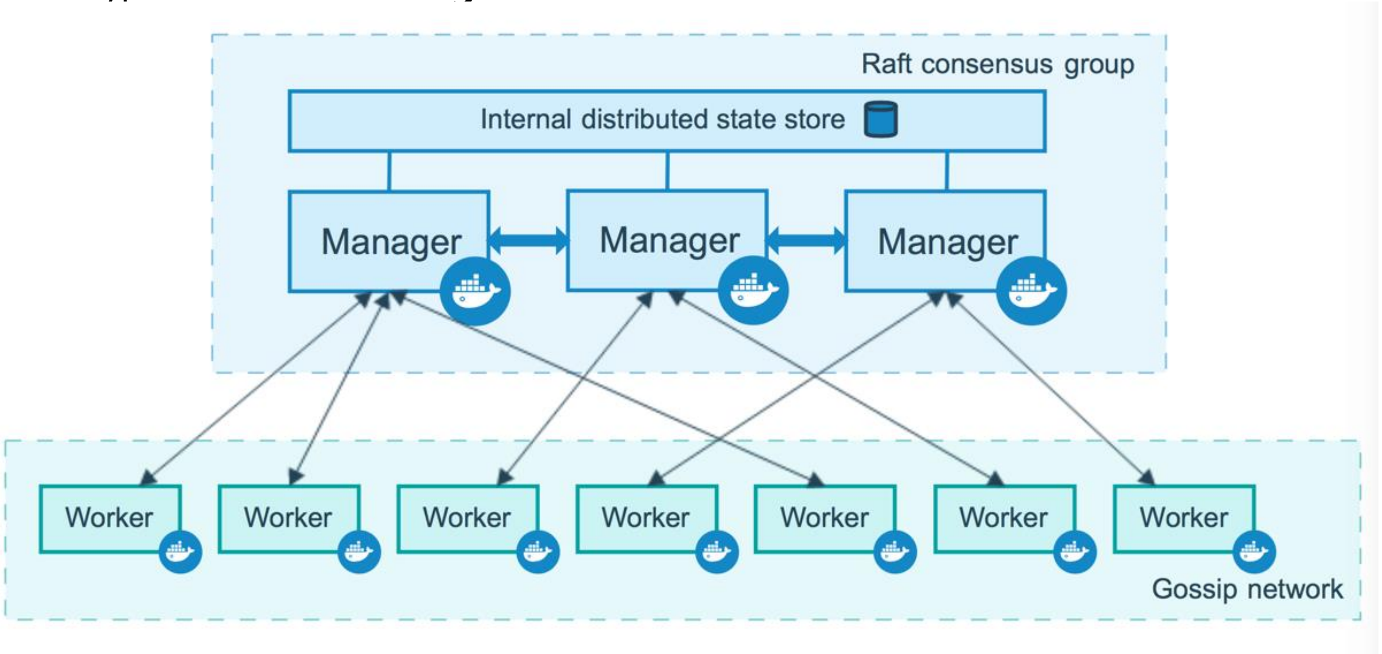
A **swarm** consists of multiple Docker hosts which run in swarm mode and act as managers (to manage membership and delegation) and workers (which run swarm services). A given Docker host can be a manager, a worker, or perform both roles

One of the key advantages of **swarm** services over standalone containers is that you can modify a service's configuration, including the networks and volumes it is connected to, without the need to manually restart the service. Docker will update the configuration, stop the service tasks with the out of date configuration, and create new ones matching the desired configuration.

Nodes

A **node** is an instance of the Docker engine participating in the swarm. You can also think of this as a Docker node. You can run one or more nodes on a single physical computer or cloud server, but production swarm deployments typically include Docker nodes distributed across multiple physical and cloud machines.

There are two types of nodes: **managers** and **workers**



Manager nodes

Manager nodes handle cluster management tasks:

- ❖ maintaining cluster state
- ❖ scheduling services
- ❖ serving swarm mode HTTP API endpoints

To take advantage of swarm mode's fault-tolerance features, Docker recommends you implement an **odd number** of nodes according to your organization's high-availability requirements. When you have multiple managers you can recover from the failure of a manager node without downtime.

Worker nodes

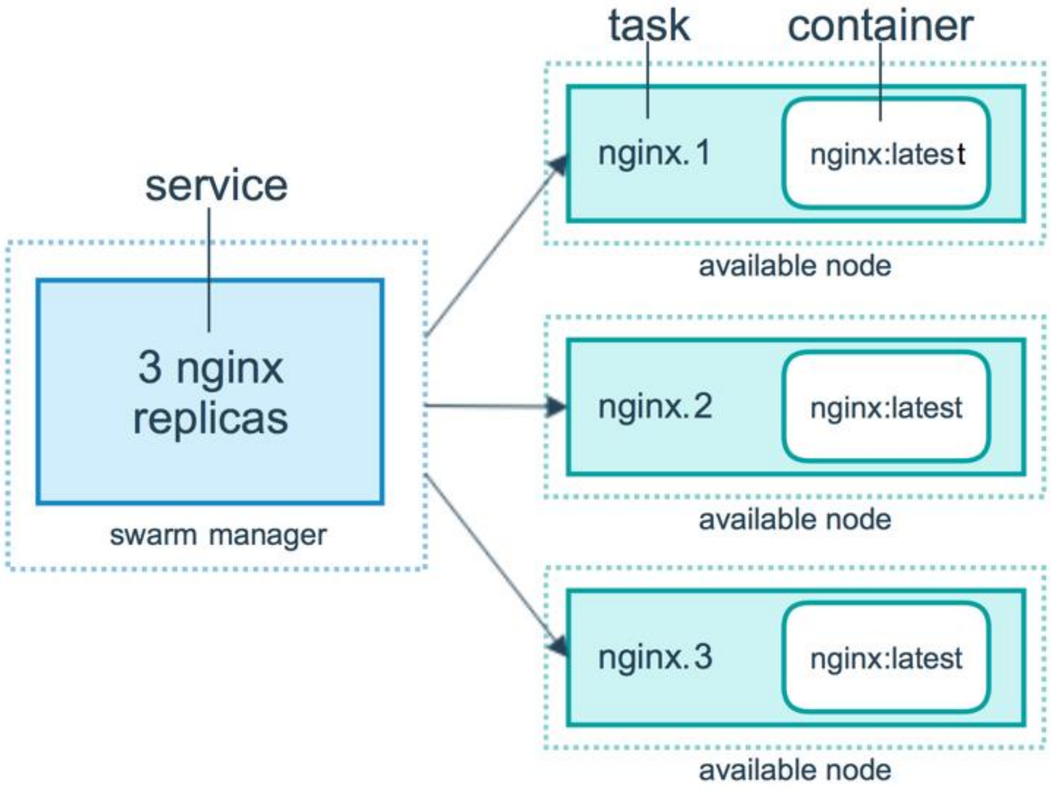
Worker nodes are also instances of Docker Engine whose sole purpose is to execute containers. Worker nodes don’t participate in the Raft distributed state, make scheduling decisions, or serve the swarm mode HTTP API.

You can promote a worker node to be a manager by running **docker node promote**. For example, you may want to promote a worker node when you take a manager node offline for maintenance

Services, tasks, and containers

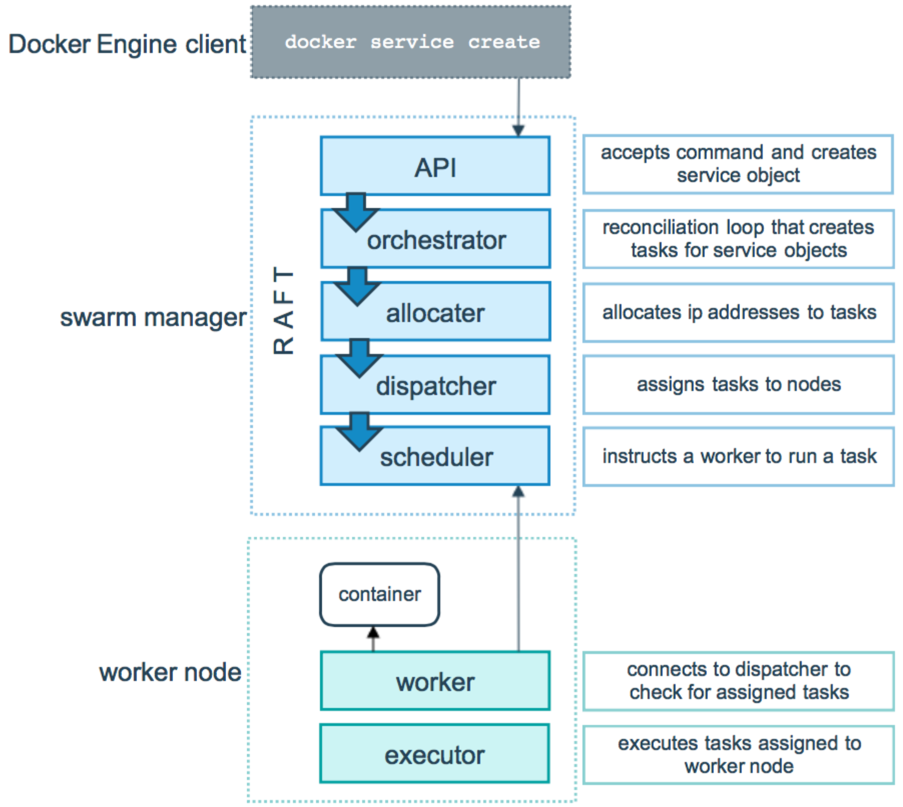
When you deploy the **service** to the swarm, the swarm manager accepts your service definition as the desired state for the service.

Then it schedules the service on nodes in the swarm as one or more replica tasks. The tasks run independently of each other on nodes in the swarm.



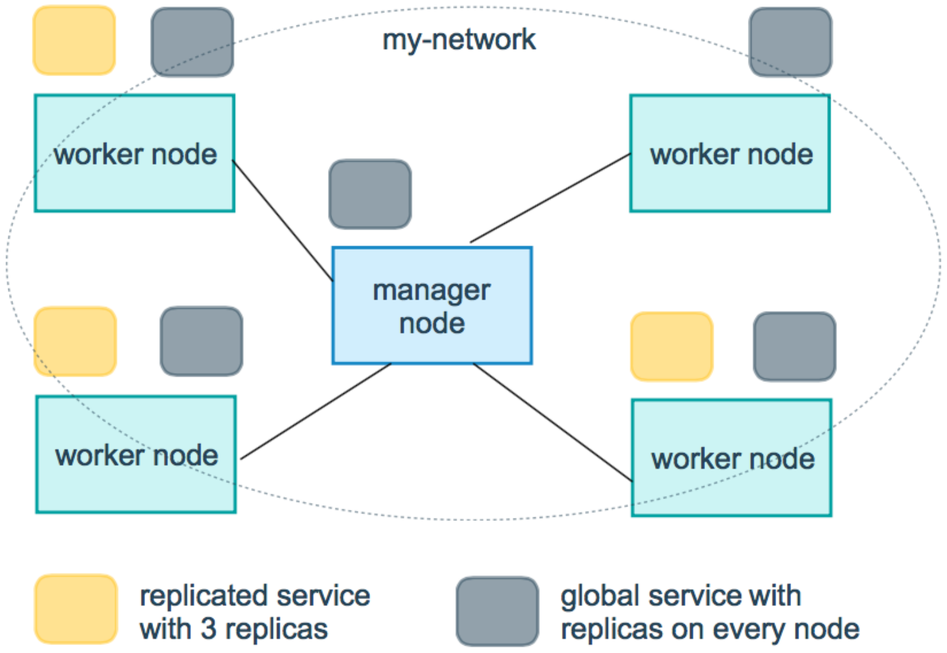
Tasks and scheduling

A **task** is the atomic unit of scheduling within a swarm. When you declare a desired service state by creating or updating a service, the orchestrator realizes the desired state by scheduling tasks


Docker Engine client
docker service create
swarm manager
RAFT
API
orchestrator
allocater
dispatcher
scheduler
accepts command and creates service object
reconciliation loop that creates tasks for service objects
allocates ip addresses to tasks
assigns tasks to nodes
instructs a worker to run a task
worker node
container
worker
executor
connects to dispatcher to check for assigned tasks
executes tasks assigned to worker node

Replicated and global services

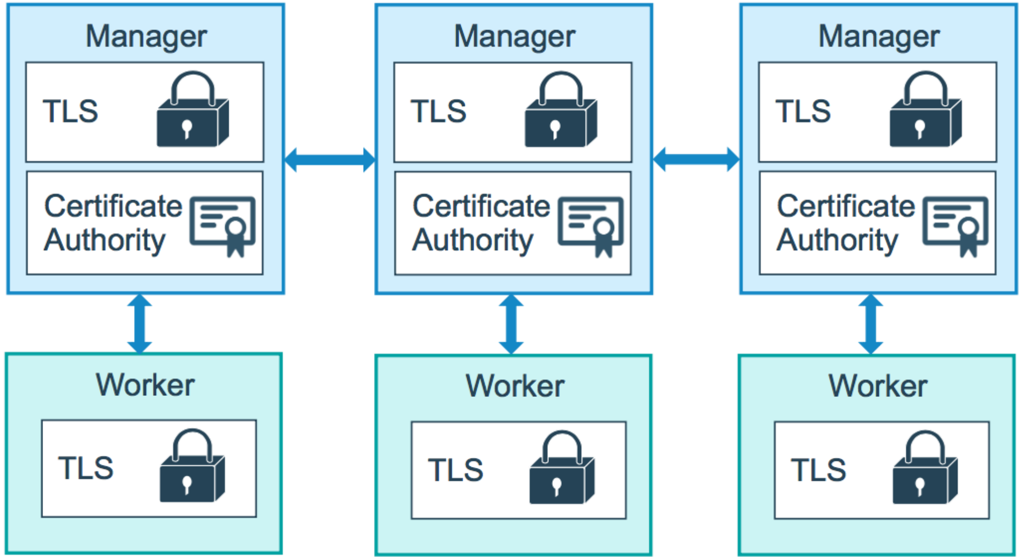
There are two types of service deployments, **replicated** and **global**.


my-network
worker node
worker node
manager node
worker node
worker node
replicated service with 3 replicas
global service with replicas on every node

Manage swarm security with public key infrastructure (PKI)

The swarm mode **public key infrastructure (PKI)** system built into Docker makes it simple to securely deploy a container orchestration system.

The nodes in a swarm use **mutual Transport Layer Security (TLS)** to authenticate, authorize, and encrypt the communications with other nodes in the swarm.



Run Docker Engine in swarm mode

When you first install and start working with Docker Engine, swarm mode is disabled by default. When you enable swarm mode, you work with the concept of services managed through the docker service command.

There are two ways to run the Engine in swarm mode:

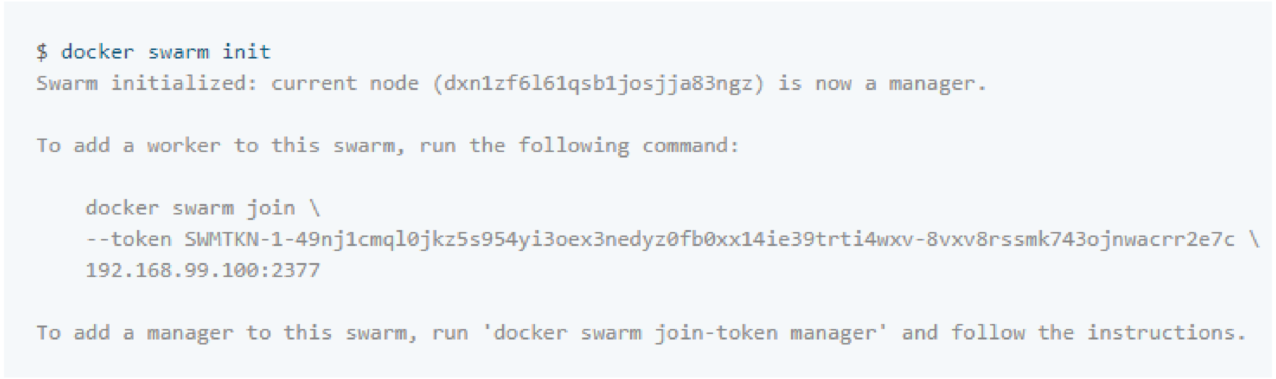
- Create a new swarm, covered in this article.
- Join an existing swarm.

Create a new swarm

When you run the command to create a swarm, the Docker Engine starts running in swarm mode.

Run docker **swarm init** to create a single-node swarm on the current node

The output for **docker swarm init** provides the connection command to use when you join new worker nodes to the swarm:



```
$ docker swarm init
Swarm initialized: current node (dxn1zf6l61qsb1josjja83ngz) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \
    192.168.99.100:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

Join node to swarm as worker node

To retrieve the join command including the join token for worker nodes, run the following command on a manager node:

```
$ docker swarm join-token worker

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \
    192.168.99.100:2377
```

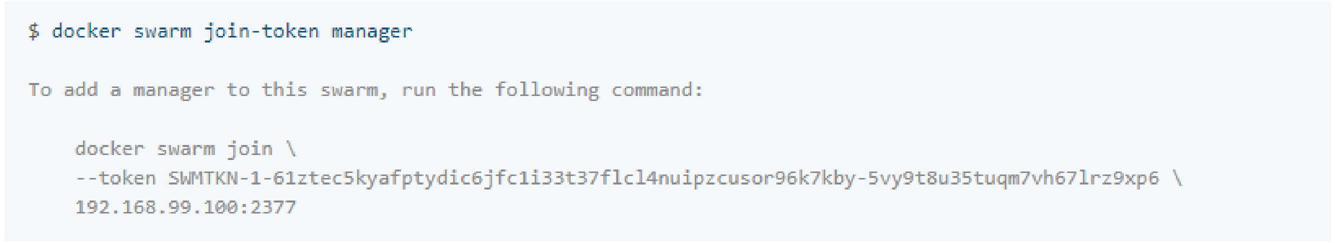
Run the command from the output on the worker to join the swarm:

```
$ docker swarm join \
  --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \
  192.168.99.100:2377

This node joined a swarm as a worker.
```

Join node to swarm as master node

To retrieve the join command including the join token for manager nodes, run the following command on a manager node:

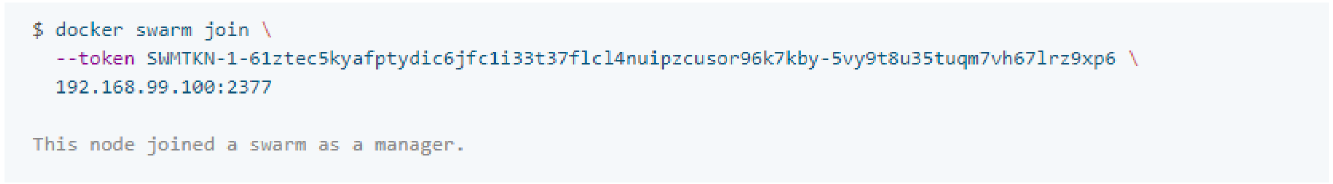


```
$ docker swarm join-token manager

To add a manager to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-61ztec5kyafptydic6jfc1i33t37flcl4nuipzcusor96k7kby-5vy9t8u35tuqm7vh67lrz9xp6 \
    192.168.99.100:2377
```

Run the command from the output on the new manager node to join it to the swarm:



```
$ docker swarm join \
  --token SWMTKN-1-61ztec5kyafptydic6jfc1i33t37flcl4nuipzcusor96k7kby-5vy9t8u35tuqm7vh67lrz9xp6 \
  192.168.99.100:2377

This node joined a swarm as a manager.
```

Manage nodes in a swarm

As part of the swarm management lifecycle, you may need to view or update a node as follows:

- list nodes in the swarm
- inspect an individual node
- update a node
- leave the swarm

List nodes in the swarm

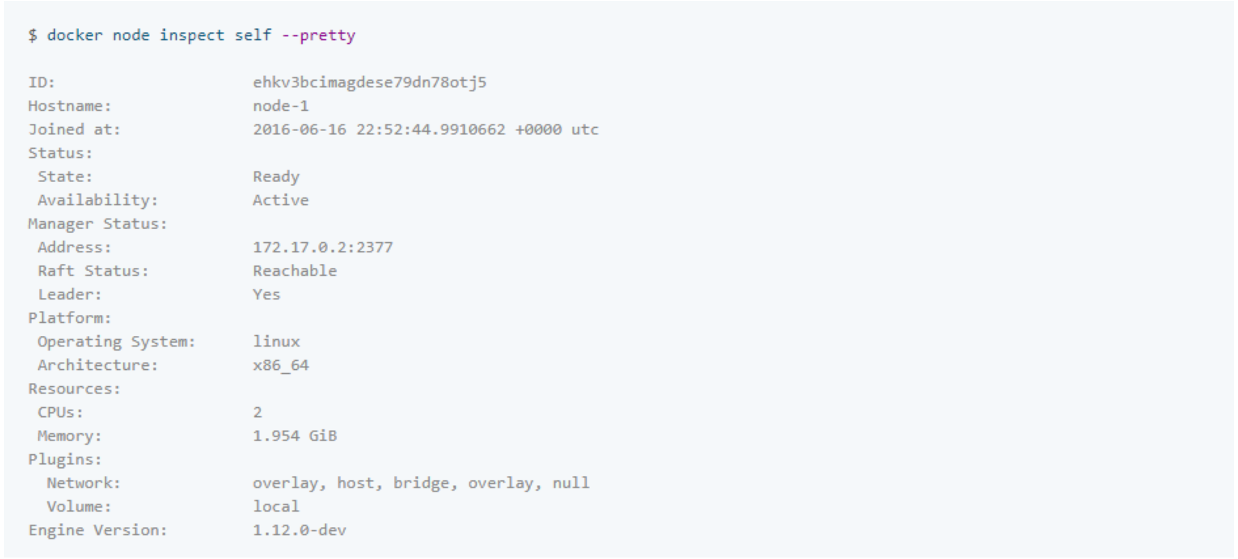
To view a list of nodes in the swarm run **docker node ls** from a manager node:

```
$ docker node ls

ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
46aqrk4e473hjbt745z53cr3t     node-5    Ready   Active        Reachable
61pi3d91s0w3b90ijw3deeb2q     node-4    Ready   Active        Reachable
a5b2m3oghd48m8eu391pefq5u     node-3    Ready   Active
e7p8btxeu3ioshyuj6lxiv6g0     node-2    Ready   Active
ehkv3bcimagdese79dn78otj5 *   node-1    Ready   Active        Leader
```

Inspect an individual node

You can run **docker node inspect <NODE-ID>** on a manager node to view the details for an individual node. The output defaults to JSON format, but you can pass the --pretty flag to print the results in human-readable format



```
$ docker node inspect self --pretty

ID:                       ehkv3bcimagdese79dn78otj5
Hostname:                 node-1
Joined at:                2016-06-16 22:52:44.9910662 +0000 utc
Status:
 State:                   Ready
 Availability:            Active
Manager Status:
 Address:                 172.17.0.2:2377
 Raft Status:             Reachable
 Leader:                  Yes
Platform:
 Operating System:        linux
 Architecture:            x86_64
Resources:
 CPUs:                    2
 Memory:                  1.954 GiB
Plugins:
  Network:                overlay, host, bridge, overlay, null
  Volume:                 local
Engine Version:           1.12.0-dev
```

Update a node

You can modify node attributes as follows:

- ❖ change node availability
- ❖ add or remove label metadata
- ❖ change a node role

Leave the swarm

Run the **docker swarm leave** command on a node to remove it from the swarm.

When a node leaves the swarm, the Docker Engine stops running in swarm mode. The orchestrator no longer schedules tasks to the node.

If the node is a manager node, you receive a warning about maintaining the quorum. To override the warning, pass the --force flag. If the last manager node leaves the swarm, the swarm becomes unavailable requiring you to take disaster recovery measures.

Deploy services to a swarm

To create a single-replica service with no extra configuration, you only need to supply the image name:

```
$ docker service create nginx
```

The service is scheduled on an available node. To confirm that the service was created and started successfully, use the docker service ls command:

```
$ docker service ls
```

To provide a name for your service, use the --name flag

```
$ docker service create --name my_web nginx
```

Update a service

You can change almost everything about an existing service using the docker service update command. When you update a service, Docker stops its containers and restarts them with the new configuration.

```
$ docker service update --publish-add 80 my_web
```

Remove a service

To remove a service, use the **docker service remove** command. You can remove a service by its ID or name, as shown in the output of the **docker service ls** command

```
$ docker service remove my_web
```

PRACTICE DOCKER NETWORKING

Q&A