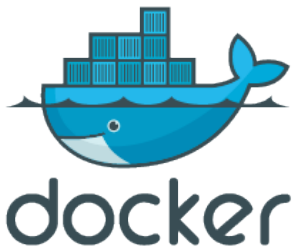
Getting started with Swarm mode & Create a swarm



This tutorial introduces Docker Swarm mode, which allows you to deploy and manage containerized applications across a cluster of Docker nodes. In Swarm mode, Docker Engine transforms multiple Docker hosts into a single, distributed system, making it easier to scale, orchestrate, and manage applications.

What This Tutorial Covers:

- Initializing a Docker Swarm
- Adding nodes to the Swarm
- Deploying services to the Swarm
- Managing the Swarm (Lightweight Version)

Prerequisites:

- You need **three Linux hosts** (physical or virtual machines) that can communicate over a network, with Docker installed.
- **Open ports** between the hosts.
- The **IP address** of the manager node.

Step 1: Setting Up the Environment

To get started, you'll need three Linux hosts that can communicate over a network. These hosts can be physical machines, virtual machines, or cloud instances (e.g., Amazon EC2).

- One of these hosts will be the **manager** (we'll call it `manager1`).
- The other two hosts will be **workers** (`worker1` and `worker2`).

You can follow most steps of this tutorial on a **single-node Swarm** (with just one host), but for full multi-node functionality, you'll need three hosts.

Install Docker Engine on Linux Hosts

Follow Docker's official installation instructions for your Linux distribution to install Docker on each of your machines. Once Docker is installed, you're ready to create your Swarm.

Check the Manager Node's IP Address

You'll need the IP address of the **manager node** (`manager1`) for the Swarm to function properly. To find it, run the following command on `manager1`:

```
ifconfig
```

This will display a list of available network interfaces. Pick the IP address that is accessible to the other nodes in the network. The tutorial assumes `manager1` has the IP address `192.168.99.100`.

Step 2: Open Required Ports

Ensure the following ports are open between all your nodes:

- **Port 2377 (TCP)**: For communication between manager nodes.
- **Port 7946 (TCP/UDP)**: For node discovery within the overlay network.
- **Port 4789 (UDP)**: For overlay network traffic (VXLAN).

If you plan to use an **encrypted overlay network**, ensure **IPSec ESP** traffic is allowed on **IP protocol 50**.

To secure your swarm further, consider applying the following **iptables** rule to block untrusted traffic from reaching the Swarm's data path port (4789):

```
iptables -I INPUT -m udp --dport 4789 -m policy --dir in --pol none -j DROP
```

Step 3: Initializing the Swarm

Once your setup is ready, it’s time to initialize the Swarm on your **manager node** (`manager1`).

1. SSH into the `manager1` machine.
2. Run the following command to initialize the Swarm, specifying the IP address of the manager node:

   ```
   docker swarm init --advertise-addr 192.168.99.100
   ```

> If successful, the output will look like this:

```
Swarm initialized: current node (dxn1zf6l61qsb1josjja83ngz) is now a manager.
```

It will also provide the command to add worker nodes to the swarm:

```
docker swarm join --token <worker-token> 192.168.99.100:2377
```

Step 4: Adding Worker Nodes to the Swarm

Now that the Swarm is initialized, you can add your **worker nodes** (`worker1` and `worker2`).

1. SSH into each worker node (`worker1` and `worker2`).
2. Run the `docker swarm join` command provided when you initialized the Swarm on `manager1`.

For example, on `worker1`, the command might look like this:

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

>

> Repeat this step for `worker2`. Once completed, both `worker1` and `worker2` will join the swarm and begin listening for tasks from the manager.

Step 5: Verifying the Swarm

To verify that all nodes have successfully joined the swarm, SSH into the **manager node** (`manager1`) and run the following command:

```
docker node ls
```

You should see all three nodes (`manager1` , `worker1` , and `worker2`) listed, along with their roles and statuses:

```
ID                            HOSTNAME   STATUS  AVAILABILITY  MANAGER STATUS
dxn1zf6l61qsb1josjja83ngz *   manager1   Ready   Active        Leader
8l3nse6qox9pxdj67c5utodl4     worker1    Ready   Active
fxp1kjvthh2qyuodhd83uixg5     worker2    Ready   Active
```

The `*` next to `manager1` indicates that you're currently connected to this node. The `MANAGER STATUS` column shows that `manager1` is the leader.

Step 6: Deploying a Service to the Swarm

Now that your swarm is ready, you can deploy a service to it. For example, you can deploy an Nginx web server service with three replicas:

1. SSH into the **manager node** (`manager1`).
2. Run the following command to deploy the service:

   ```
   docker service create --name web --replicas 3 -p 8080:80 nginx
   ```

   This command creates a service called `web` with three replicas, and each replica runs an Nginx container listening on port 80. The service is exposed to the outside world on port

8080.

Step 7: Managing the Swarm

After deploying a service, you can monitor and manage your swarm using several Docker commands.

Viewing Services

To see the list of services running in your swarm, use:

```
docker service ls
```

Viewing Nodes

To check the status of nodes in your swarm, use:

```
docker node ls
```

Scaling Services

If you want to scale the number of replicas for a service (e.g., increase Nginx replicas from 3 to 5), you can run:

```
docker service scale web=5
```

Removing Services

To remove a service, use the following command:

```
docker service rm web
```

Conclusion

Docker Swarm Mode simplifies the process of managing containerized applications across multiple machines. By setting up a swarm and deploying services, you can build a scalable, fault-tolerant infrastructure with minimal effort. This tutorial has covered the essential steps to get started with Docker Swarm, from initializing the swarm to managing services on it.

Revision #5
Created 29 August 2024 14:51:03 by aeoneros
Updated 11 September 2024 14:27:50 by aeoneros