Swarm Mode Introduction for IT Pros

Difference between Docker Compose and Docker Swarm Mode:

Compose is used to control multiple containers on a single system. Much like the Dockerfile we looked at to build an image, there is a text file that describes the application: which images to use, how many instances, the network connections, etc. But Compose only runs on a single system so while it is useful, we are going to skip Compose1 and go straight to Docker Swarm Mode.

Swarm Mode tells Docker that you will be running many Docker engines and you want to coordinate operations across all of them. Swarm mode combines the ability to not only define the application architecture, like Compose, but to define and maintain high availability levels, scaling, load balancing, and more. With all this functionality, Swarm mode is used more often in production environments than it's more simplistic cousin, Compose.

The application

Initialize Your Swarm

First thing we need to do is tell our Docker hosts we want to use Docker Swarm Mode.

Initializing Docker Swarm Mode is easy. In the first terminal window labeled [node1] enter the following:

```
docker swarm init —advertise-addr $(hostname -i)
```

```
[node1] (local) root@192.168.0.23 ~
$ docker swarm init --advertise-addr $(hostname -i)
Swarm initialized: current node (8tpf0r70axea09u2ivfu5zbe9) is now
a manager.

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

    docker swarm join --token SWMTKN-1-1af2bgknote8ng2obzqqek5e5rum
4in211ze03ub6kzm24jz9z-1gjmzcegkq3kzqqnkwgkbhh1k 192.168.0.23:2377

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

We are going to add a worker. Copy the "docker swarm join…" command from your manager's output and paste it in the 2nd terminal window on your screen.

```
[node2] (local) root@192.168.0.22 ~
$ docker swarm join --token SWMTKN-1-1af2bgknote8ng2obzqqek5e5rum4i
n211ze03ub6kzm24jz9z-1gjmzcegkq3kzqqnkwgkbhh1k 192.168.0.23:2377
This node joined a swarm as a worker.
[node2] (local) root@192.168.0.22 ~
```

Show Swarm Members

Use command in first window:

docker node ls

```
[node1] (local) root@192.168.0.23 ~
$ docker node ls
ID                            HOSTNAME            STATUS
   AVAILABILITY         MANAGER STATUS      ENGINE VERSION
8tpf0r70axea09u2ivfu5zbe9 *   node1               Ready
   Active               Leader              18.06.1-ce
mftch46ttkin580bqsozhg3hw     node2               Ready
   Active                                   18.06.1-ce
[node1] (local) root@192.168.0.23 ~
$
```

Manager node is leader

Clone the Voting App

Clone from github:

git clone https://github.com/docker/example-voting-app
cd example-voting-app

Deploy a Stack

A stack is a group of services that are deployed together: multiple containerized components of an application that run in separate instances. Each individual service can actually be made up of one or more containers, called tasks and then all the tasks & services together make up a stack.

As with Dockerfiles and the Compose files, the file that defines a stack is a plain text file that is easy to edit and track. In our exercise, there is a file called docker-stack.yml in the current folder which will be used to deploy the voting app as a stack. Enter the following to investigate the docker-stack.yml file:

cat docker-stack.yml

```
$ cat docker-stack.yml
version: "3"
services:

  redis:
    image: redis:alpine
    networks:
      - frontend
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure
  db:
    image: postgres:9.4
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
```

Ensure you are in the [node1] manager terminal and do the following:

docker stack deploy --compose-file=docker-stack.yml voting_stack

You can see if the stack deployed from the [node1] manager terminal

docker stack ls

```
$ docker stack ls
NAME                SERVICES            ORCHESTRATOR
voting_stack        6                   Swarm
```

We can get details on each service within the stack with the following:

```
docker stack services voting_stack
```

Let's list the tasks of the vote service:

```
docker service ps voting_stack_vote
```

From the NODE column, we can see one task is running on each node. This app happens to have a built-in SWARM VISUALIZER to show you how the app is setup and running. You can also access the front-end web UI of the app to cast your vote for dogs or cats, and track how the votes are going on the result page. Try opening the front-end several times so you can cast multiple votes. You should see that the "container ID" listed at the bottom of the voting page changes since we have two replicas running.

Scaling an application

How can we tell our app to add more replicas of our vote service? Type the following at the [node1] terminal:

```
docker service scale voting_stack_vote=5
```

Now enter your docker stack services voting_stack command again. You should see the number of replicas for the vote service increase to 5