Create A Swarm Cluster

setup a Docker Swarm cluster with the machines you have been provided. Docker Swarm is native clustering for Docker. It allows you create and access a pool of Docker hosts using the full suite of Docker tools. Because Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts.

Get A Swarm ID

Swarm uses a distributed key:value pair to cluster hosts together. The first step to creating a swarm on your network is to pull the Docker Swarm image. Then, using Docker, you configure the swarm manager and all the nodes to run Docker Swarm.

Make sure the docker client is connected to the `default` machine.

`docker run --rm swarm:1.0.1 create` will use v1.0.1 of Swarm to create a new unique ID. The `--rm` flag removes the container after it exits. If you don't pass the `--rm` flag you can see the container by running `docker ps -a`.

```

student001a@student001a:~$ docker run --rm swarm:1.0.1 create

Unable to find image 'swarm:1.0.1' locally

1.0.1: Pulling from library/swarm

2bc79aec8ea0: Pull complete

dc2fb86a875a: Pull complete

435e648d0f23: Pull complete

e16042a92d05: Pull complete

045bd7b00b5b: Pull complete

3caea1253d76: Pull complete

2b4c55187a27: Pull complete

6b40fe7724bd: Pull complete

Digest: sha256:989dd783c2a2e6decd3b60f52a8a99b81a2a7ff24c8b3fd7cb4b8bd699e61f6b

Status: Downloaded newer image for swarm:1.0.1

f29ab346337368c83f4087d21900b75d

```

Save that Token/Unique ID on the last line. (ie. `f29ab346337368c83f4087d21900b75d`).

This token is used by Swarm to register with the Docker Hub, which will operate as our service discovery service. It's possible to provide other Service Discovery services.

## Configure Swarm Agents

Swarm is used to cluster hosts together, so to test Swarm we are going to need more hosts.

Docker Machine makes it trivial to create more hosts in our Swarm:

```

docker-machine create \

-d virtualbox \

--swarm \

--swarm-discovery token://<TOKEN-FROM-ABOVE> \

swarm-agent-00

```

To see a list over all hosts handled by `docker-machine` run the `docker-machine ls` command.

To switch which machine the `docker` client/CLI talks to, follow the instructions when running:

```

$ docker-machine env MACHINE\_NAME

```

Create two agents (agent-00 and agent-01) and perform a `docker ps` against each of them (Switch between them using `docker-machine env`) to verify the Swarm container is running.

## Configure Swarm Master

Now that each of our hosts are acting as resources for the pool, we have to have a manager of these resources. This manager will become the docker endpoint. This means we will redirect our docker engine commands to point to the swarm master. When we issue commands to create a new container, the swarm master is responsible for looking at the pool of resources and deciding where to place the container.

Docker Machine takes away a lot of the hassle of creating a Swarm Manager:

```

docker-machine create \

-d virtualbox \

--swarm \

--swarm-master \

--swarm-discovery token://<TOKEN-FROM-BEFORE> \

swarm-master

```

If you connect directly to this machine using the instructions from `docker-machine env swarm-master`, `docker ps` will display something like this:

```

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

0ed11962dae4 swarm:latest "/swarm join --advert" About a minute ago Up About a minute 2375/tcp swarm-agent

ff356faa35ea swarm:latest "/swarm manage --tlsv" About a minute ago Up About a minute 2375/tcp, 0.0.0.0:3376->3376/tcp swarm-agent-master

```

We can see here that this machine is running both a swarm agent and a swarm master container. We can also see that the Swarm Manager is listening on port 3376. We can connect to the Swarm by following the instructions given by `docker-machine env --swarm swarm-master`.

Now, do `docker ps`.

Run `docker info` and you will see something different than before

```

$ docker info

Containers: 2

Images: 2

Role: primary

Strategy: spread

Filters: health, port, dependency, affinity, constraint

Nodes: 2

swarm-agent-00: 192.168.99.109:2376

Status: Healthy

Containers: 1

Reserved CPUs: 0 / 1

Reserved Memory: 0 B / 1.021 GiB

Labels: executiondriver=native-0.2, kernelversion=4.1.13-boot2docker, operatingsystem=Boot2Docker 1.9.1 (TCL 6.4.1); master : cef800b - Fri Nov 20 19:33:59 UTC 2015, provider=virtualbox, storagedriver=aufs

swarm-agent-01: 192.168.99.110:2376

Status: Healthy

Containers: 1

Reserved CPUs: 0 / 1

Reserved Memory: 0 B / 1.021 GiB

Labels: executiondriver=native-0.2, kernelversion=4.1.13-boot2docker, operatingsystem=Boot2Docker 1.9.1 (TCL 6.4.1); master : cef800b - Fri Nov 20 19:33:59 UTC 2015, provider=virtualbox, storagedriver=aufs

CPUs: 2

Total Memory: 2.043 GiB

Name: b187c3b2664f

```

## Using Docker with Swarm

Now that you have a two container hosts combined with Swarm, use the `docker run` command against the swarm master (`docker-machine env --swarm swarm-master`).

Run `docker run -tid busybox` and then to a `docker ps`. If you notice, the name of the container has changed and now has a prefix. This prefix indicates what machine it's running on. The following shows the new container running from `swarm-agent-01`:

```

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

61d851072c24 busybox "sh" 3 seconds ago Up 2 seconds swarm-agent-01/romantic\_galileo

```

Run `docker run -tid busybox` a few more times. You will see that the amount of containers on each host is spread evenly. This is because we are using the `spread` strategy for Swarm. The other options would have been `binpack` and `random`.

The `spread` and `binpack` strategies compute rank according to a node’s available CPU, its RAM, and the number of containers it is running. The `random` strategy uses no computation. It selects a node at random and is primarily intended for debugging.