Lab 12.2: Docker Swarm
IN720 Virtualisation

Introduction

Docker Swarm is a system that lets us manage containers across a collection of Docker hosts. It's already included in more recent versions of Docker, so there is nothing extra to install. In one sense, it's just an additional set of `docker` commands to run.

For this lab you will get a new set of three virtual machines to use. Go ahead and open ssh settings on each of them. You will find that one is named `master` and the other two are `worker0` and `worker1`. There's nothing different in how the three machines are set up at this point, it's just a matter of how we plan to use them.

1 Set up our swarm

We begin by initialising our swarm on what will become our swarm management host. The decision on which of your three system is to be used as the manager is left as an exercise. Perform the initialisation with the command

```
docker swarm init
```

This will create our new swarm and add the current system to it as a manager. In the output for this command you will see the command you need to run on other systems to add them to the swarm as workers. In particular, you will see the value of the *join token* needed to join the swarm. Don't worry if you lose this, however, because you can always retrieve it with the command

```
docker swarm join-token worker
```

or

```
docker swarm join-token manager
```

if you want to add managers to your swarm.

From this point on, any commands we use will be entered on the manager unless noted otherwise.

Now add your two remaining machines to your swarm as workers. You can see the systems in your swarm with the command

```
docker node ls
```

Be sure that all tree systems have been added to your swarm and that one of them is the manager.

2 Create and manage a service

A swarm *service* is a set of containers built from the same image that you have directed to be run on your swarm. Start your first service with the command

```
docker service create --name hi --replicas 2 tclark/hello
```

entered on your manager host. This command creates a service named "hi" that runs two containers built from the `tclark/hello` image. The two containers will run somewhere on your swarm, probably but not necessarily on two different nodes. Check the status of your service with the command

```
docker service ls
```

and then see where your containers are running with the command

`docker service ps hi` (since our service is named `hi`).

On one of the systems where a container is running, use `docker ps` to find the id of the container and then stop it with the command

```
docker stop <container-id>
```

Now go back to the manager node and run `docker service ps hi`. You should observe two things:

1. The listing will show that one of your containers was stopped.
2. A new container has been started since you specified that you want two instances running.

We can actually change the number of containers running in our service with a command like this:

```
docker service scale hi=4
```

that specifies that we now want four containers running on our swarm. Enter the command and then use `docker service ps hi` to see that they are running on your swarm.

Finally, you can stop your service with the command

```
docker service rm hi
```

You can see the relevant docker commands at `https://docs.docker.com/engine/reference/commandline/swarm/` and `https://docs.docker.com/engine/reference/commandline/service/`.

3 Set up a service with a compose file

Now that you can work with a swarm, take the compose file you wrote in lab 12.1 and adapt it for use on your swarm. In particular:

1. Since you can't use `build` in a swarm setting, remove that.
2. Any images you need to build must be pushed up to Docker Hub.
3. Specify that you will use an `overlay` network.