Lab 10.2: Using Docker Images
IN720 Virtualisation

Introduction

In our initial Docker trial we worked with some base Ubuntu images. In this lab we will see how we can download and launch images from a registry that encapsulate complete services.

Continue using the virtual server you used in the previous labs to complete this lab.

1 Pulling Docker images

Last time we used `docker run` commands to download, if necessary, and then run Docker images. Once we have downloaded a particular image onto a system we can launch new containers based on that image quickly and easily. Use the command `sudo docker images` to see the list of images on your system right now. You should see an image from the `ubuntu` repository with the tag `latest`. When we downloaded that image in the previous lab, we didn't ask for a particular tag, so we got the `latest` tagged image by default.

Let's download a Debian Jessie image to add to our collection. Generally, when you download an image you want to be able to select a particular version, and these are identified by *tags*. We can specify the desired tag when we get an image. Do this now with the command

```
sudo docker pull debian:jessie
```

We use `pull` to download the image without starting a container that uses it yet. We asked for the image with from the `debian` repository with the `jessie` tag. While we're at it, pull another image with

```
sudo docker pull debian:8.1
```

and now list our images with the command

```
sudo docker images
```

We see two `debian` images, one for each tag, but they both have the same image ID. This is because the same image can be identified with multiple tags.

2 Finding and using more interesting images

So far we have just used base images. The real power of Docker comes from using images that encapsulate the resources to run some service. We'll see how to create such images soon, but first let's use some that are easily available. Our Docker systems are configured to work with the *Docker Hub*, a shared online registry of images. We can find and download prebuilt images from the Docker Hub easily. Let's look for images that run Wordpress. We can find such images at the command line with the command

```
sudo docker search wordpress
```

which returns several matches. Notice that the first one is simply called `wordpress`. A repository whose name is a bare word like that is curated by the Docker team. Other repositories are named according to the `username/repo-name` scheme and have been uploaded by Docker Hub users. Anybody can create and share images on Docker Hub, so we should exercise appropriate care when using them.

Note that you can do the same search on the web by going to `https://hub.docker.com` and searching there. Search for "wordpress" on the web site now. We are interested in the `tlongren/docker-wordpress-nginx-ssh` images, so select that repository from the search results. We want to pull and run a container from this image, which we will do with the command

```
sudo docker run -p 80:80 -p 2222:22 --name wordpress -d tlongren/docker-wordpress-nginx-ssh:stable
```

The `-p` option map ports between the container and the host system. Once your container is running, you should be able to browse to your virtual machine with a web browser and see the initial Wordpress setup screen. You can also ssh into your container by ssh'ing to your host machine's ip address at port 2222. For ssh, the user name and password are both `wordpress`.

As an aside, we've been saying that a Docker container's purpose is to run one process, but containers made from this Wordpress image clearly are running more that one. How is this the case?