Lab 10.2: Dockerfiles
IN720 Virtualisation

Introduction

We will see that creating Docker images is basically a type of coding. We assemble a directory of code and other resources, called a *build context* in Docker parlance, and then we build our images from that code.

You will need a GitHub account to do this lab.

1 Create a repository

The easiest way to get started is to create an empty repository on the GitHub web site and then clone it on your local machine.

1. Log onto the GitHub site and start creating your new repository by clicking on the little "+" menu at the top right and selecting "New repository".
2. Name your repository "docker-lab10.2".
3. Tick the box to initialise your repository with a README and add an appropriate license from the menu. Don't bother adding a `.gitignore` right now. There is no standard Docker `.gitignore` template as yet.
4. Click the "create repository" button.

2 Cloning your (empty) build context

SSH into your Ubuntu VM. You will probably need to install git with the command

```
sudo apt-get install git
```

before proceeding. You will need to generate an ssh key and register it with your GitHub account. Instructions to do this are at `https://help.github.com/articles/generating-ssh-keys/` if you do not know how to do this.

Once this is done, you can clone your repository with the command

```
git clone git@github.com:<your-username>/docker-lab10.2.git
```

When this completes you will have a local directory named `docker-lab10.2`. We call this directory your *build context* and we will place the resources used to build your container inside it.

3 Prepare your build context

Inside your new build context directory create a new file called `Dockerfile`. We will edit this file, adding a series of directives, one per line, that build up our container in a step-by-step manner.

Add the following lines to your Dockerfile:

```
FROM ubuntu:16.04
```

Dockerfiles always start with a `FROM` statement that specifies the base image type from which this container is built.

```
LABEL maintainer="<your name/email address>"
```

This identifies the person responsible for this image.

```
RUN apt-get -q update && apt-get -yq dist-upgrade
RUN apt-get -yq install apache2
```

`RUN` commands run their arguments as if they were entered into a shell inside the container.

```
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
ENV APACHE_LOCK_DIR /var/run/apache
ENV APACHE_PID_FILE /var/run/apache/httpd.pid
RUN mkdir /var/run/apache
```

`ENV` commands populate environment variables inside the container with values. This is a common way to pass configuration information to services running inside them.

```
ADD index.html /var/www/html/index.html
```

This command will take a file named `index.html` from your build context and place a copy at the location specified inside the container. If the first argument to `ADD` is a directory, indicated by a trailing slash, then it will add the directory and its contents.

```
EXPOSE 80
```

This will cause our container to open port 80 on the host system. Note that this doesn't mean that the host's port 80, or any other port, will be open or associated with the container's port. This is done with the `docker run` comand.

```
ENTRYPOINT ["/usr/sbin/apache2"]
CMD["-DFOREGROUND"]
```

Recall that Docker containers typically run one main process. The `ENTRYPOINT` specifies the command to run inside the container to start that process. The following `CMD` provides arguments to the `ENTRYPOINT`'s command.

This complete's our Dockerfile. Since the `ADD` command specified an HTML file, you'll need to place one inside your build context alongside the Dockerfile. This can just be a simple hello world page.

Don't forget to commit and push your changes.

4 Build your image

From within your build context directory, build your new image with the command

```
docker build -t=‘‘your-username/lab10.2’’ .
```

Once this completes successfully, you can run a container based on this image with the `docker run` command and see your web page if you visit your server with a browser. If you make any changes to your build context you will need to rebuild the image, but this will not affect any containers built from older versions of the image.

There are many more dockerfile directives beyond the ones we used here. You can consult the text to learn more, and you can check online documentation at `https://docs.docker.com/engine/reference/builder/`. Note that the web site will have somewhat more up-to-date information.