Automated Docker Builds
IN720 Virtualisation

Introduction

We've seen 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. Once we realise this it becomes clear that

- We should manage this code with revision control systems;
- We should use systems to automatically build (and test) our images.

We've parenthesised "and test" above because, even though it's a good idea, we're not going to solve this problem today. We will, however, commit our code to GitHub and trigger automatic builds from our repositories.

You will need accounts on GitHub and complete today's lab, so if you don't already have one[1]

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-lab3.1".
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 button to create your repository.

2 Cloning your (empty) build context

SSH into your Ubuntu VM, or get a new one from the lecturer if you need one. You will probably need to install git with teh 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.

[1] WTF? How did you get to this point in life without having a GitHub account? I thought we were all adults here.

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

`git clone git@github.com:<your-username>/docker-lab3.1.git`

When this completes you will have a local directory named `docker-lab3.1`.

3 Prepare your image

Inside your new build context directory create a new `Dockerfile` that does the following:

- Builds from `[ubuntu:14.04]`;
- Applies standard updates;
- Installs `apache2`, `openssh-server`, and `supervisor`;
- Adds the file `supervisord.conf` to `/etc/supervisor/conf.d/supervisord.conf`;
- Exposes ports 22 and 80;
- Runs the command `/usr/bin/supervisord`.

Now you need to add a copy of the file `supervisord.conf` to your build context. Use the following:

```
content...[supervisord]
nodaemon=true

[program:sshd]
command=/usr/sbin/sshd -D

[program:apache2]
command=/bin/bash -c "source /etc/apache2/envvars && exec /usr/sbin/apache2 -DFOREGROUND"
```

Once this is done, build and test your container locally.

4 Setting up an automated build

From your Docker Hub home page, click the "Add Repository" button and choose the "Automated Build" option. You will be walked through the steps to link this repository to your new GitHub repository. Once this is done, and new commits to your GitHub repository will trigger an automatic build of your `master` image.

Test this by make a small change to your image build context and then commiting and pushing your change to GitHub.