Lab 11.1: DockerHub
IN720 Virtualisation

Introduction

Now that we can create images, we need to be able to distribute them using a Docker *repository*. DockerHub is the standard repository for Docker images and we will use it in this lab.

You will need an account on DockerHub to do this lab, and you must complete lab 10.2 before doing this lab.

1 Pushing an image

Since our Docker servers are configured to use DockerHub by default, so there's almost nothing to do in order to push it upstream. Simply issue the command

```
docker push yourusername/lab3.1
```

Since this is the first time you have pushed to DockerHub on this system, a dialogue will verify your credentials before performing the push.

Check the DockerHub web site to verify that your image shows up on your profile. Then, find the name of one of your classmates' images, pull it, and run a container from it.

2 Automated builds

Although we can now distribute our images on DockerHub, the situation is less than ideal. Anytime we make a change to the image we must perform a set of manual steps to rebuild and push the image. Also, even though we are storing the build context on GitHub, there's nothing explicitly linking the image to those sources. Effectively, we can't be completely sure what is in those images. We can address all of those by setting up an *automated build*.

On DockerHub, go to your repository's page. Select the "Build" item from the top menu and then click the GitHub link to connect your DockerHub repository to the associated GitHub repository that holds your container image's build context.

From now on, any time you push a new commit to your GitHub repository it will trigger a new build of your image on DockerHub. Try is now by making a small change to your build context and committing it.

Explore the settings for this new repository. You'll see that the Dockerfile is shown here so that we can quickly see what your image contains, although we have to go to the associated GitHub repository to see the other resources. In particular, note the build settings. Here you can set this repository to build and tag images whenever you push an associated branch or tag to GitHub.

3 Conclusions

A repository like DockerHub is basically necessary to distribute and use your Docker images. The ability to set up automated builds also makes it easy to maintain and update images in an organised manner.