QA Consulting

Docker Images

TRAINING MATERIALS - MODULE HANDOUT

Contacts

robert.crutchley@qa.com
team.qac.all.trainers@qa.com

www.consulting.qa.com

Contents

Overview

A Docker image is a read only file that is effectively a snapshot of a container. For instance if you could take a snapshot of the current state of the computer you are using, move it to any another computer and run it so that its in exactly the same state as when you took the snapshot, then this is a similar concept to how Docker images work.

Image Layers

Overview

Docker images are composed of layers. Because images are read only meaning that you can't change them, only create new ones, image layers become a very useful feature for building new images.

Multiple steps are often taken to get an image configured to a state we want it before installing an application on it, wouldn't it be nice if all we had to do is install the application on an already configured image? This is what the layers allow us to do.

Practical Benefits

The benefit of image layers is you can build off other Docker images. Let's say that we wanted to run a Java application, we could take an existing image that has a Java environment already configured and all we have to do is copy our application onto the image and run it.

Each layer is a file that can be downloaded individually and reused for other images, this means that the same Java "base image" can be used for multiple applications which is very efficient.

Image Properties

Overview

Before managing Docker images it is useful to understand what different attributes they have so that they can be properly utilised when managing images. The properties discussed below can be found when you run the **docker images** command.

ID

The identifier is unique for every image thus making it the a reliable property when referencing the image

Repository

The location and name of the image, which follows the format of: **<host>/<author>/<application>**.

- Host
  This is the remote registry that the image is stored in. If it is left empty then the local registry will first be checked, if the image is not there then the default remote registry (**docker.io**) will be consulted for the image.
- Author
  Whoever created the image, for example if you are trying to upload images to Docker Hub (**docker.io**) then this value will be your Docker Hub username.
- Application
  The name of the application, service or solution that has been built and configured inside of the docker image.

Tag

Tagging images is a way of differentiating between them, more commonly known as versioning. If no tag is provided when referencing an image then the default **latest** tag is used.

Managing Images

Searching

One of the most useful parts about Docker is the contributions from the community and developers. For most application services and base images for popular languages, there are already images that exist for them.

docker search <image>:<tag>

```
docker search java:8
```

Downloading

Before an image can be used it must be present in the local registry, the **docker pull** command can accomplish this. Note that the **docker run** command downloads the image as well as running it if it does not exist locally.

docker pull <image>:<tag>

```
docker pull java:8
```

Uploading

Docker images that you have created can also be uploaded using the docker push command, just provide the image name.

docker push <image>:<tag>

```
docker push my/java:8
```

Deleting

An important part of managing Docker images is deleting them, although Docker is efficient at storing them, depending on what images you are building they can take up quite a lot of space.

docker rmi <image>:<tag>

```
docker rmi java:8
```

Renaming

The name of the image effects where it will be uploaded and how it can be referenced, there are many reasons you might want to do this, the **docker tag** command effectively “renames” the image.

docker tag <old-image>:<old-tag> <new-image>:<new-tag>

```
docker tag java:8 bobcrutchley/java:8
```

Tasks

1. Find the latest, official Java image
2. Download the latest Java image
3. Delete the Java Docker image that you downloaded
4. Download the Java 8 Docker image
5. Rename the Java 8 Docker image so that it is prefixed with your name, for example: **bob/java:8**
6. Login to your Docker Hub account using the **login** command
7. Push the java image to your Docker Hub
8. Delete the local images
9. Download the image from the Docker Hub