Creating Docker Containers

Mark Dunning

Last modified: 08 Nov 2016

The creation of Docker images is specified by a *Dockerfile*. This is a text file containing the sequence of instructions required to re-create your image from some starting point, which could be the standard Ubuntu image. Essentially we list the commands we use, step-by-step to install all the software required. If you already have a shell script to install your software, then translating to a Dockerfile is relatively painless.

In this section we show how to create a Dockerfile and use this file to build a docker image. A useful reference is the official Docker documentation on Dockerfiles (https://docs.docker.com/engine/reference/builder/), which goes into far more detail than we will here.

In this example we show the ***Dockerfile*** used to create a Ubuntu image with the `build-essential` and `wget` tools installed.

```
FROM ubuntu
MAINTAINER YOU NAME<your.name@cruk.cam.ac.uk>
RUN apt-get update
RUN apt-get install -y wget build-essential
```

The `FROM` instruction is mandatory as it defines the starting point for your image. It is good practice to have a contact address too.

The remaining lines of the file are the command line steps you would run in order to create the image.

The `docker build` command is then used to build an image that we can use. The argument `-t` in this case specifies a name for the image (tag?) and traditionally the `Dockerfile` is located in the current directory. The current directory will also be used to define the *context*, meaning you can include files or directories on your local machine in the image. Note that we do not need a `sudo` as the image is built with super-user privileges.

```
docker build -t="docker-test/ubuntu-build-essential" .
```

To create a `Dockerfile` for our samtools image, we can firstly take advantage of the `build-essential` image that we just created to ensure we have `wget` and the `build-essential` tools available.

```
FROM docker-test/ubuntu-build-essential
MAINTAINER YOU NAME<your.name@cruk.cam.ac.uk>
RUN apt-get update
RUN apt-get install -y ncurses-dev zlib1g-dev
RUN wget https://github.com/samtools/samtools/releases/download/1.3.1/samtools-1.3.1.
tar.bz2
RUN mv samtools-1.3.1.tar.bz2 /opt
WORKDIR /opt
RUN tar -jxf samtools-1.3.1.tar.bz2
WORKDIR samtools-1.3.1
RUN ./configure
RUN make
RUN make install
```

The remaining steps translate the steps we used in the previous section to the `Dockerfile` syntax. The only tricky bit is to change directory you need the `WORKDIR` command rather than a unix `cd` command.

```
docker build -t="docker-test/samtools" -f Dockerfile.samtools .
```

We already mentioned that 'dockerhub' is a repository where people can distribute their docker containers. You can easily sign-up for a dockerhub account if you already have github. Then any container can be pushed with the `docker push` command. Here, I tag my image with the prefix `markdunning` (which is my github and dockerhub account)

```
docker build -t="markdunning/ubuntu-build-essential" .
docker push markdunning/ubuntu-build-essential
```

The container is then available on dockerhub (https://hub.docker.com/r/markdunning/ubuntu-build-essential/)

N.B. you'll notice the image builds much quicker because it has already been built once. To stop this behaviour you can specify the `--no-cache` option

In fact, you can build images from a github repository with Automated Builds (https://docs.docker.com/docker-hub/builds/)

There are more commands available in a Dockerfile, as described in the reference guide (https://docs.docker.com/engine/reference/builder/). Some particular ones I have used, or encountered…

- `USER` to change the user
- `ENV` to create an environment variable
  - e.g. to link to a particular java file

```
ENV PICARD /opt/picard-tools....
```

- `ENTRYPOINT` change what command is automatically run
  - e.g. the sanger cgpbox is defined to run a particular script

```
ENTRYPOINT $OPT/bin/runCgp.sh
```

- `EXPOSE` make a particular port number available