R Docker exercises for useR!2018

Symbolix (www.symbolix.com.au)

Inside the Repository

Dockerfiles are used to build up an image. We start **FROM** a base image. Then we **COPY** files or **RUN** extra commands or set specific **ENV** variables. The Dockerfile lives in the top of the project and should be called **Dockerfile** with a capital **D**.

In this example, we are starting from the rocker/studio image. These are public (not official) but they are solid and very well supported. Rocker also have images for r-base (rocker/r-base) and a geospatial suite (rocker/rstudio-geospatial). This has all the basic spatial libraries (sp, sf) installed plus all the stuff you require outside of R to make them work (e.g. GDAL).

To install extra libraries we specify them in `requirements.R`. On build, this is copied onto the instance and run to install the libraries.

Finally the build copies our files over - the `Analysis` folder and the `Data` folder. We put these in the home directory of our user, called `rstudio`.

Build it

Type the following command into the command line. You must be in the same directory as your Dockerfile.

```
sudo docker build --rm --force-rm -t rstudio/hello-world .
```

the `--rm --force-rm` just forces the container to delete itself once its scripts run or you log out. It just stops us filling up the server with lots of containers doing nothing. Once this has built run

```
sudo docker image list
```

to see your image added to the list. We've called it `rstudio/hello-world` but you can call it anything.

Run it

We want to use this image to access rstudio so we want it running as a background service (i.e. in detached mode) we use the flag `-d` to do this. If you want to access a bash shell or other interactive mode, you need to specify `-it`.

Rstudio runs on port 8787 within the container. We need to map this to an unused port on the host machine with a `-p <host port>:<container port>` We will use 28787, but this can be any used port.

We will call our container `hello-world`. This is the simple run command:

```
sudo docker run -d --rm -p 28787:8787 --name hello-world rstudio/hello-world
```

Run this command and access the container through your webbrowser at `<yourhostip:28787>`. Username and password are both `rstudio`.

In rstudio, type

```
source("Analysis/hello.world.R")
```

You will see that you can see the Analysis and Data folder but there are two problems.

1. In order to write to a file within Docker (through rstudio) you need to have the right userid. With these rocker

images you can get that by specifying -e USERID=$UID in the run command. Then you can write and you can make changes to files and save them within the container.

2. It's all well and good to write to the local container but this data won't be permanent. We can write our output back to the host directory by mounting a host directory as a volume on the container with -v /full/path/to/dir . This is also useful in development as you can make changes in your permanent host folder which are then immediately available on the container without rebuilding it.

Before we fix the problem we need to stop the container that's running (it's no good for us):

```
sudo docker stop hello-world
```

Now lets try again. If you look in run_docker.sh you will see a better version and explanation. Basically its:

```
DATA_DIR=${PWD}/Data
sudo docker run -d --rm -p 28787:8787 --name hello-world2 -e USERID=$UID -e PASSWORD=SoSec
ret! -v $DATA_DIR:/home/rstudio/Data rstudio/hello-world
```

Note in the above I have also set the password manually – you can make it anything you want.

Run the commands above, log into `<yourhostip:28787>` and try sourcing the script. It should run and write to the Data folder. `</yourhostip:28787>`

Finally, go back to the command line once more. Type `ls Data` and you should see the output file there also.

One more thing. In the rstudio window, open up `Analysis/hello.world.R` add a line to the bottom - any command you want and save it and run it.

Final question - if I check the contents of `Analysis/hello.world.R` on the command line (i.e. back on the host) will it have your new line? Why?

Have a play

- Pass environment variables (-e) to
  - give you permissions to save changes to `hello-world.R`
  - change `username` and `password` to something more secure
- Save your data
  - Save the output data back to your laptop by mounting directory (-V)
- Set yourself up so you can make changes in hello.world.R on your local directory and immediately have them show up and work in the container Rstudio. When might this be useful?

Other things to try:

- Can you set the `username` and `password` in the Dockerfile instead of the run command?
- `docker stop` your container and use `docker ps` to confirm it's gone.
- Run two versions of the container at the same time. What runtime settings need to change?
- Change the `requirements.R` (add/remove a library) and rebuild. Does it rebuild all the layers? Why?

Open docker play/questions time:

Feel free to:

- Continue tinkering with our R-docker repo and questions above
- Extend our example to set up a container suited to their particular interests/favourite libraries (choose good rocker image, add requirements, build….)
- Extend our example into a set up to produce three related datasets at the same time, with different inputs. How will you set up your output folders? How will you configure the required differences (e.g. R environment variable, log into each one separately??)