Module 12 Pre-Read - Docker Part 2

Wednesday, June 5, 2024 05:55

Pushing an image to DockerHub

Make sure the image has a correct tag (word that describes the image)

`docker build -t decathinator/image_name:latest`

Format is docker hub username, slash, the image name, and then optional description word

`docker push decathinator/image_name`

Push to dockerhub

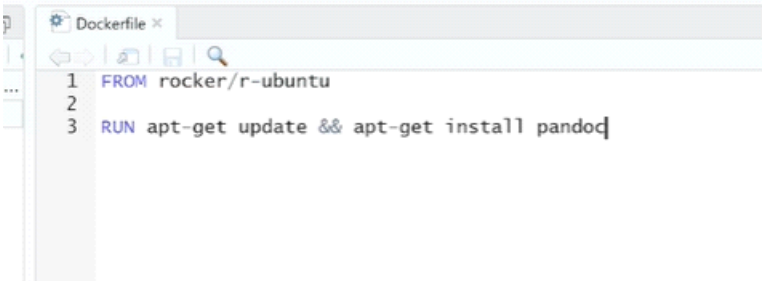
R in Docker, the easy way

There is a group called Rocker on the DockerHub website that have very useful images

`docker pull rocker/r-ubuntu`

In console. Now we have installed the rocker/r-ubuntu image

Example of docker file after we pulled that shows how we can build on the image:



RStudio IDE in Docker

`docker pull rocker/rstudio`

Pull the docker image

`docker run -e PASSWORD="secret123" -p 8787:8787 rocker/rstudio`

`-e` sets the environment variable password

`-p` is port option

(optional) -d means detached mode, means that container continues running in the background

Navigate to http://localhost:8787

Username is `rstudio`

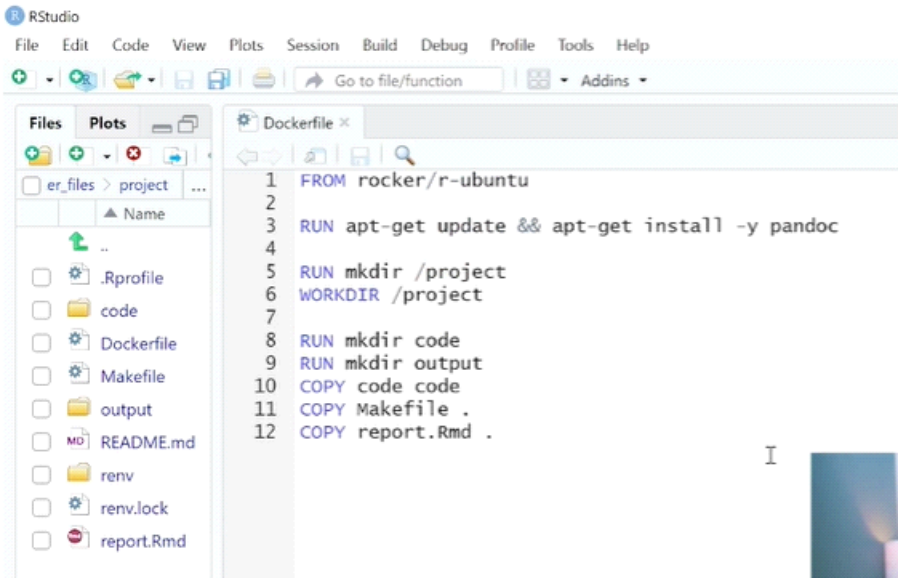
Password is `secret123`

To shut down the container, you can just close the terminal window or type `docker kill mycontainerid` (you can find the container id if you do `docker container ls`)

Building a fully automated workflow with R and docker

Navigate to project directory and make a dockerfile via `touch dockerfile`

Example of a dockerfile:



1. Navigate to project directory
2. `docker build -t project_image .`
3. `docker run -it project_image bash`

   Check contents to see if contents were copied correctly

   Now you can run bash and check via `ls`

renv and Docker

Need to manually copy over the renv files except for all the stuff in renv library

In the dockerfile:

```
COPY .Rprofile .
COPY renv.lock .
RUN mkdir renv
COPY renv/activate.R renv
COPY renv/settings.dcf renv

RUN Rscript -e "renv::restore(prompt = FALSE)
```

Mounting directories

I want the directory on container to sync with directory on local computer

`docker run -it -v <path to local directory>:<path to image directory>`

The `-v` option mounts the directory

On a windows machine, I need to do `docker run -it -v "//c/<path to local directory>":<path to image directory>`

After you build the image, you should be able to view whatever file

Using command substitution to shorten docker run command

Instead of writing out the whole file path, you can use a command substitution

`docker run -it -v $(pwd):<path to image directory> project_image bash`

For windows, need to do `docker run -it -v "/$(pwd)"/final_report:<path to image directory/final_report> project_image bash`

Finalizing the automated build

Need to create an entry point to the container

In the dockerfile:

```
CMD make && mv report.html final_report
```

Now the report should be made automatically

A Make rule for docker builds

In the makefile:

```
# DOCKER-ASSOCIATED RULES
PROJECTFILES = report.Rmd code/01_make_output.R code/02_render_report.R Makefile
RENVFILES = renv.lock renv/activate.R renv/settings.dcf

# rule to build image
project_image: Dockerfile $(PROJECTFILES) $(RENVFILES)
  docker build -t project_image .
  touch $@
```

`$@` evaluates to the target name in make

This way if any stuff changes then the image will rebuild

A Make rule for docker run

In the makefile:

```
# rule to build the report automatically in our container
final_report/report.html: project_image
  docker run -v "/$$(pwd)/final_report":/project/final_report project_image
```

Make requires two dollar signs not just one

An alternative to renv

Do manual package installation inside the dockerfile

```
Dockerfile
FROM rocker/r-ubuntu

RUN Rscript -e "install.packages('here')"
RUN Rscript -e "install.packages('rmarkdown')"
```

Contrasting renv and install.packages for use with Docker

renv

PRO	CON
• renv.lock gives record of package versions • Package versions are same across multiple builds • Easy to synchronize docker images with local computer	• Docker builds don't cache efficiently • Careful Dockerfile coding required

install.packages()

PRO	CON
• Docker builds efficiently use cache • No need to understand renv intricacies	• Package versions may change across builds • (Slightly) harder to list of package versions • Harder to synchronize versions across local computer and Docker image

Create an ad-hoc lock file

```
Terminal 1 | MINGW64:/c/Users/David Benkeser/Documents/docker_files/project
$ docker run project_image Rscript -e "installed.packages()[,'Version']"
 base64enc      bslib     cachem     digest   evaluate    fastmap         fs
   "0.1-3"    "0.4.1"    "1.0.6"   "0.6.30"     "0.18"    "1.1.0"    "1.5.2"
      glue       here      highr  htmltools  jquerylib   jsonlite      knitr
   "1.6.2"    "1.0.1"      "0.9"    "0.5.3"    "0.1.4"    "1.8.3"     "1.40"
```

```
$ docker run project_image Rscript -e "installed.packages()[,'Version']" > bootleg.lock
```

So this is when you aren't using renv, and showing that we can still retrieve this information

<u>A containerized development workflow using the RStudio IDE</u>

Mount everything and edit and work interactively etc

```
Console   Terminal ×   Background Jobs ×
Terminal 1 | MINGW64:/c/Users/David Benkeser/Documents/docker_files/project
$ docker run -e PASSWORD="secret123" -p 8787:8787 -v "/$(pwd)":/home/rstudio rocker/rstudio
```

And once you save the file in the container, it will also update locally