Docker Helper

Zero to (container) hero

Dani Arribas-Bel

Table of contents

License **2**

Introduction **2**

The anatomy of Docker: images and containers **2**
- Images . . . 2
- Containers . . . 3

Running your first container **3**

Options when running a container **3**

Using `docker-compose` **4**

Popular images for (geographic) data science **5**
- `gds_env` . . . 5
- Docker . . . 5
- Compose . . . 6
- Rocker . . . 6
- Docker . . . 6
- Compose . . . 7
- `gdsrpy` . . . 7
- Docker . . . 8
- Compose . . . 8

Building your own images **9**

References **10**

License

Introduction

This document presents a sequence of steps to walk through in a live session to get your head around Docker for (geographic) data science. On its own, it is probably of limited utility.

Warning

The reader is expected to be familiar with the following two sections of Chapter 2 in Rey, Arribas-Bel, and Wolf (2023) [1]:

- Reproducible platforms
- Containerised platform

The anatomy of Docker: images and containers

Images

```
docker image ls
```

```
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
darribas/gds_dev    10.0      e723f55128e3   3 months ago     12.2GB
```

[1] Note the book is also available online for free at: geographicdata.science/book

Containers

```
docker ps
```

Running your first container

```
docker run darribas/gds_dev:10.0
```

Ensure you have the `darribas/gds_dev:10.0` downloaded locally (otherwise, make sure you have a good internet connection and prepare to wait for a bit while id downloads.

Which is the same as:

```
docker run \
    darribas/gds_dev:10.0
```

In this context, \ simply "breaks" the code line, ensuring the next one is read as part of the same code snippet. We will use this pattern a lot below for legibility.

Options when running a container

Let's make the command above useful. The following ensures the server stays running and, once exited, the container is discarded.

```
docker run \
    --rm \  # Destroy when exited #<<
    -ti \   # Run on terminal and interactive #<<
    darribas/gds_dev:10.0  # Docker image
```

Now we also forward the port from the container to the host, so if we visit `localhost:8888`, we can access the server.

```
docker run \
    --rm \  # Destroy when exited
    -ti \   # Run on terminal and interactive
    -p 8888:8888 \  # Port #<<
    darribas/gds_dev:10.0  # Docker image
```

This is great, a usable Jupyter instance. But it is fully isolated from the data in the host. To "mount" the host in the file system of our container, and access its files, we add the following line:

```
docker run \
    --rm \  # Destroy when exited
    -ti \   # Run on terminal and interactive
    -p 8888:8888 \  # Port
    -v ${PWD}:/home/jovyan/work \ #<<
    darribas/gds_dev:10.0  # Docker image
```

Using `docker-compose`

The above is great but requires a great amount of memory muscle to remember all those settings (and perhaps even more to type them every time you want to run a container!). We can encode all those options into a helper file that'll "remember" them for us. Enter `docker-compose`, the piece of software that will read and interpret `compose.yml` files. An in depth introduction of `compose` is well beyond this session. Instead, we will simply whet your apetite to learn more about it by translating the command above into a `compose`-able file:

Listing 1 `compose.yml`

```
services:
 jupyter:
   image: darribas/gds_dev:10.0
   ports:
     - "8888:8888"
   volumes:
     - /home/dani:/home/jovyan/work
```

Paste the above into a text file that you place in your machine, name it `compose.yml`, and navigate in the command line to the folder where you have placed the file.

> ⚠ Warning
>
> Make sure to replace `/home/dani` on the volumes
>
> section by the path in the host you would like to mount into the running container.

```
docker compose up
```

This will spin up a container with the parameters specified on the file named `compose.yml` in the folder where the command is being run.

In the example above, this will launch a Jupyter server, and you can exit it with `Ctrl + C`. In other cases (e.g., see Rstudio case below), the server or app will run in the background and you will not see much activity in the command line. If you then want to spin down the app:

```
docker compose down
```

Popular images for (geographic) data science

`gds_env`

- `Frontend`: Jupyter Lab
- `Stack`: Python and R
- `URL`: https://darribas.org/gds_env/
- Launch:

Docker

```
docker run \
    --rm \
    -ti \
    -p 8888:8888 \
    -v ${PWD}:/home/jovyan/work \
    darribas/gds_dev:latest
```

Notes:

- `${PWD}` will pick up the working directory from where you run the command. Replace accordingly with the path to the host folder you want to mount.

- If you want to pin to a specific version of the image, replace `latest` by the version (e.g., `10.0`).

Compose

Listing 2 `compose.yml`

```yaml
services:
 jupyter:
   image: darribas/gds_dev:latest
   ports:
     - "8888:8888"
   volumes:
     - /home/dani:/home/jovyan/work
```

Notes:

- If you want to pin to a specific version of the image, replace `latest` by the version (e.g., `10.0`).
- Replace `/home/dani` with the path to the host folder you want to mount.

Rocker

- `Frontend`: Rstudio Server
- `Stack`: R
- `URL`: https://rocker-project.org/
- Launch:

Docker

```
docker run \
    --rm \
    -ti \
    -p 8787:8787 \
    -e PASSWORD=yourpassword \
    -v ${PWD}:/home/rstudio/work \
```

```
    rocker/geospatial:latest
```

Notes:

- `-e` signifies an "environment variable" that can pass the password into the container dynamically.
- `${PWD}` will pick up the working directory from where you run the command. Replace accordingly with the path to the host folder you want to mount.
- If you want to pin to a specific version of the image, replace `latest` by the version (e.g., `4.3.2`).

Compose

Listing 3 `compose.yml`

```yaml
services:
  rstudio:
    image: rocker/geospatial:latest
    ports:
      - "8788:8787"
    volumes:
      - /home/dani:/home/rstudio/work
    environment:
      PASSWORD: "mypassword"
```

Notes:

- If you want to pin to a specific version of the image, replace `latest` by the version (e.g., `10.0`).
- Replace `/home/dani` with the path to the host folder you want to mount.

`gdsrpy`

- `Frontend`: Rstudio Server
- `Stack`: R and Python
- `URL`: https://github.com/GDSL-UL/gdsr/

- Launch:

Docker

```
docker run \
    --rm \
    -ti \
    -p 8787:8787 \
    -e PASSWORD=yourpassword \
    -v ${PWD}:/home/rstudio/work \
    rocker/geospatial:latest
```

Notes:

- `${PWD}` will pick up the working directory from where you run the command. Replace accordingly with the path to the host folder you want to mount.
- If you want to pin to a specific version of the image, replace `latest` by the version (e.g., `4.3.2`).

Compose

Listing 4 `compose.yml`

```
services:
  rstudio:
    image: rocker/geospatial:latest
    ports:
      - "8788:8787"
    volumes:
      - /home/dani:/home/rstudio/work
    environment:
      PASSWORD: "mypassword"
```

Notes:

- If you want to pin to a specific version of the image, replace `latest` by the version (e.g., `10.0`).

- Replace `/home/dani` with the path to the host folder you want to mount.

Building your own images

Listing 5 `Dockerfile`

```
FROM darribas/gds_dev:10.0

ADD ./mydata.zip /home/jovyan/mydata.zip

RUN conda install longboard

RUN R -e "install.packages("officeverse", repos='http://cran.rstudio.com');"
```

See here for all commands you can use in a `Dockerfile`.

References

Rey, Sergio, Dani Arribas-Bel, and Levi John Wolf. 2023. *Geographic Data Science with Python*. CRC Press.