Docker cheatsheet

docker container vs image:

- can delte/close container; has no effect on image

command line input	description
`docker ps -a`	view any existing containers; confirm removed containers
`docker rm <container_id>`	delete the container (but not the image)
`docker images`	tells you what images are installed on your machine
`docker pull <dockerhub_username/repo_name>`	downloads a Docker image from Docker Hub
`docker run -it <dockerhub_username/repo_name>`	launch a container from the image and poke around
`docker run -it --rm -v <absolute path on laptop>:<rel container path> dockerhub_username/repo_name`	run a docker file; `-it` flag means run it interactively; `--rm` flag means remove automatically upon exit; -v flag mounts a volume of your laptop to the Docker container; note that can be named anything as long as it's lower case, e.g. /home/analysis
`exit`	leave the container

1. Create the Docker file

- from command line: `atom Docker`
- as a base image, use something like rocker/tidyverse for R or continuumio/anaconda3 for Python
- Installing R packages inside a Dockerfile
  - recommended practice:

```
RUN apt-get update -qq && apt-get -y --no-install-recommends install \
    && install2.r --error \
      --deps TRUE \
      PACKAGE_NAME
```

  - this strategy automatically installs any R package dependencies
- Installing Python packages inside a Dockerfile
  - For non-complicated packages try:

```
RUN pip3 install numpy
```

(Install things one at a time and test as you go.)

2. Build the Docker image locally

- From inside the directory containing the Dockerfile (e.g. project root):
  - `docker build --tag IMAGE_NAME:VERSION .`

3. Test the image locally

- *(NOT SURE ABOUT THIS)*
- `docker run IMAGE_NAME:VERSION`
- OR:
- `docker run --rm -it -e PASSWORD='<password>' IMAGE_NAME:VERSION /bin/bash`
- OR:
- `docker run --rm -it -e PASSWORD='<password>' -v <absolute path on laptop>:<rel container path> IMAGE_NAME:VERSION /bin/bash`
- if it doesn't work, delete the image with 'docker rmi IMAGE_NAME:VERSION, edit the Dockerfile and try to build and run it again.

4. **Test the Makefile interactively using Docker**
   - then `cd` into the `<rel container path>` and then run `make all`.
   - OR:
   - `docker run --rm -it -e PASSWORD=<password> -v <absolute path on laptop>:<rel container path> /bin/bash`
5. **Run Make non-interactively**
   - `docker run --rm -v <absolute path on laptop>:<rel container path> IMAGE_NAME:VERSION make -C '<rel container path>' all`
   - general form:
   - `docker run --rm -v <absolute path on laptop>:<rel container path> IMAGE_NAME:VERSION PROGRAM_TO_RUN PROGRAM_ARGUMENTS`
   - Note that the above executes the Makefile and writes the outputs to the "rel container path2" specified, which can be different from "rel container path."
6. **Once everything is working, push Dockerfile to GitHub**
7. **Build the Docker image using Automated builds on Docker Hub**
   - DockerHub.com
   - Select "Create" > "Create an Automated build" & follow the instructions
   - Go to the Docker Hub repository created, click on "Build Settings", and on that page, click "Trigger"
   - After several minutes, your Docker Hub repository should say "AUTOMATED BUILD" if it successfully built and you should be able to view the Dockerfile on Docker Hub as well.
   - To verify, delete the local image and try pulling it from Docker Hub
     - `docker pull <dockerhub_username/repo_name>`
8. **Test running from the Docker image**
   - `docker run --rm -it -e PASSWORD='<password>' -v <absolute path on laptop>:/<rel container path> <dockerhub_username/repo_name> /bin/bash`
   - `make -C '<rel container path>' all`
   - `make -C '<rel container path>' clean`
9. **Final test that it runs on its own:**
   - clone/download the repository, use the command line to navigate to the root of the project locally, and then type the following:
   - `docker run --rm -v <absolute path on laptop>:<rel container path> <dockerhub_username/repo_name> make -C '<rel container path>' clean`
   - `docker run --rm -v <absolute path on laptop>:<rel container path> <dockerhub_username/repo_name> make -C '<rel container path>' all`
   - note that `clean` needs to be run before `all` if the cloned repo had the files to start with