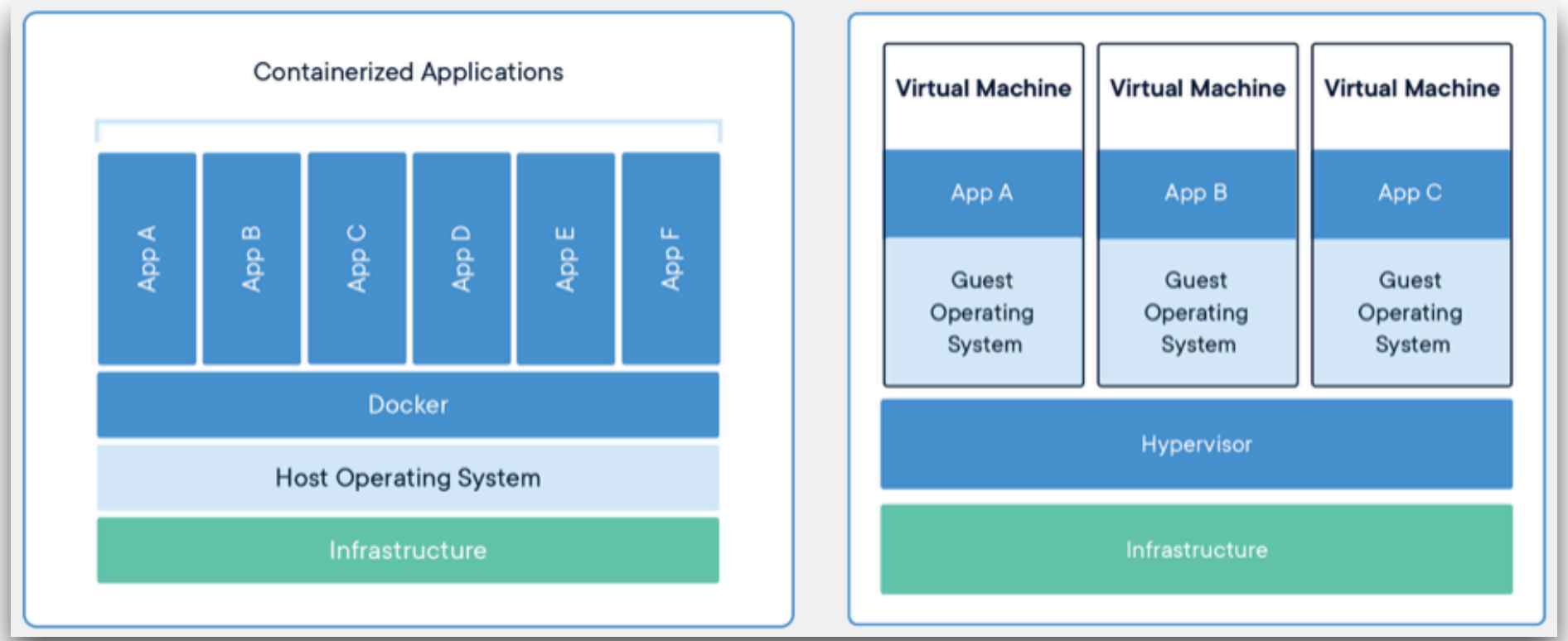
Machine Learning Environments with *Docker*

Docker allows you to work with a containerized environment that still interfaces with your local file system but does not interfere with your local libraries (e.g. tensorflow==1.5 or numpy==1.19). In particular, this makes it easy to share projects with others: just share the according docker image - no further installations (other than a one-time docker installation) are required.

Docker containers are similar to virtual machines, but containers require less space on disk and thus are more portable and efficient [1].

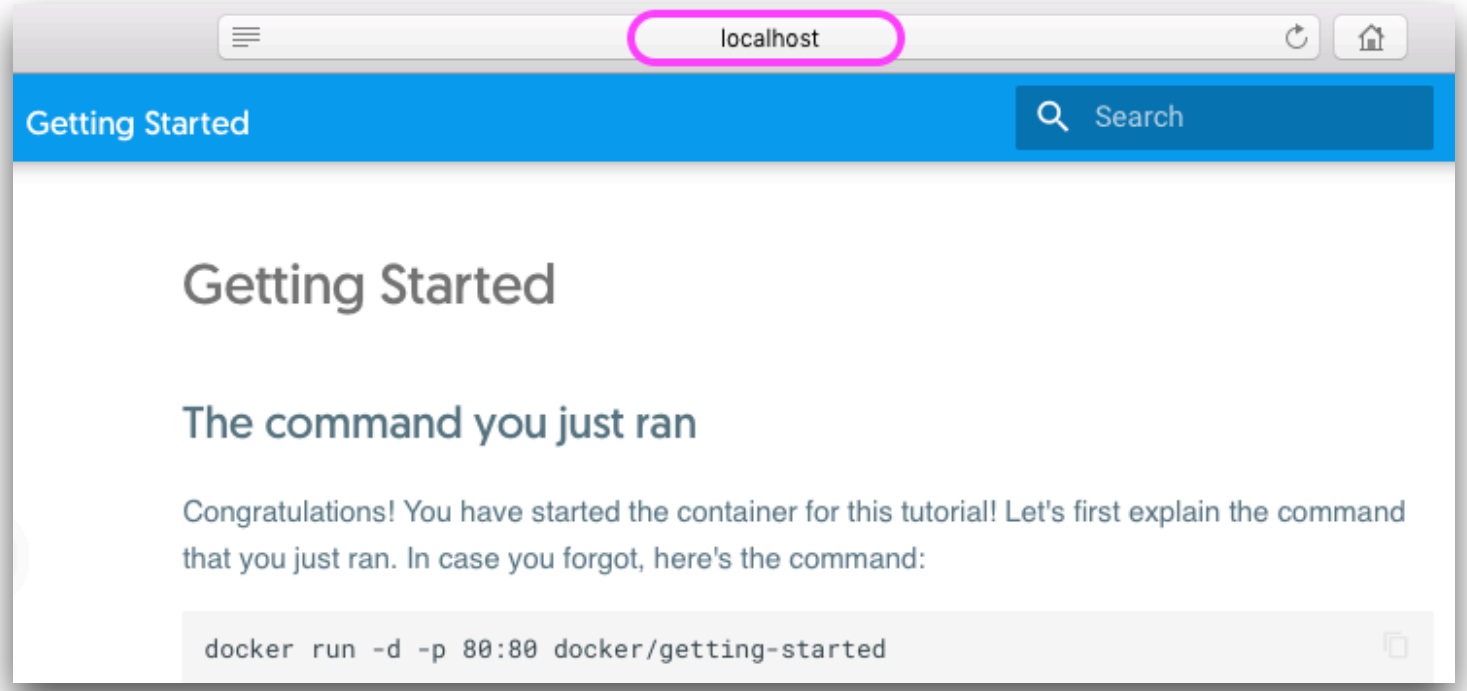


start >>

1. Go to https://docs.docker.com/get-docker and install docker on your machine [2].

2. Start docker and then run *docker run -dp 80:80 docker/getting-started* in a terminal. After that, open a browser window and navigate to *localhost*. [3]

```
[redacted]$ docker run -dp 80:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
cbdbe7a5bc2a: Pull complete
85434292d1cb: Pull complete
75fcb1e58684: Pull complete
2a8fe5451faf: Pull complete
42ceeab04dd4: Pull complete
bdd639f50516: Pull complete
c446f16e1123: Pull complete
Digest: sha256:79d5eae6e7b1dec2e911923e463240984dad111a620d5628a5b95e036438b2df
Status: Downloaded newer image for docker/getting-started:latest
2496c3bb4287d8fedb31134e79ed87a6a0776c7b7947bbeb379f95e23d895afd
```



continue >>

3. The tutorial you should see now is super useful to get first hands-on experience with docker. You do not have to understand everything immediately. Go through the sections in a linear fashion, focussing on dockerfile, uploading to docker hub, and container volumes.

4. In the following, the goal is to build a *docker image* that provides a machine learning environment with all the relevant libraries and an interface to our local file system so we can read/write data that persists beyond the runtime of a *docker container*. We will use a *Dockerfile* [4] to build the image (a docker compose [5] file would also work but that is beyond the scope of this tutorial).

5. We stick to the convention [6] of naming the dockerfile *Dockerfile* (with no file extension). Here is the final dockerfile (built with the *brackets* editor [7]).

/Users/[redacted]/Desktop/Dockerfile (Erste Schritte) — Brackets

```
1  # Use the specified pytorch image as base image, update apt-get, and install python3-pip.
2  FROM pytorch/pytorch:1.5.1-cuda10.1-cudnn7-runtime
3  RUN apt-get update && apt-get -y update
4  RUN apt-get install -y python3-pip
5  # The current directory is "worskapce". Copy over the requirements.txt file, and install the requirements via pip3.
6  COPY requirements.txt .
7  RUN pip3 -q install pip --upgrade
8  RUN pip3 install torchaudio
9  RUN pip3 install -r requirements.txt
10 # In "workspace", create the directory "container-vol" (we will mount our volume to "container-vol").
11 RUN mkdir container-vol
12 # Start jupyter notebook using the specified port.
13 CMD ["jupyter", "notebook", "--port=8888", "--no-browser", "--ip=0.0.0.0", "--allow-root"]
```

continue >>

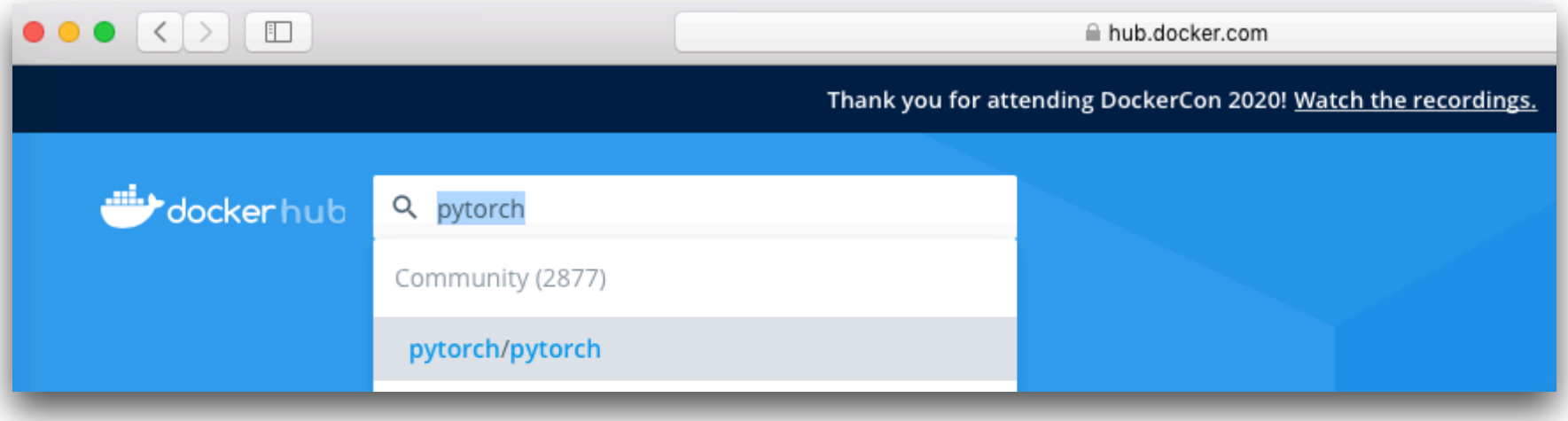
5.1. The first line begins with # and thus is a comment. All lines that begin with # are comments.

```
1  # Use the specified pytorch image as base image, update apt-get, and install python3-pip.
```

5.2. The second line begins with the FROM instruction [8] and thus specifies the *base image*. A base image has no parent image which means that during runtime, its container is not running inside another container. An image can have multiple base images / FROM instructions. (This means that at runtime, docker would launch multiple containers that run in parallel to each other instead of inside one another). The FROM instruction usually comes first in a dockerfile. It can only be preceded by the ARG instruction [9] that provides an argument for the subsequent FROM instruction.

```
2  FROM pytorch/pytorch:1.5.1-cuda10.1-cudnn7-runtime
```

A great place to start looking for a suitable base image is docker hub (which is covered by the getting-started tutorial [3]). Once a reasonable base image has been identified further fine tuning can be done with additional instructions in the dockerfile. Here, we use the latest (as of this writing) official PyTorch image.



continue >>

5.3. The third line begins with the RUN instruction [10] and thus runs a shell command on top of all commands that have implicitly run inside the base image. Here, we use it to update *apt-get*.

```
3  RUN apt-get update && apt-get -y update
```

5.4. The fourth line begins with another RUN instruction that installs pip for python3. (Our base image has python3 already installed).

```
4  RUN apt-get install -y python3-pip
```

5.5. The fifth line is another comment

5.6. Thes sixth line uses the COPY instruction [11]. It copies files and/or folders from the source location on the host (first argument) to the destination in the container image (second argument). Here, the file *requirements.txt* is located in the same folder on the host as the dockerfile and we copy it into the current folder in the container image.

```
6  COPY requirements.txt .
```

5.7. The seventh line upgrades pip.

```
7  RUN pip3 -q install pip --upgrade
```

5.8. The eighth line installs *torchaudio* [12] via pip.

```
8  RUN pip3 install torchaudio
```

5.9. The ninth line runs pip to install the libraries in the file *requirements.txt* [13], which we have copied from the host into the container image in line 6. In particular, this will install *jupyter* as it is listed in the requirements file.

```
9  RUN pip3 install -r requirements.txt
```

continue >>

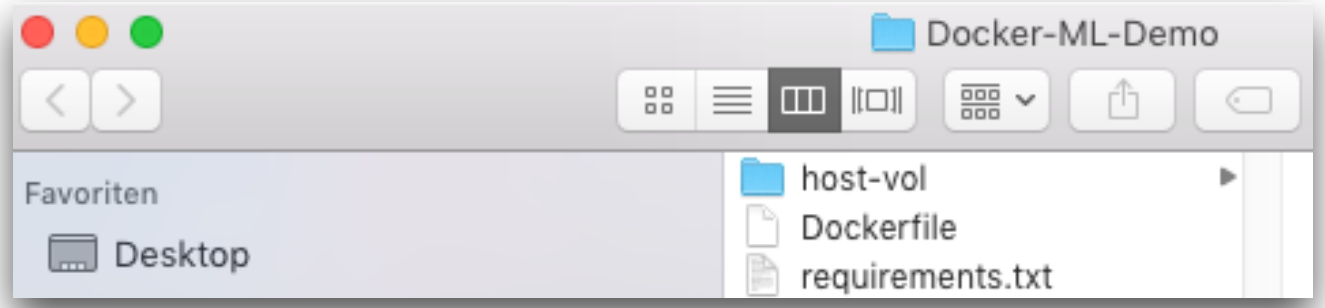
5.10. The tenth line is another comment.
5.11. The eleventh line runs the *mkdir* command to create a folder, named *container-vol*, on the docker image. Any folder name would work but here, *container-vol* is chosen because at runtime, we will mount a volume [14] on the host machine to this folder in the image.

```
11  RUN mkdir container-vol
```

5.12. The twelfth line is the last comment.
5.13. The thirteenth line uses the CMD instruction [15]. There can be only one CMD instruction in a dockerfile. Unlike the RUN instruction, CMD does not do anything when building the docker image from the dockerfile. Instead, CMD specifies what happens when a container is invoked from the image. Here, we use CMD to start jupyter notebook, map the output to port 8888 and use no browser (in the container – however we will map port 8888 of the container to a port on the host and use a browser on the host to display the output of jupyter.)

```
13  CMD ["jupyter", "notebook", "--port=8888", "--no-browser", "--ip=0.0.0.0", "--allow-root"]
```

6. The folder *Docker-ML-Demo* contains the files *Dockerfile* and *requirements.txt* that we need to build our docker image and the folder *host-vol* whose contents we will mount to the image's container-vol at runtime.



continue >>

7. Navigate the terminal into the Docker-ML-Demo folder and run
   docker build -t jupytorch .
   to build [16] the docker image. Here, *docker build* tells docker to build an image and the *-t* flag allows the user to give the image a name. Here, the image is named *jupytorch*. The final dot tells docker that the dockerfile is located in the current directory.

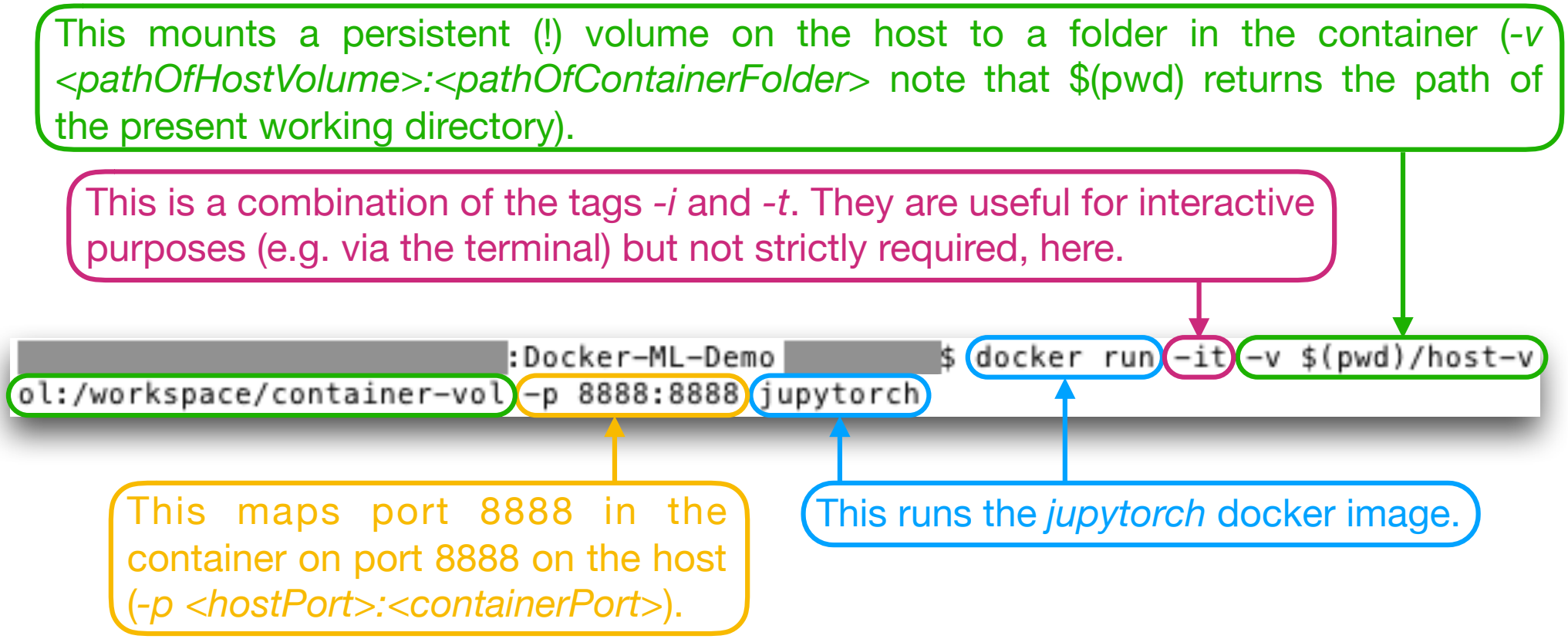
docker is running →

Docker-ML-Demo — -bash — 87×34

```
[                              Docker-ML-Demo          $ ls                    ]
Dockerfile                host-vol                  requirements.txt
[                              Docker-ML-Demo          $ docker build -t jupytorch .  ]
Sending build context to Docker daemon  174.6kB
Step 1/9 : FROM pytorch/pytorch:1.5.1-cuda10.1-cudnn7-runtime
 ---> d89e4943dd55
Step 2/9 : RUN apt-get update && apt-get -y update
 ---> Using cache
 ---> eb149fdcc592
Step 3/9 : RUN apt-get install -y python3-pip
 ---> Using cache
 ---> 477bcd8f543f
Step 4/9 : COPY requirements.txt .
 ---> Using cache
 ---> a1a3f7599e0d
Step 5/9 : RUN pip3 -q install pip --upgrade
 ---> Using cache
 ---> 8d599178ff1b
Step 6/9 : RUN pip3 install torchaudio
 ---> Using cache
 ---> 4faa7df6dd4d
Step 7/9 : RUN pip3 install -r requirements.txt
 ---> Using cache
 ---> 16e1084b8e30
Step 8/9 : RUN mkdir container-vol
 ---> Using cache
 ---> 99007f0c6809
Step 9/9 : CMD ["jupyter", "notebook", "--port=8888", "--no-browser", "--ip=0.0.0.0", "
--allow-root"]
 ---> Using cache
 ---> d60d40830a03
Successfully built d60d40830a03
Successfully tagged jupytorch:latest
                               Docker-ML-Demo          $
```

continue >>

8. Instantiate a container in which the *jupytorch* image runs via *docker run -it -v $(pwd)/host-vol:/workspace/container-vol -p 8888:8888 jupytorch*. See Ref. [17] for details on the *docker run* command.

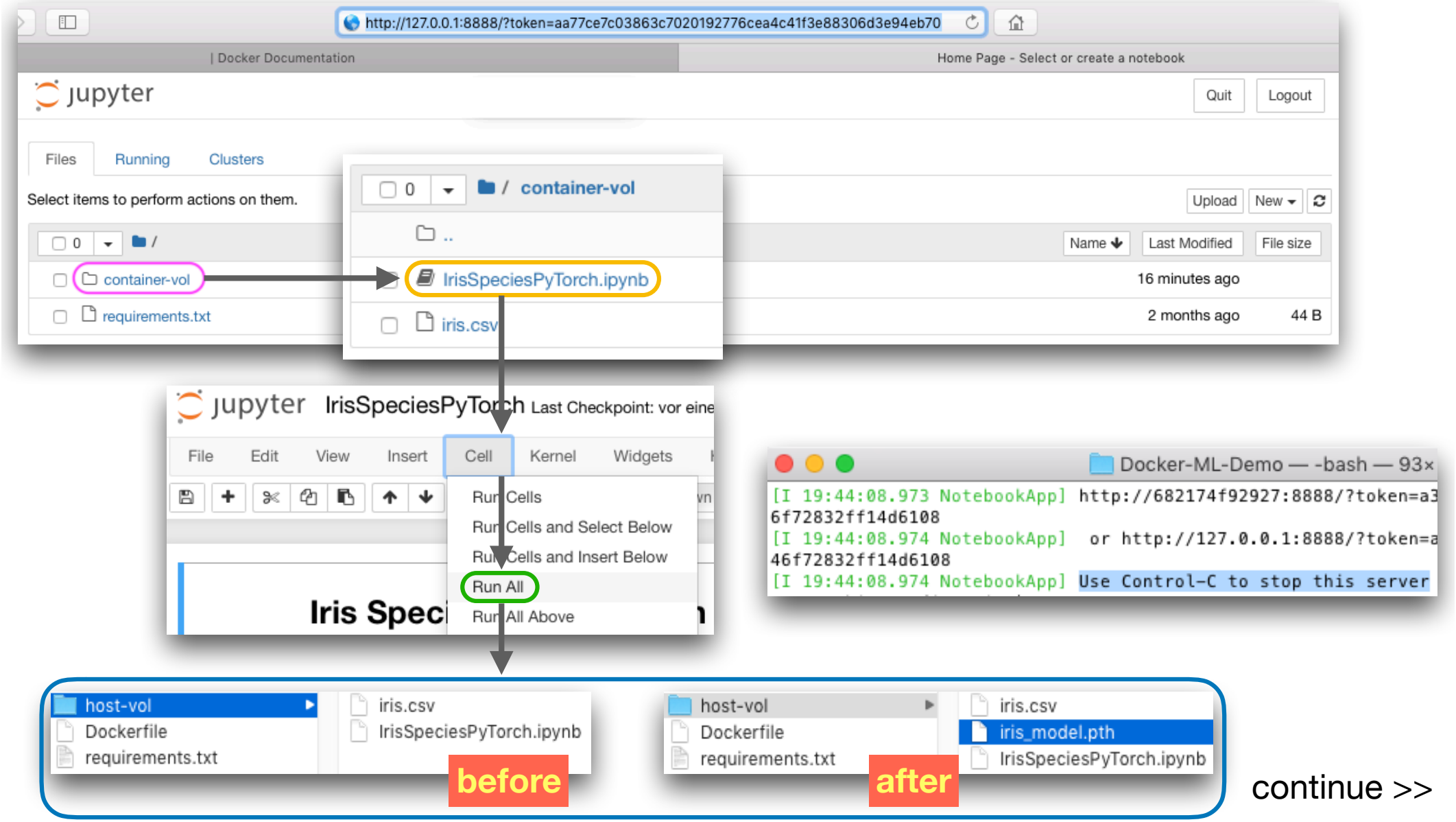


8.1. Now, a container is up and running the jupytorch image. To access jupyter, copy the link (see pic below) and paste it into the browser address of the host.

```
To access the notebook, open this file in a browser:
    file:///root/.local/share/jupyter/runtime/nbserver-1-open.html
Or copy and paste one of these URLs:
    http://fbf0e09a34dc:8888/?token=aa77ce7c03863c7020192776cea4c41f3e88306d3e94eb70
 or http://127.0.0.1:8888/?token=aa77ce7c03863c7020192776cea4c41f3e88306d3e94eb70
```

continue >>

8.2. Wait until jupyter appears in your browser > click on *container-vol* (This is the folder to which we have mounted the volume *host-vol* on the host.) > the files inside the volume appear > click on *IrisSpeciesPyTorch.ipynb* (This is indeed the file on the host machine. Changes to this file will persist on the host.) > click on Cell > click *Run All* > the file *iris_model.pth* is created > double-type Control-C in the terminal to stop jupyter and the docker container > *iris_model.pth* persists on the host.

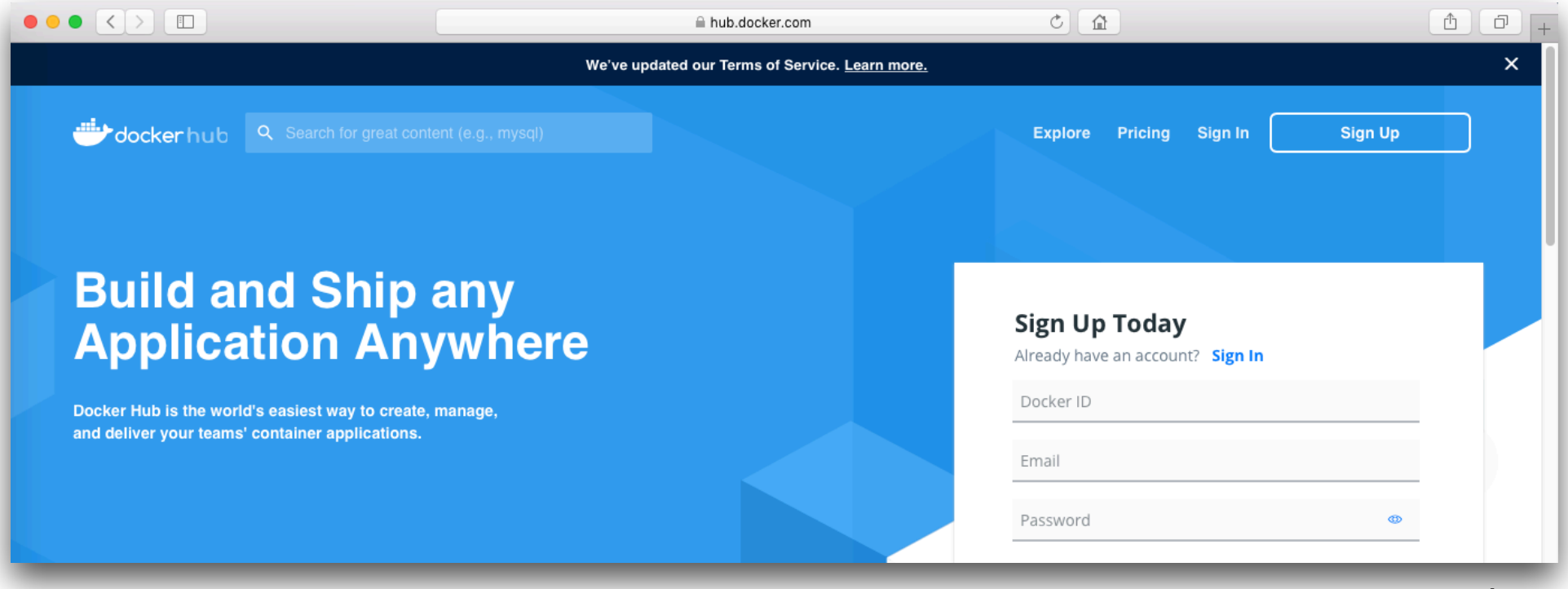


continue >>

9. There are several ways of sharing an image. The most lightweight one is probably to simply share the dockerfile and its context (e.g. requirements.txt). Alternatively, the image can be saved as a file or it can be uploaded to docker hub.

9.1. To save [18,19] the newly built *jupytorch* image as a local file, run
docker save -o jupytorch.tar jupytorch
in the terminal. The *-o* flag specifies that the output of saving the *jupytorch* image should be a file, namely *jupytorch.tar*. Running the command
docker load -i jupytorch.tar
will load the image from the jupytorch.tar file into docker [18,20].

9.2. Uploading an image to docker hub [21] requires a docker hub account. Once such …



continue >>

… an account exists, the image can be uploaded to docker via the following steps:

> list your local docker images via *docker images*

> use the listed ID of the docker image you want to push to docker hub to tag it with a name via *docker tag <ID> <dockerHubAccount>/<dockerImageName>:<tag>*
(the tag *:<tag>* can be omitted)

```
[redacted]:Docker-ML-Demo [redacted]$ docker images
REPOSITORY                  TAG                            IMAGE ID            CREATED             SIZE
jupytorch                   latest                         d60d40830a03        2 months ago        6.14GB
docker/getting-started      latest                         1f32459ef038        2 months ago        26.8MB
pytorch/pytorch             1.5.1-cuda10.1-cudnn7-runtime  d89e4943dd55        3 months ago        3.14GB
[redacted]:Docker-ML-Demo [redacted]$ docker tag d60 mrdroth/jupytorch:latest
[redacted]:Docker-ML-Demo [redacted]$ docker images
REPOSITORY                  TAG                            IMAGE ID            CREATED             SIZE
jupytorch                   latest                         d60d40830a03        2 months ago        6.14GB
mrdroth/jupytorch           latest                         d60d40830a03        2 months ago        6.14GB
```

> log in to your docker hub account via *docker login -u=<dockerHubAccount>*

```
[redacted]Docker-ML-Demo [redacted]$ docker login -u=mrdroth
Password:
Login Succeeded
```

> push the image to your docker hub account via
docker push <dockerHubAccount>/<dockerImageName>:<tag>
(the tag *:<tag>* can be omitted)

```
[redacted]:Docker-ML-Demo [redacted]$ docker push mrdroth/jupytorch
The push refers to repository [docker.io/mrdroth/jupytorch]
b8607271d909: Preparing
6ad182e8e06d: Pushed
9495162538b9: Pushed
```

Now, you can logout from docker hub via *docker logout*. Being logged in is not required for pulling your image from docker hub, so anybody can pull it via *docker pull <yourDockerHubAccount>/<yourDockerImageName>:<yourTag>* (unless you choose to make it private under the settings on its docker hub web page).

continue >>

References

[1] https://www.docker.com/resources/what-container
[2] https://docs.docker.com/get-docker
[3] https://github.com/docker/getting-started
[4] https://docs.docker.com/engine/reference/builder
[5] https://docs.docker.com/compose/compose-file
[6] https://stackoverflow.com/questions/26077543/how-to-name-dockerfiles
[7] http://brackets.io
[8] https://docs.docker.com/engine/reference/builder/#from
[9] https://docs.docker.com/engine/reference/builder/#understand-how-arg-and-from-interact
[10] https://docs.docker.com/engine/reference/builder/#run
[11] https://docs.docker.com/engine/reference/builder/#copy
[12] https://pytorch.org/audio
[13] https://pip.pypa.io/en/stable/reference/pip_install/#example-requirements-file
[14] https://docs.docker.com/storage/volumes/#choose-the--v-or---mount-flag
[15] https://docs.docker.com/engine/reference/builder/#cmd
[16] https://docs.docker.com/engine/reference/commandline/build
[17] https://docs.docker.com/engine/reference/commandline/run
[18] https://dockerlabs.collabnix.com/beginners/saving-images-as-tar
[19] https://docs.docker.com/engine/reference/commandline/save
[20] https://docs.docker.com/engine/reference/commandline/load
[21] https://hub.docker.com

Important Docker Commands

docker images	lists all locally available docker images
docker ps -a	lists all containers (-a => including stopped ones)
docker rmi imageID	removes the image with ID *imageID*
docker stop containerID	stops the running container with ID *containerID*
docker rm containerID	removes the (stopped!) container with ID *containerID*
docker rm -f containerID	removes the container with ID *containerID*
docker ID inspect	returns details about the docker object (container, image, volume, etc.) that has the ID *ID*
docker --help	help on anything docker, including further docker commands