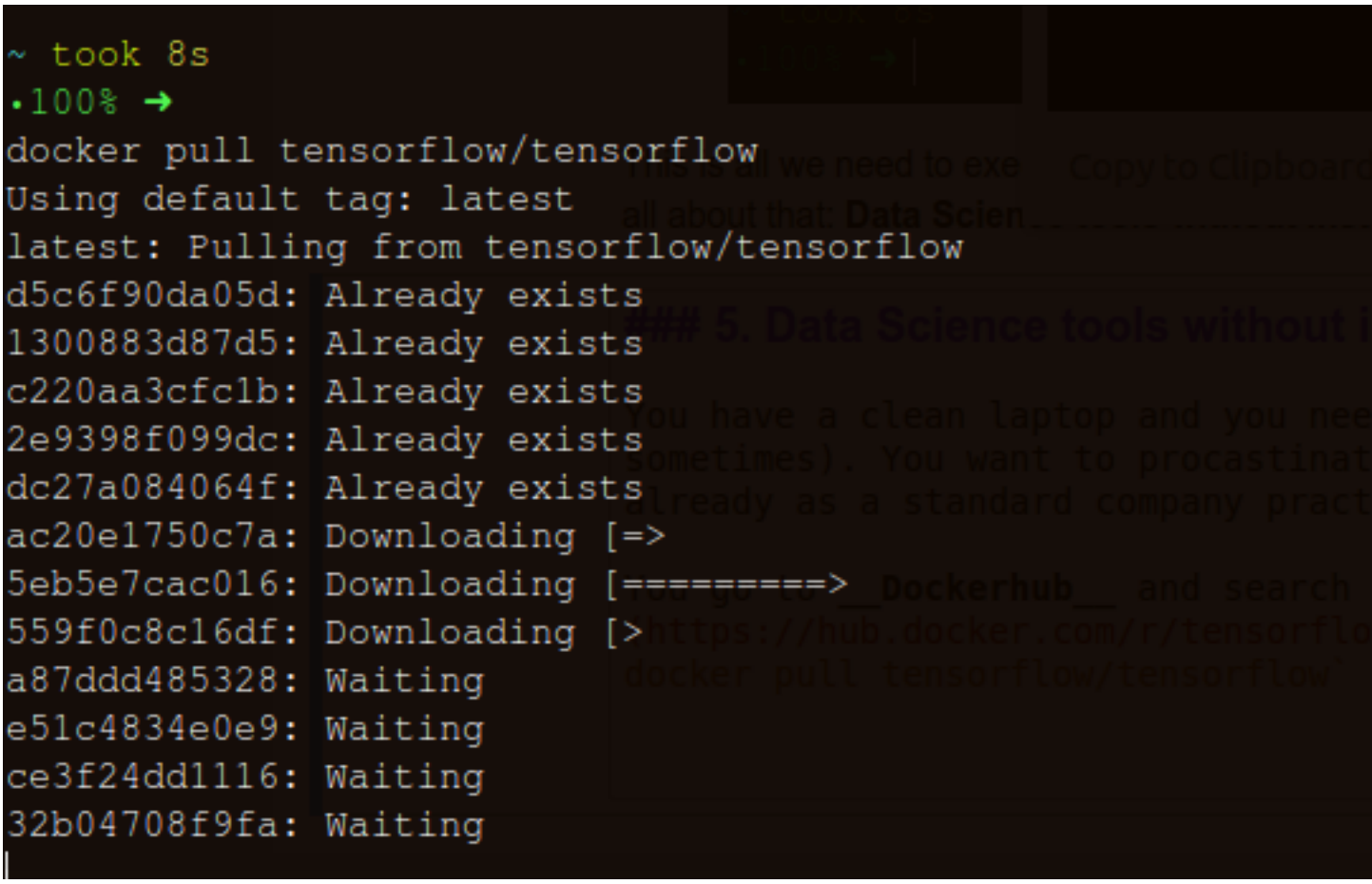
Then use the Docker for Mac menu to create, view, or navigate directly to your Cloud resources, including organizations, repositories, and swarms.

Data Science tools without installation:

You have a clean laptop and you need to install **TensorFlow** in your system, but you are lazy (yes we all are sometimes). You want to procrastinate and not install things on your laptop, but you have **Docker** installed already as a standard company practice. Hmm, interesting times, you ponder!

You go to **Dockerhub** and search for the official **Docker** image for **TensorFlow**. All you need to run on your terminal is: `docker pull tensorflow/tensorflow`



As discussed above (in **Docker Terminology** section), the **tensorflow** docker image is also a layered object that forms images. Once all the intermediate layers are downloaded, run: docker images to check whether our docker pull was successful.

```
~ took 8s
•100% ➜
docker pull tensorflow/tensorflow
Using default tag: latest
latest: Pulling from tensorflow/tensorflow
d5c6f90da05d: Already exists
1300883d87d5: Already exists
c220aa3cfc1b: Already exists
2e9398f099dc: Already exists
dc27a084064f: Already exists
ac20e1750c7a: Pull complete
5eb5e7cac016: Pull complete
559f0c8c16df: Pull complete
a87ddd485328: Pull complete
e51c4834e0e9: Pull complete
ce3f24dd1116: Pull complete
32b04708f9fa: Pull complete
Digest: sha256:4b014f1ddfdaf046a02c7477de5a2053f93d68f4d6dcf
Status: Downloaded newer image for tensorflow/tensorflow:lat

~ took 12m 38s
•100% ➜ docker images
REPOSITORY                                  TAG
hello-world                                 latest
pratos/ocr_python                           latest
<none>                                      <none>
tensorflow/tensorflow                       latest
ubuntu                                      16.04
python                                      3.6
python                                      3.6-alpine
alpine                                      latest
gcr.io/cloud-datalab/datalab                local
gcr.io/tensorflow/udacity-assignments       latest
tutum/nginx                                 latest
```

To run the image, run the command: `docker run -it -p 8888:8888 tensorflow/tensorflow`

[**NOTE:** At the time of writing, port 8888 was already used up so running it on 8889. You can run it on any port though *shrugs*]

Now the above `docker run` command packs in a few more command line argurments. A few which you need to know better are as follows:

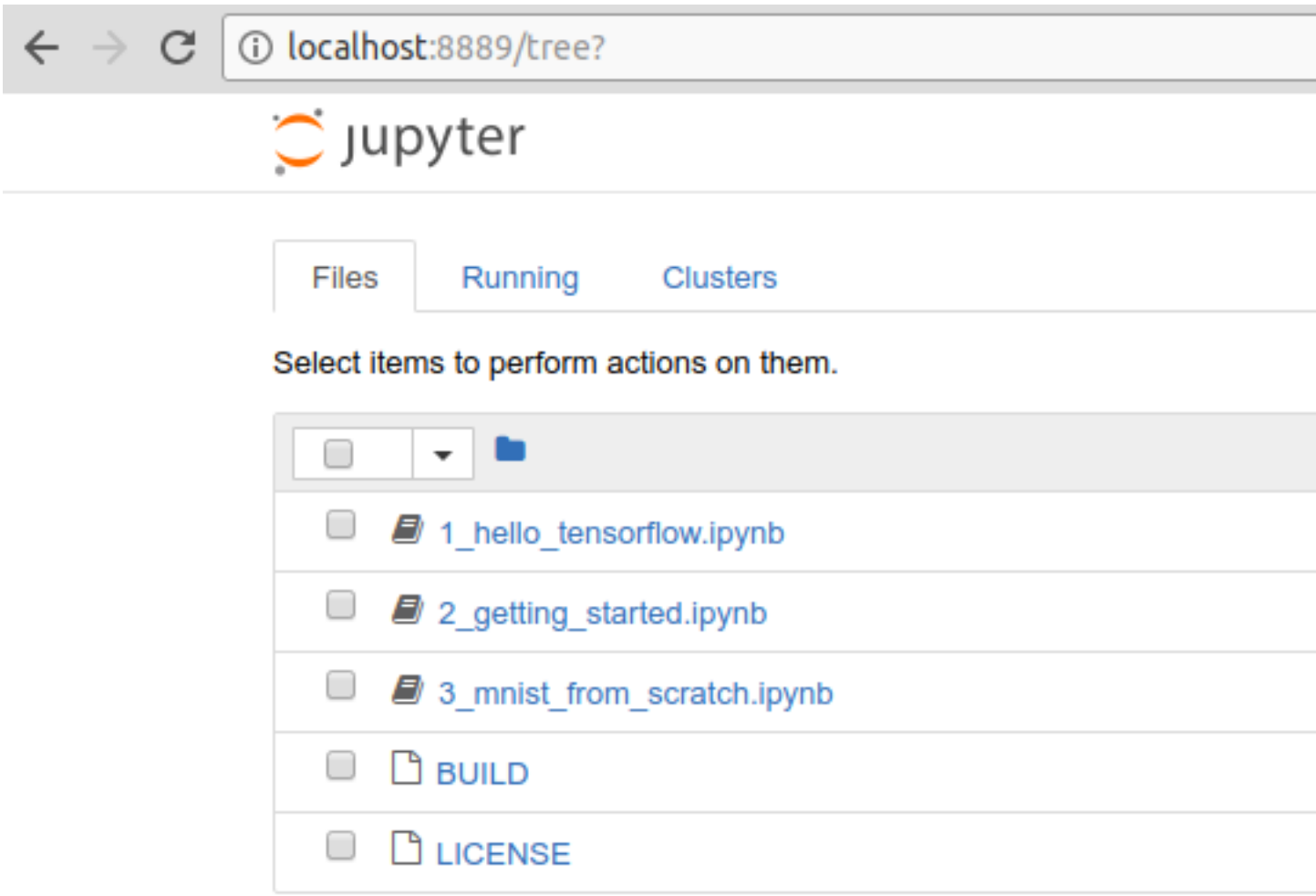
1 `i` is running the image interactively.

2 `t` is to run bash inside the docker container created.

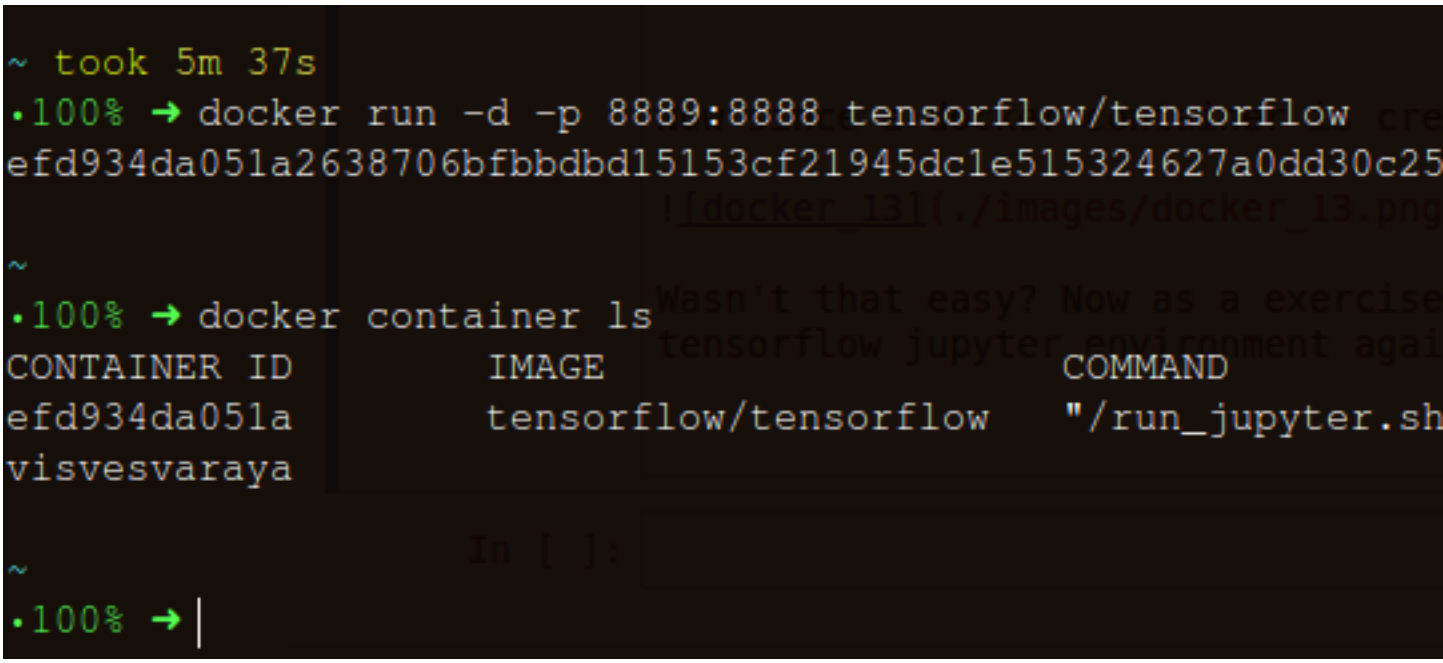
3 `p` is connect/publish the container ports to host. Here localhost:8888 to 8888 of container.

4 `d` is to run the container in **detached** mode i.e. the container would run in the background unlike the above (`i` where once you stop the process the container gets automatically removed).

Now since a docker container is created, you can visit: http://localhost:8889 where you can try out `tensorflow`.



Wasn't that easy? Now as a exercise, replace `-it` in the `docker run` command by `-d`. See whether you can get the tensorflow jupyter environment again or not?
You should get the following outputs as in the screenshot below:



Exercise: Create more containers with different ports using the docker run command and see how many get created.

6. Your first Docker Image

We as Data Science folks are picky about what tools we use for our analysis, some like to work with **R** while others prefer **Python**. Personally, I'd whine about the above TensorFlow image. I don't know what's there in it (unless I look at the source code i.e. the **Dockerfile** aka **recipe**). Tensorflow isn't enough on it's own, suppose you want to use **OpenCV** too and maybe **scikit-learn** & **matplotlib**.

Let's see how to create your own custom **TensorFlow** image!

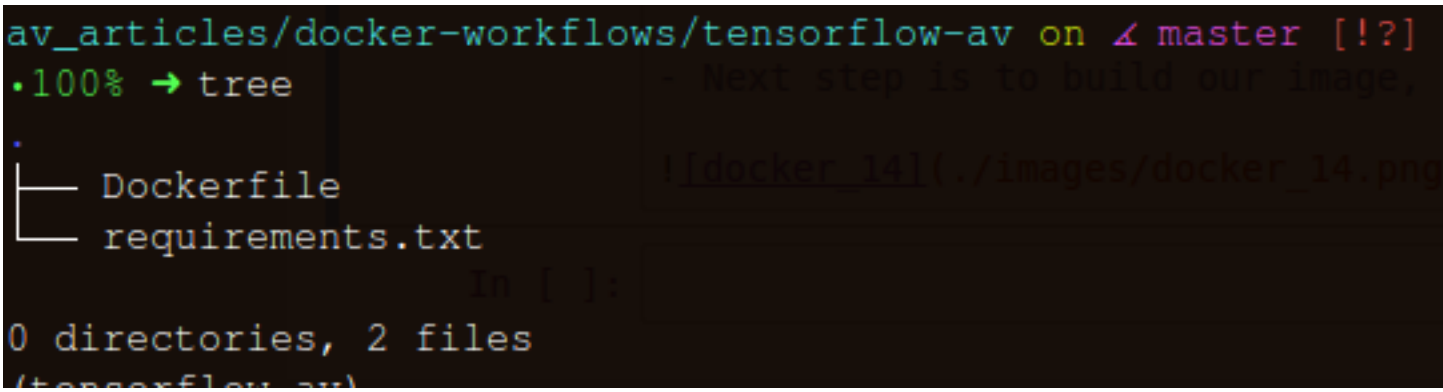
- First thing you need is to create a requirements.txt file. For reference, below is the file that you might want to use: requirements.txt
- Our **Dockerfile** would be comprised of the below components:
  - For the base image, we'll use the **official docker image for python i.e. python:3.6**.
  - Command to update the source repositories (the image uses **Debian** distribution).\
  - Copy the requirements.txt file and pip install the python libraries from the **requirements.txt** file.

Command to expose the ports.
Command to run the jupyter notebook command.

- The final **Dockerfile** would look as below:

```
# Base image
FROM python:3.6

# Updating repository sources
RUN apt-get update

# Copying requirements.txt file
COPY requirements.txt requirements.txt

# pip install
RUN pip install --no-cache -r requirements.txt

# Exposing ports
EXPOSE 8888

# Running jupyter notebook
# --NotebookApp.token ='demo' is the password
CMD ["jupyter", "notebook", "--no-browser", "--ip=0.0.0.0", "--allow-root", "--
NotebookApp.token='demo'"]
```

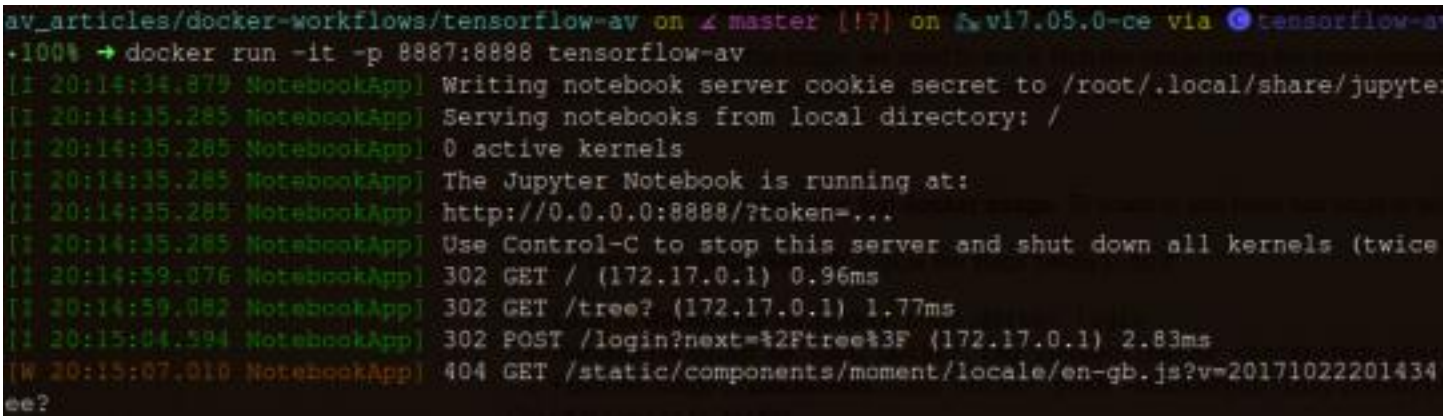
- Next step is to build our image, below is the tree structure that can be followed:



- To build the image, run: docker build -t tensorflow-av . (**Note:**-t is to tag the image as you wish too. You can version it as well, eg: docker build -t tensorflow-av:v1 .
- The logs for all the run is provided here. Once the entire process is completed, the image will be visible in your local docker registry. Run: docker images to check!

```
av_articles/docker-workflows/tensorflow-av on  master [!?]
•100% ➜ docker images
REPOSITORY                          TAG
tensorflow-av                       latest
```

- Now that you have created the image, we need to test it. Run the image using the same command you used to run the original **tensorflow docker image**. Run: docker run -p 8887:8888 -it tensorflow-av

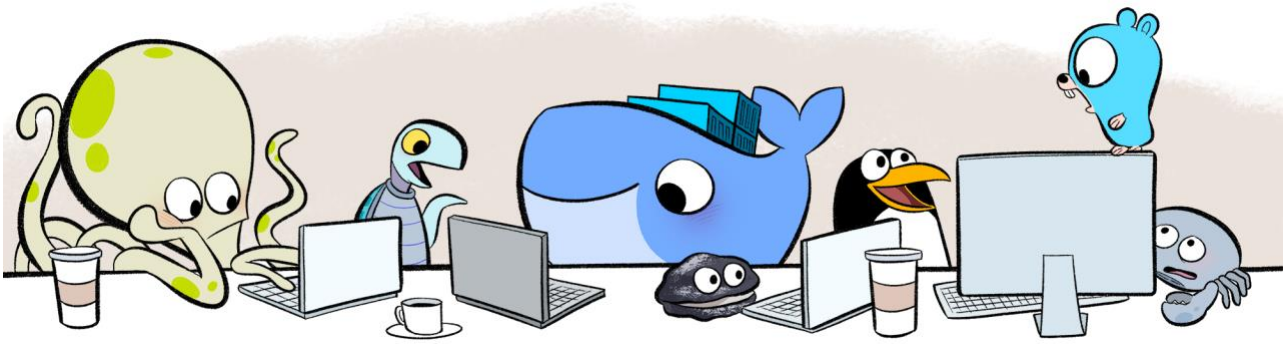


- Congratulations! You have made your first **docker image**. To share it, you have two ways in which you could do it:
  - Upload the image to **Dockerhub**. Follow the steps below to do it:
    - Login to **Dockerhub** via terminal: sudo docker login
    - Rename the docker file: sudo docker tag tensorflow-av <dockerhub-id>/tensorflow-av
    - Push the image to **Dockerhub**: sudo docker push <dockerhub-id>/tensorflow-av
  - Export the image to **.tar file**.
    - docker save <dockerhub-id>/tensorflow-av > <path>/tensorflow-av.tar
  - We can even export the container to a **.tar file**, along with all the running instances/state and other meta-data.
    - docker export <container-id> > <path>/tensorflow-av-run1.tar

7. Docker Eco-system

Docker provides a good support to build up from a prototype level

scale to production levels. Purely from a deployments perspective: **docker-machine**, **docker-compose** & **docker-swarm** are components that help achieve that.



Source: Official Docker Blog

- Want to take your ML API & deploy it to **any cloud provider**? **docker-machine** helps you do that.
- Your deployed API is growing in usage, want to scale it up? **docker-swarm** is there to help you do it without many changes.
- Want to use multiple Docker images in a single application? **docker-compose** makes it possible for you to do that!