Dockerizing your Jupyter Notebooks

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. You can learn more about Jupyter on their website.

We can Dockerize Jupyter notebooks in a couple of different ways but in this module, we'll focus on containerize the Jupyter development lab.

Step 01 - Create a Dockerfile

Navigate to your working directory and create an empty Dockerfile.

```
FROM pytorch/pytorch:latest

RUN pip install jupyterlab

CMD ["jupyter", "lab", "--ip=0.0.0.0", "--port=9999", "--allow-root",
"--NotebookApp.token=''",  "--NotebookApp.password=''"]
```

We use the pytorch base image. Next we install jupyterlab. Then we run the command to start the JupyterLab by specifying the IP, port, give root permissions, the notebook token and notebook password.

Step 02 - Running Jupyter Lab Container

Docker has two options for containers to store files in the host machine, so that the files are persisted even after the container stops: volumes, and bind mounts.

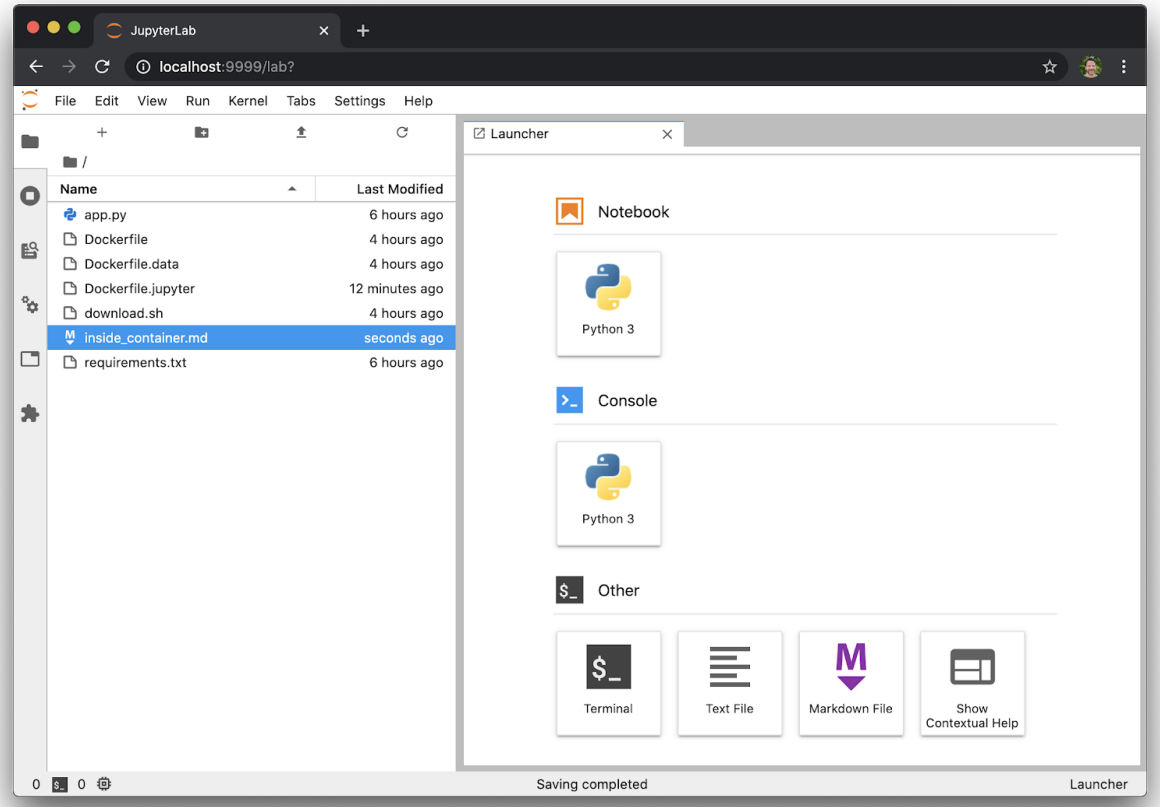
Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While bind mounts are dependent on the directory structure and OS of the host machine, volumes are completely managed by Docker.

We will use a bind mount in this module to persist our data. Bind mounts have been around since the early days of Docker. When you use a bind mount, a file or directory on the host machine is mounted into a container. The file or directory is referenced by its absolute path on the host machine.

```
$ docker run --rm -p 9999:9999 -v $(pwd):/workspace docker-ml:jupyter
```

Step 03 - Connect to Jupyter Lab

Open your favorite browser and navigate to http://localhost:9999. You will be presented with the Jupyter Lab interface.



Step 04 - Sharing files between host and Notebook

Since we started the container with a bind mount, we can now view files that are in our local working directory and those that are created inside of the container.

Click the “File” menu and then hover over the “New” item. Now click on the “Markdown file” menu item.

Enter “Hello from container land.” inside the newly created file and then save it.

Open a new terminal and navigate to your working directory. List the directory’s contents.

```
$ cd /path/to/your/working directory/
$ ls -l
```

We can also create a file outside of the container and have that file shared inside of the container.

```
$ cat <<EOF >> outside.md
Hello from outside the container.
EOF
```

Go back to the web browser. You should now see the outside.md file located in your directory tree on the left navigation panel.