Lab 2 Jupyter and iPython

Docker

Docker is a tool that allows running "Container" type applications. They differ from virtual machines in that they do not virtualize an entire operating system, but instead create an isolated space that uses the operating system kernel on the host machine.
To install Docker, follow the instructions from: https://docs.docker.com/get-docker/
Once installed, we will use the Command Line Interface (CLI) to interact with it.

We will use Docker to run a Container that has Jupyter and Spark already installed. In today's lab we will only use pure Python, but further on we will use the same image to run Spark applications. This container is found in the Docker Hub repository:
https://hub.docker.com/r/jupyter/pyspark-notebook

Pull image

First we need to "pull" the image from Docker Hub.
We can run the following command:

```
      docker pull jupyter/pyspark-notebook
```

This image will be stored on disk and we can run it with Docker.

Docker Volumes

The Docker image we use uses a local folder (in the image) where the files we work with are saved. If we stop the application then this data will disappear.
To ensure data persistence we will use a folder on the host machine that we will "volume attach" to the Docker application.

Create a directory (anywhere on your machine) that we will use as the volume for the pyspark-notebook application.
If you want to do this from the command line:

```
      mkdir -p /home/adrian/bdt/jupyter-volume
```

The above command will create a folder called jupyer-volume in the /home/adrian/bdt directory. The path to this folder will need to be saved to be used as a volume.

To attach a volume to a Docker application we will use the -v argument when running the application. E.g.:

```
      -v /home/adrian/bdt/jupyter-volume:/home/jovyan/work
```

will associate the jupyter-notebook folder on the host machine with the /home/jovyan/work folder in the Container application.

Ports

Applications in Docker will expose an interaction interface over HTTP. Docker associates each container with its own set of ports, but if we want to access them from outside the application, then we will have to map these ports to some on the host machine. We can do this using the -p parameter when running the application. For example:

```
-p 80:8080
```

assumes that the application opens port 80 in the Container, and we will associate this port with port 8080 on the host machine. We will be able to access this endpoint at http://localhost:8080

Running the application

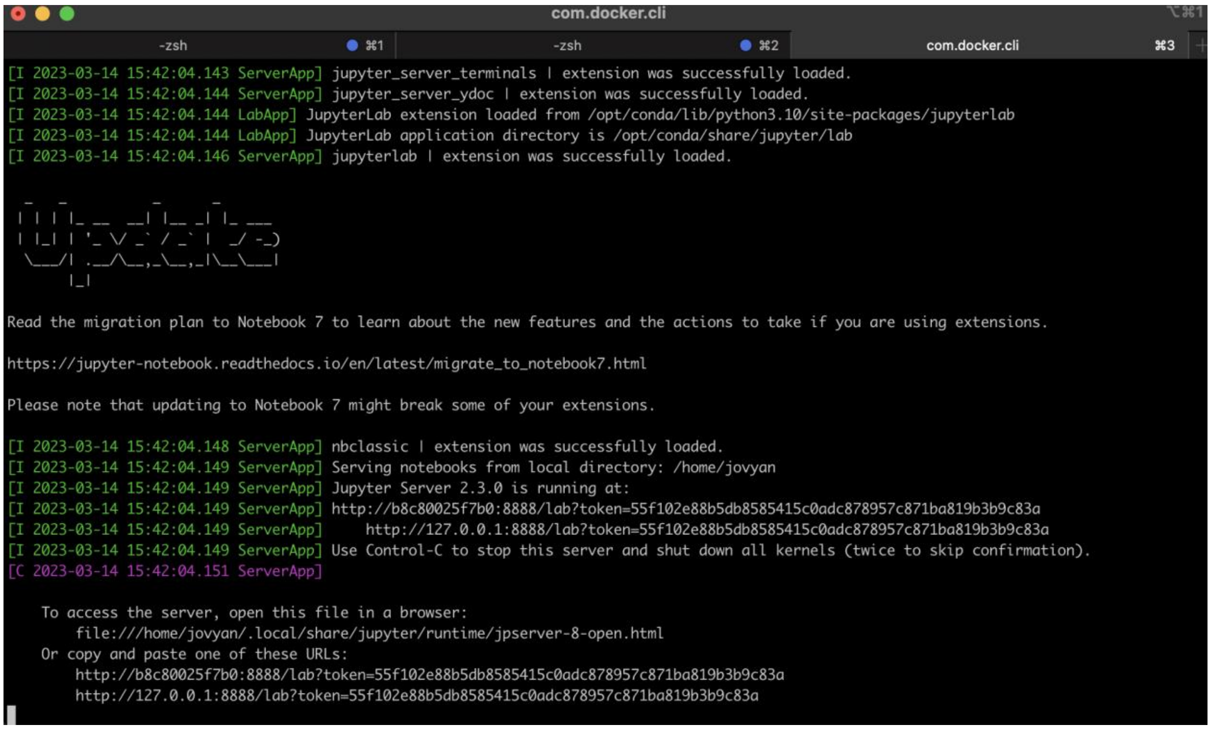
To run the application we can use the following command:

```
docker run -it -p 8888:8888 -p 4040:4040 -p 4041:4041 -v
~/Documents/data:/home/jovyan/work  jupyter/pyspark-notebook
```

Remember to replace the first argument of the -v parameter with the folder you created on the host machine.

In case of a successful execution, we will have the following message and we can click on the link below:

http://127.0.0.1/.....

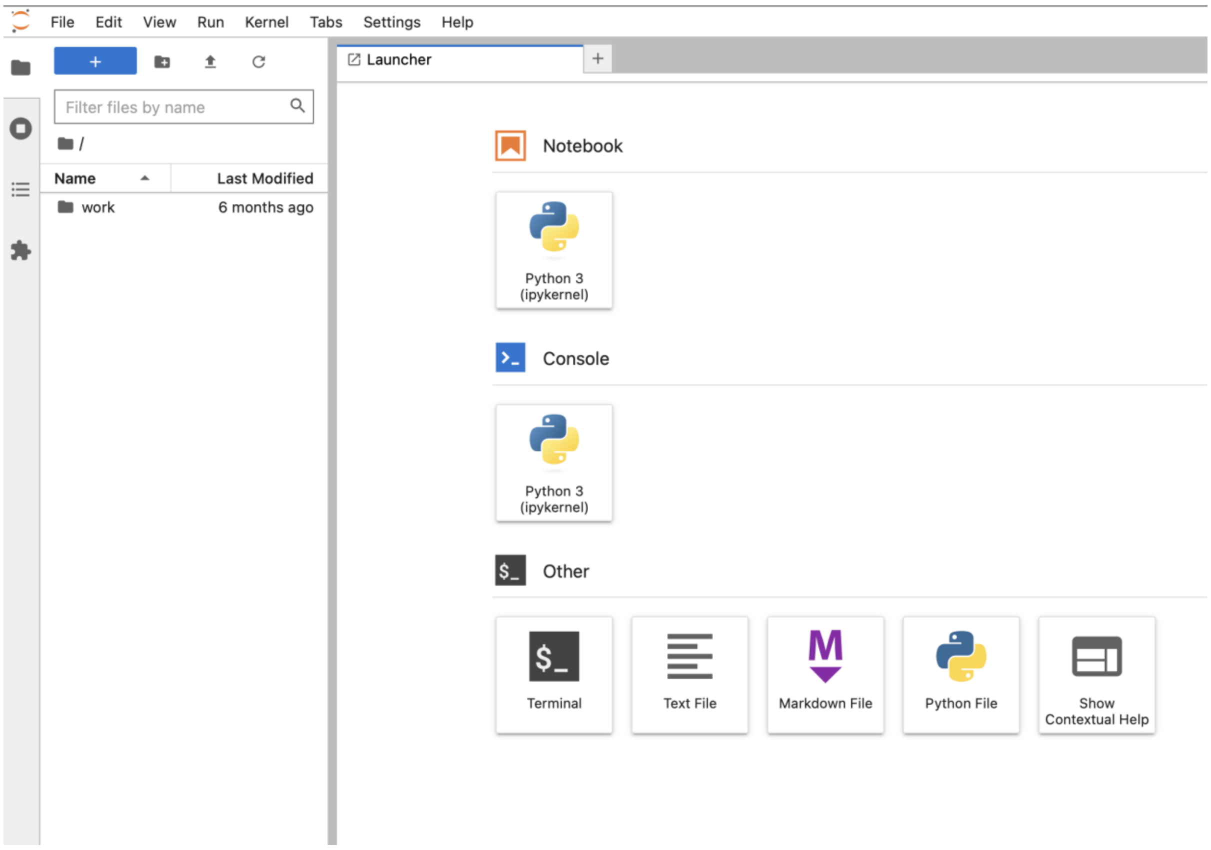


```
[I 2023-03-14 15:42:04.143 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2023-03-14 15:42:04.144 ServerApp] jupyter_server_ydoc | extension was successfully loaded.
[I 2023-03-14 15:42:04.144 LabApp] JupyterLab extension loaded from /opt/conda/lib/python3.10/site-packages/jupyterlab
[I 2023-03-14 15:42:04.144 LabApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 2023-03-14 15:42:04.146 ServerApp] jupyterlab | extension was successfully loaded.

 _   _           _       _
| | | |_ __   __| | __ _| |_ ___
| | | | '_ \ / _` |/ _` | __/ _ \
| |_| | |_) | (_| | (_| | ||  __/
 \___/| .__/ \__,_|\__,_|\__\___|
      |_|

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions.

https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html

Please note that updating to Notebook 7 might break some of your extensions.

[I 2023-03-14 15:42:04.148 ServerApp] nbclassic | extension was successfully loaded.
[I 2023-03-14 15:42:04.149 ServerApp] Serving notebooks from local directory: /home/jovyan
[I 2023-03-14 15:42:04.149 ServerApp] Jupyter Server 2.3.0 is running at:
[I 2023-03-14 15:42:04.149 ServerApp] http://b8c80025f7b0:8888/lab?token=55f102e88b5db8585415c0adc878957c871ba819b3b9c83a
[I 2023-03-14 15:42:04.149 ServerApp]     http://127.0.0.1:8888/lab?token=55f102e88b5db8585415c0adc878957c871ba819b3b9c83a
[I 2023-03-14 15:42:04.149 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2023-03-14 15:42:04.151 ServerApp]

    To access the server, open this file in a browser:
        file:///home/jovyan/.local/share/jupyter/runtime/jpserver-8-open.html
    Or copy and paste one of these URLs:
        http://b8c80025f7b0:8888/lab?token=55f102e88b5db8585415c0adc878957c871ba819b3b9c83a
        http://127.0.0.1:8888/lab?token=55f102e88b5db8585415c0adc878957c871ba819b3b9c83a
```

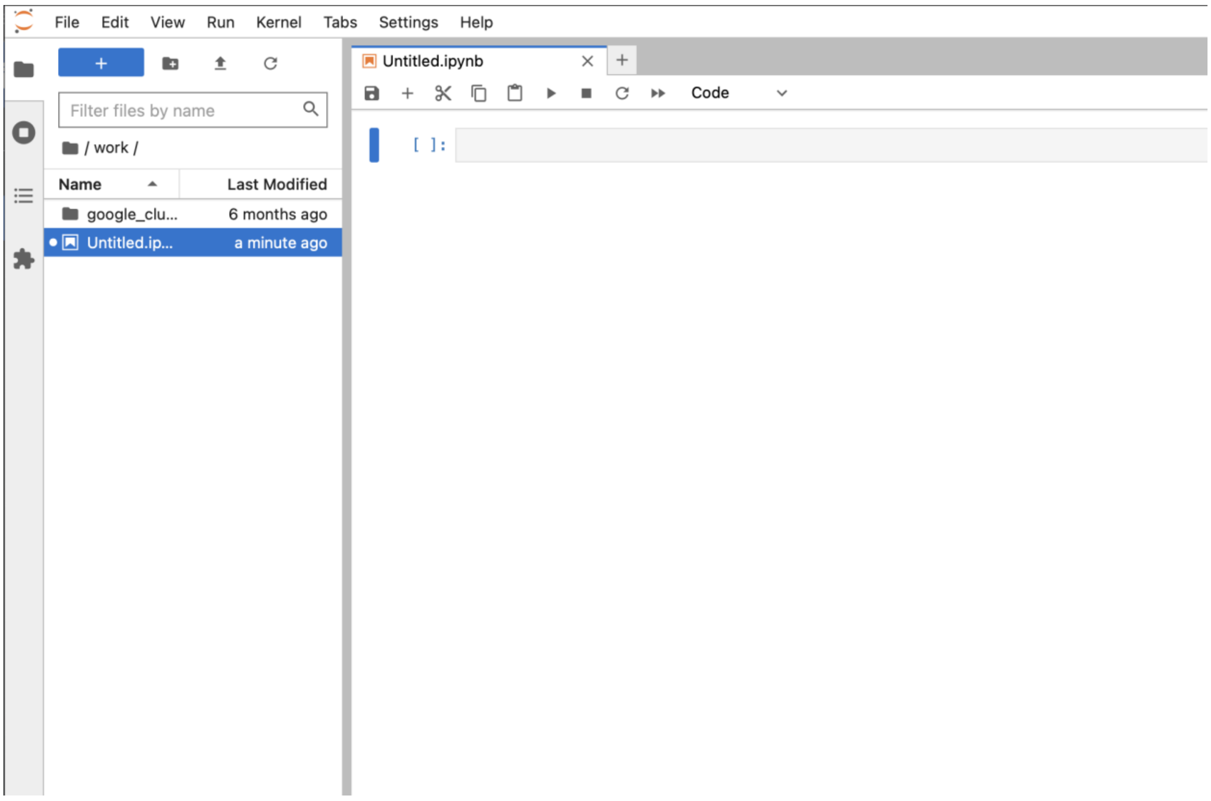
Jupyter interface

On the left side we can see the "work" folder which is associated with our local folder. Whatever we write in this work folder will also be visible in the folder on the host machine. If we want to copy files that we can access from Python, then they can be copied directly to the folder on the host machine.

Next we can create iPython files, which contain cells that can be executed one at a time. To create a file it would be good to be in the work folder, we can double click on it. Then we can click on Python3 on the right side of the images.



In the next image we can see an iPython notebook. We have different buttons for Save, Cut, Copy, Paste. Then follow the run buttons:
- to run a cell,
- to stop the execution of a cell
- to run all cells

To save the file we can use CTRL+S or the Save button.
To execute a cell we can use SHIFT+ENTER

File Edit View Run Kernel Tabs Settings Help
Filter files by name
/ work /
Name
Last Modified
google_clu...
6 months ago
Untitled.ip...
a minute ago
Untitled.ipynb
Code
[ ]: