

## Systems & Toolchains for AI Engineers: Mac M1/M2 Installation Guide

**Disclaimer:** this guide aims to help you get a high-level idea of required software installations. The provided steps in this guide may get outdated over time and/or may not match your exact operating system file hierarchy. Please use this document as a guide and not a strict document to follow.

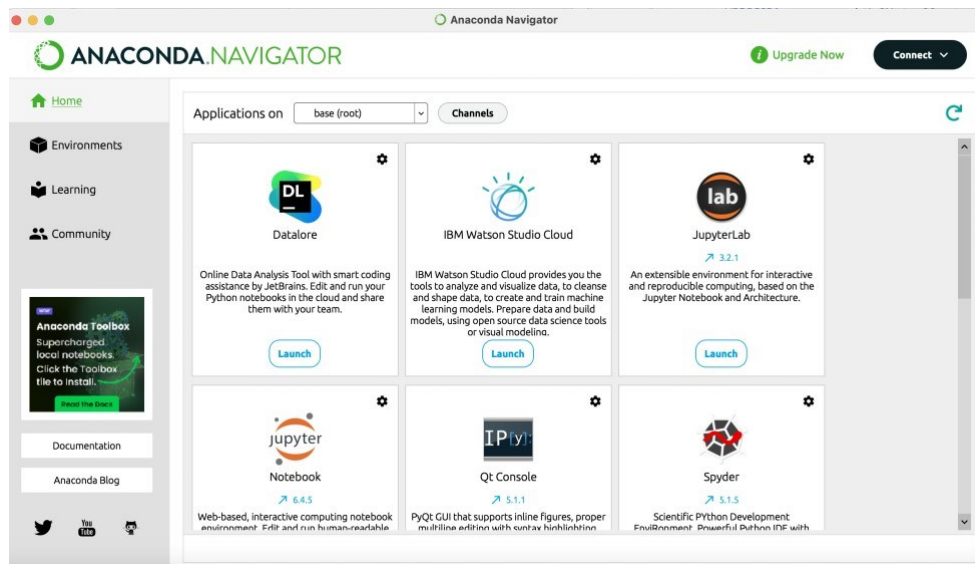
### Overview

The following applications and frameworks need to be installed for this class:

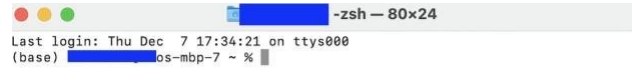
1. Anaconda Python 3.x
2. Jupyter
3. PostgreSQL DB
4. PgAdmin4
5. Apache Spark
6. TensorFlow
7. PyTorch
8. Docker

### Anaconda Python 3.x

1. Install the Anaconda Desktop App, *called Anaconda Navigator*, [here](#). This is a step-bystep guide on how to install Anaconda on Mac.
2. Once it has been installed, the interface should look like this:

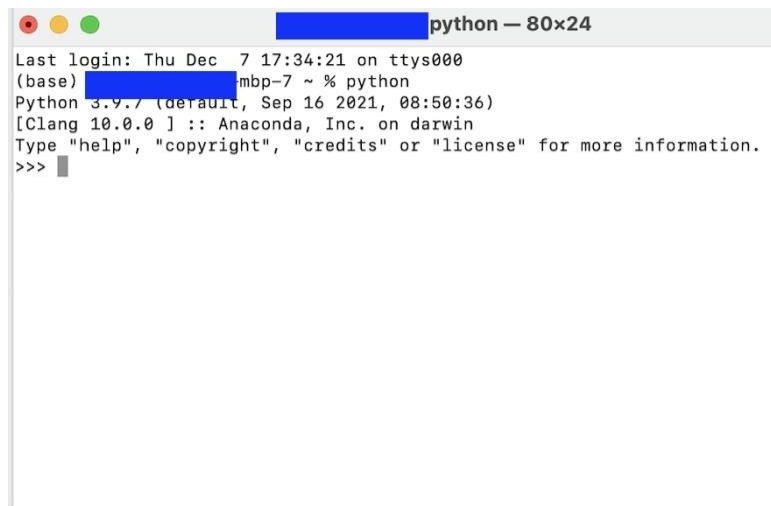


3. Verify the install on Terminal by opening Terminal and ensuring the command line starts with “(base)” as below. This means you are in the *base* Anaconda environment. More “virtual environments” can be created. Read more [here](#).



```
-zsh — 80x24
Last login: Thu Dec  7 17:34:21 on ttys000
(base) [redacted]os-mbp-7 ~ %
```

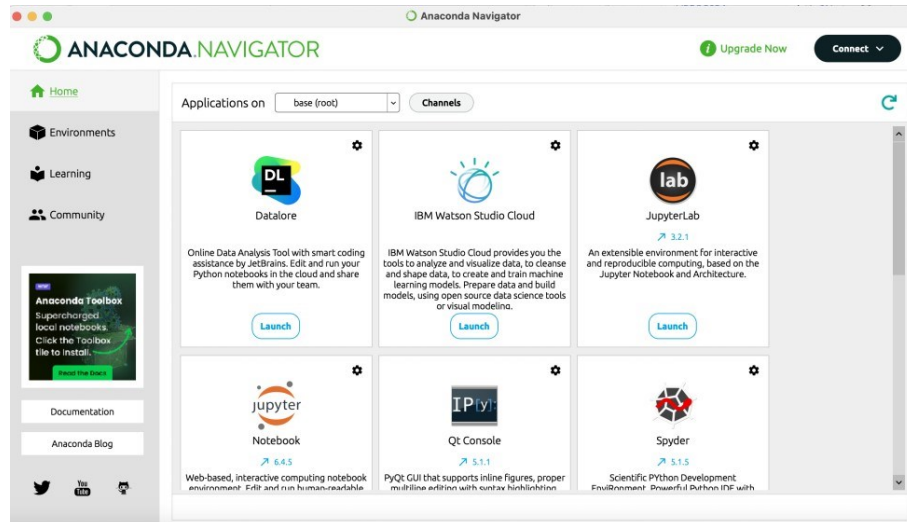
4. Python should be installed when you installed Anaconda Navigator. Type *python* in Terminal to confirm. The output should be similar below. Ensure the version is 3.x.



```
python — 80x24
Last login: Thu Dec  7 17:34:21 on ttys000
(base) [redacted]mbp-7 ~ % python
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## Jupyter Notebook

1. When Anaconda was installed, Jupyter Notebook should have also been installed. This can be verified in two ways:
  - a. See if there is a Jupyter Notebook option on the Navigator as below. Press “Launch”.

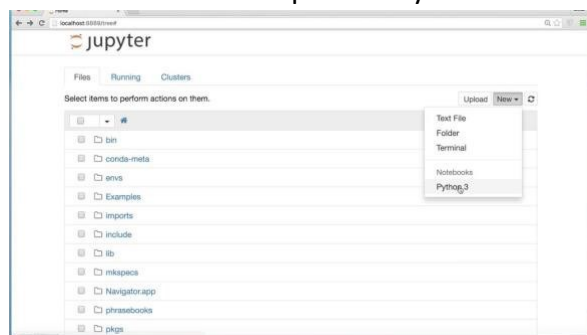


- b. Go to Terminal, and type `jupyter notebook`. This should launch a notebook server as below.

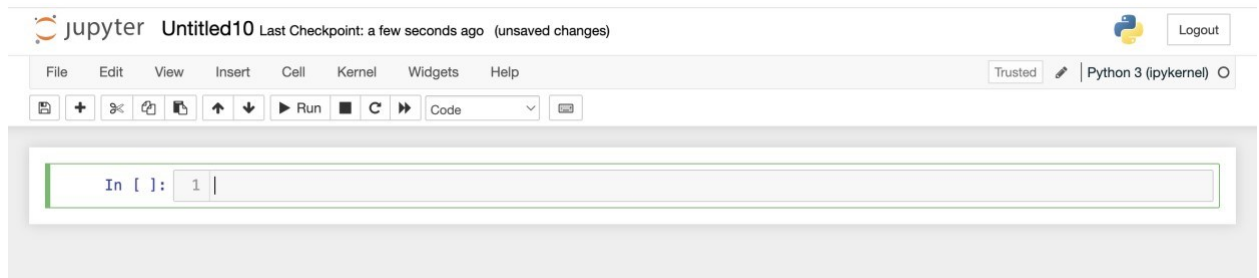
```
(base) [redacted] jupyter notebook
[I 2023-12-08 16:11:45.762 LabApp] JupyterLab extension loaded from /opt/anaconda3/lib/python3.9/site-packages/jupyterlab
[I 2023-12-08 16:11:45.762 LabApp] JupyterLab application directory is /opt/anaconda3/share/jupyter/lab
[I 16:11:45.766 NotebookApp] Serving notebooks from local directory: [redacted]
[I 16:11:45.766 NotebookApp] Jupyter Notebook 6.4.5 is running at:
[I 16:11:45.766 NotebookApp] http://localhost:8888/?token=734eddd9a3bb7db25c33cbd235e89a4be6daf85e8202dad9
[I 16:11:45.766 NotebookApp] or http://127.0.0.1:8888/?token=734eddd9a3bb7db25c33cbd235e89a4be6daf85e8202dad9
[I 16:11:45.766 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 16:11:45.777 NotebookApp]

To access the notebook, open this file in a browser:
file:///redacted/jupyter/runtime/nbserver-38809-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=734eddd9a3bb7db25c33cbd235e89a4be6daf85e8202dad9
or http://127.0.0.1:8888/?token=734eddd9a3bb7db25c33cbd235e89a4be6daf85e8202dad9
```

2. In both cases, a notebook should have opened in your browser as below.



3. Create a new notebook, by clicking “New” and choosing “Python 3”. A notebook should open as below.



## PostgreSQL DB

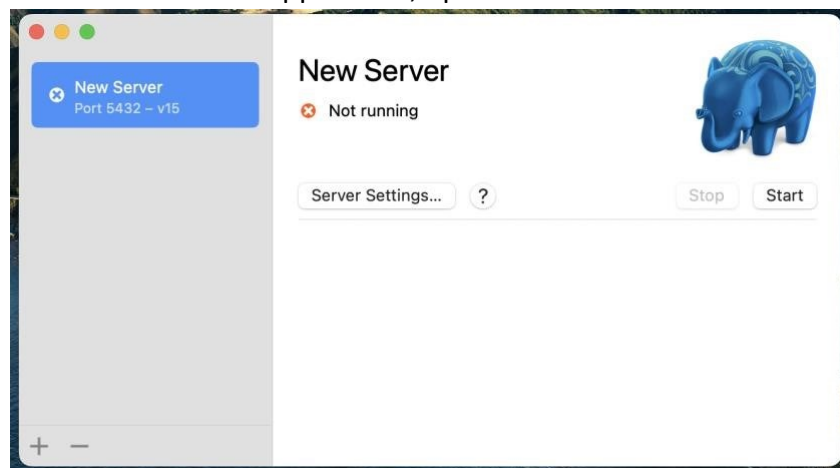
1. To download Postgres, go [here](#) and download the latest release. This will download the Postgres desktop application.

### Latest Release

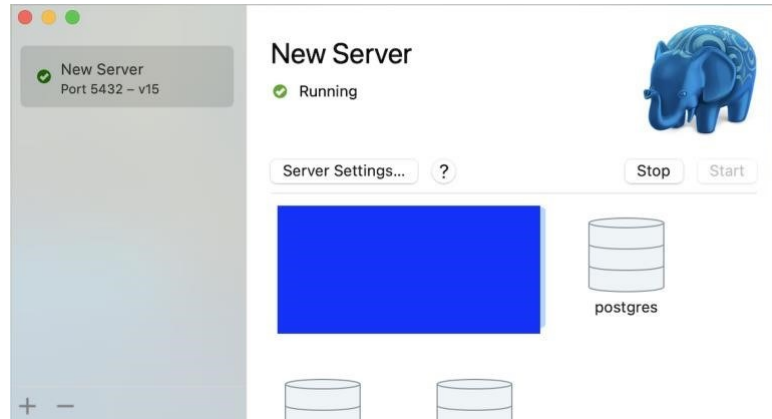
If you're new to Postgres, this is the file you should download. It includes everything you need to get started with PostgreSQL and PostGIS.



2. Once you have installed the application, open it. It should look like this.



3. Click on "Start", to start the server.



4. Double click on any of the servers to connect to it. A Terminal window should launch and look like this.

```
-p5432 postgres — 80x24
Last login: Fri Dec 8 16:17:03 on ttys001
"/Applications/Postgres.app/Contents/Versions/15/bin/psql" -p5432 "postgres"
(base) [redacted] ~ % "/Applications/Postgres.app/Contents/Versions/15/
bin/psql" -p5432 "postgres"
psql (15.4)
Type "help" for help.

postgres=#
```

5. Configure your \$PATH using the instructions [here](#).
6. After you have started the server through the app, you can directly go to Terminal and type *psql*. This should connect to the server too.

```
~ -- jupyter-notebook * python [redacted] psql -- 80x24
X ...book * python ~ -- zsh ~ -- psql ~ -- zsh
Last login: Fri Dec 8 16:38:46 on ttys003
(base) [redacted] s-mbp-7 ~ % psql
psql (15.4)
Type "help" for help.

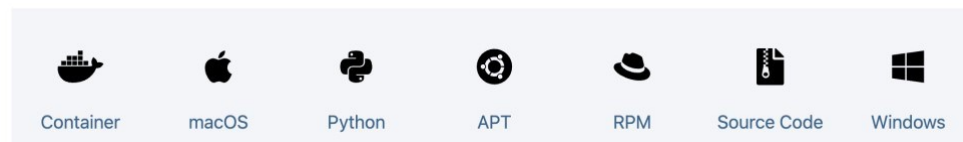
[redacted]=#
```

## PgAdmin4

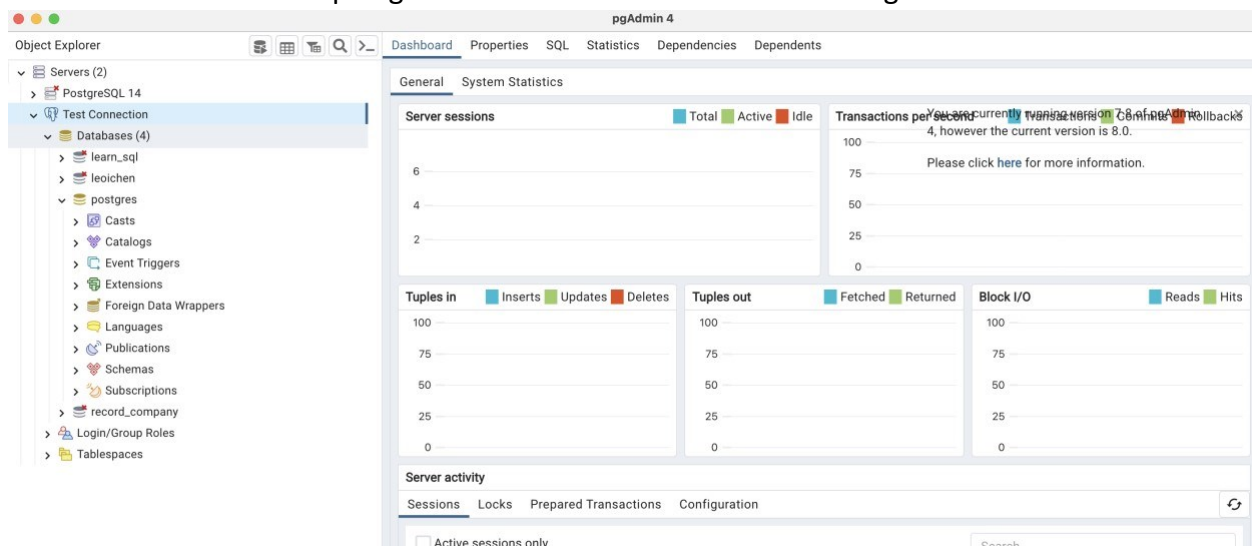
1. Go to [here](#) and download the Mac version for PgAdmin4.  
(Note: arm64 refers to the apple chip while x86\_64 refers to the intel chip)

### pgAdmin 4

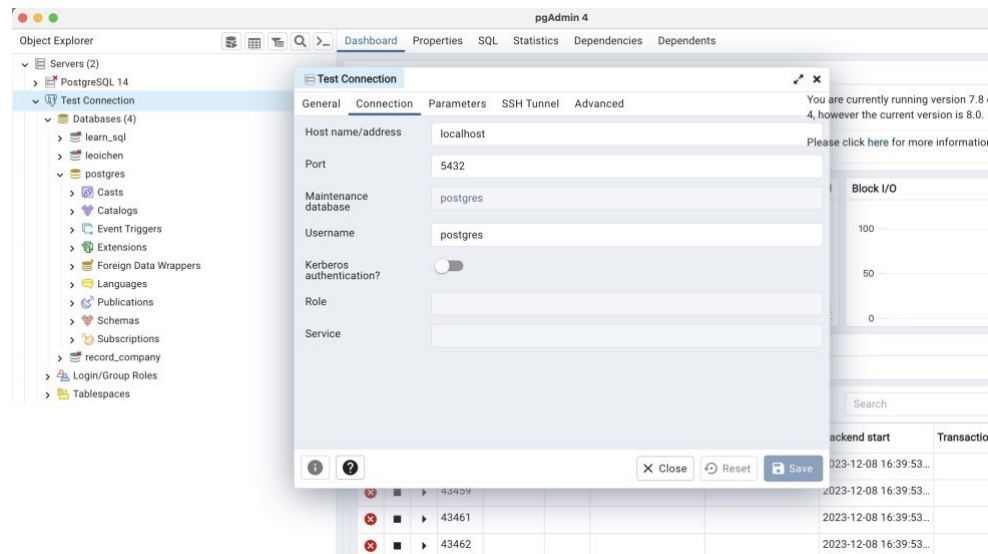
pgAdmin 4 is a complete rewrite of pgAdmin, built using Python and Javascript/jQuery. A desktop runtime written in NW.js allows it to run standalone for individual users, or the web application code may be deployed directly on a web server for use by one or more users through their web browser. The software has the look and feels of a desktop application whatever the runtime environment is, and vastly improves on pgAdmin III with updated user interface elements, multi-user/web deployment options, dashboards, and a more modern design.



2. Open the app once it has been downloaded.
3. Click the add new server button and create a server with the configuration shown in step-6. Make sure that Postgres is running.
4. Go to the left toolbar and press the downward arrows for “Test Connection”, “Databases” and “postgres”. Your left toolbar should be configured as below.



5. Right click on “Test Connection”, and press “Properties”.
6. Once the “Properties” window opens, click on the “Connection” tab to see the connection details.



## Apache Spark

Follow the steps in [here](#).

**NOTE:** Before you start installing Spark, create an Anaconda virtual environment. It is NOT recommended to install this on your base environment. If you install on your base environment and you mess up, it will be hard to return to your initial starting point. With a virtual environment, you can always just delete it and start over.

So, **CREATE A VIRTUAL ENVIRONMENT FIRST.**

To create a virtual environment, go to Terminal and type: `conda create -n env_name python=3.7`.

The `env_name` can be whatever you want. I have chosen `python=3.7` because this is the version that has worked for me. However, I think 3.8 and 3.9 should work too.

After the environment has been created, activate it with the command `conda activate env_name`.

Then, the key steps are:

1. Install Homebrew.
  - You *may* need to add brew to your ~/.zprofile using these commands:
    - i echo 'eval "\$(/opt/homebrew/bin/brew shellenv)'" >> ~/.zprofile
    - ii source ~/.zprofile
    - iii eval "\$(/opt/homebrew/bin/brew shellenv)"
2. Install Java.
3. Install Scala.
4. Install Python (although this should have been done when you created the virtual environment).
5. Install Spark.
6. Verify Spark – Scala.
7. Verify Spark – Python

```
Koushikthota@Koushiks-Laptop ~ % pyspark
Python 3.10.8 (main, Oct 21 2022, 22:22:30) [Clang 14.0.0 (clang-1400.0.29.202)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
22/11/21 17:25:24 WARN Utils: Your hostname, Koushiks-Laptop.local resolves to a loopback address: 127.0.0.1; using 192.168.0.103 instead (on interface en0)
22/11/21 17:25:24 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/11/21 17:25:25 WARN NativeCodeLoader: Unable to load native-heapson library for your platform... using builtin-java classes where applicable
22/11/21 17:25:25 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
22/11/21 17:25:25 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
22/11/21 17:25:25 WARN Utils: Service 'SparkUI' could not bind on port 4042. Attempting port 4043.
Welcome to

      ____
     / ___/
    /  /_  /
   / ___/ /
  /  /_  /
 /___/_/

version 3.3.1

Using Python version 3.10.8 (main, Oct 21 2022:22:22:30)
Spark context Web UI available at http://192.168.0.103:4043
Spark context available as 'sc' (master = local[*], app id = local-1669831725713).
SparkSession available as 'spark'.
>>>
```

8. Once you get PySpark working, try the test `spark.ipynb` provided in class.

### A few other things:

The below environment variables need to be set in your `~/.bashrc` file. To access your `.bashrc` file, enter in Terminal `nano ~/.bashrc` or `vim ~/.bashrc` depending on the Terminal editor you prefer. Make sure these paths *\*exist\** on your machine or find the equivalent ones.

Also, note that the command to find your JAVE HOME is: `/usr/libexec/java_home`

Also, the command to find your python3 is: **which python3**

#java

```
export JAVA_HOME=/Library/java/JavaVirtualMachines/adoptopenjdk-8.jdk/contents/Home/
```

```
export JRE_HOME=/Library/java/JavaVirtualMachines/openjdk-13.jdk/contents/Home/jre/
```

```
#spark export SPARK_HOME=/usr/local/Cellar/apache-
```

```
spark/2.4.4/libexec export PATH=/usr/local/Cellar/apache-
```

```
spark/2.4.4/bin:$PATH
```

#pyspark

```
export PYSPARK_PYTHON=/usr/bin/python3 # or your path to python
```

```
export PYSPARK_DRIVER_PYTHON=jupyter
```

```
export PYSARK DRIVER PYTHON OPTS='notebook --no-browser --port=8889'
```



For java, JAVA\_HOME should be the path of your java installation above. JRE\_HOME should not be necessary. But this could depend on the computer.

For spark, SPARK\_HOME should be the installation path. To find spark installation directory, enter in Terminal `echo 'sc.getConf.get("spark.home")' | spark-shell`. PATH should be like SPARK\_HOME, but instead of the libexec folder, it is the bin folder.

For PYSPARK\_PYTHON, this is just the path of your python installation.  
PYSPARK\_DRIVER\_PYTHON and PYSPARK\_DRIVER\_PYTHON\_OPTS should be exactly as above.

**ALSO:** After you update the ~/.bashrc, make sure you source it by typing in Terminal `source ~/.bashrc`. This will rerun the ~/.bashrc file, making your changes go into effect.

Normally, when you open a new Terminal instance, the ~/.bashrc file is run, so you don't need to source everytime. But, if there is an error, just source it again. You may need to source it every time you run PySpark.

## TensorFlow

To install TensorFlow, follow the guide [here](#).

**NOTE:** I would use the same virtual environment as Spark so you can use them together in the same notebook. If you are afraid the TensorFlow installation will get messed up, you can test the installation on a new virtual environment and come back to install it again on the Spark environment once you are sure you can install TensorFlow successfully.

The key steps are:

1. Install Xcode command line tools.
2. Install Miniforge.
3. Install TensorFlow-MacOS.
4. Install base TensorFlow.
5. Install metal plugin.

At this point, it should be installed. Test it on a Jupyter notebook, by typing `import tensorflow as tf`.

## PyTorch

To install PyTorch, follow the guide [here](#).

**NOTE:** Again, I would install it on the same environment as Spark.

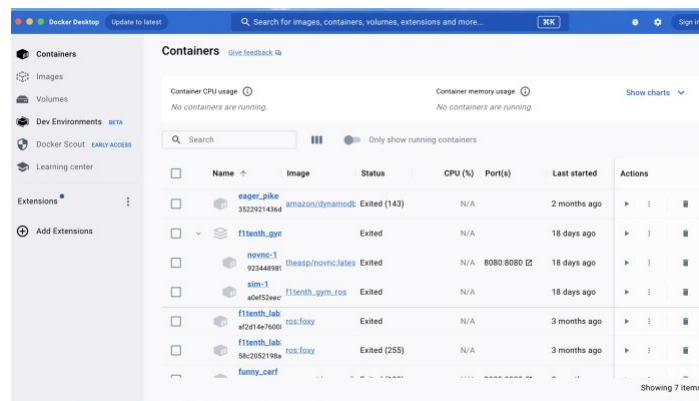
The key steps are:

1. Install Xcode
2. Install PyTorch Packages

At this point, it should be installed. Test it on a Jupyter notebook by typing *import torch*.

## Docker

1. To install Docker, go [here](#) and choose the version for Mac Silicon.
2. Once installed, open the application. It should look like this.



3. Test it Terminal by typing *docker --version*. The version should appear as

```
...ook ▶ python    ~ — -zsh    ...    ~ — -zsh    ~ — -zsh
Last login: Fri Dec 8 17:05:40 on ttys005
(base) [redacted]@leos-mbp-7 ~ % docker --version
Docker version 24.0.5, build ced0996
(base) [redacted]@leos-mbp-7 ~ %
```

## FAQ

**Q:** Do you need to run the Postgres app every time to launch the server?

**A:** No, but that is the easiest way. To launch the server through the command line, read more [here](#).

**Q:** What is the `/.bashrc` file?

**A:** This file is a configuration file for your Terminal session. It essentially contains some commands or scripts that are executed every time you start a bash session. This can help to avoid you typing in the same commands every single time you start a new bash session.

**Q:** Is there a way to quickly switch between Anaconda virtual environments when on a single notebook?

**A:** Yes. Virtual environments can be added to Jupyter notebooks. See the guide [here](#). You can then open a Jupyter notebook in any environment and easily switch to it without having to launch that environment and creating a new notebook there.

**Q:** Is there a difference between M1 and M2 in terms of the above installation?

**A:** All the guides I have provided should work for both M1 and M2. I know they work for M1 since I have a M1 Mac. I have not tested this for M2. However, even if this doesn't work, I am sure you can easily find a guide for M2, or even M3 in the future.

**Q:** What if everything has been attempted (i.e., this guide, any online guide, instructor assistance), and you still can't get everything to work?

**A:** In this case, Google Colab can be used. PyTorch and TensorFlow are built in, so installation is not needed. Just directly import with *import torch* and *import tensorflow as tf*.

For PySpark, installation can be done with the commands below:

```
!pip install pyspark  
!pip install -U -q PyDrive  
!apt install openjdk-8-jdk-headless -qq import  
os  
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
```