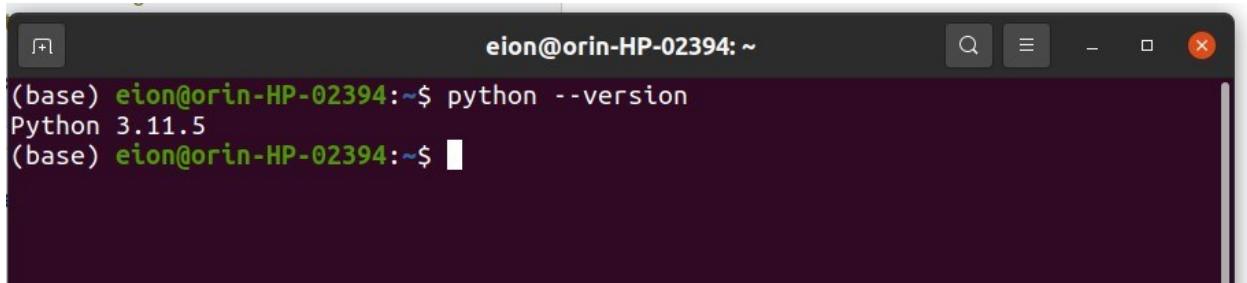


Systems & Toolchains for AI Engineers: Linux Ubuntu Installation Guide

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.

- Anaconda

- Follow the official guide <https://docs.anaconda.com/free/anaconda/install/linux/>
- Python

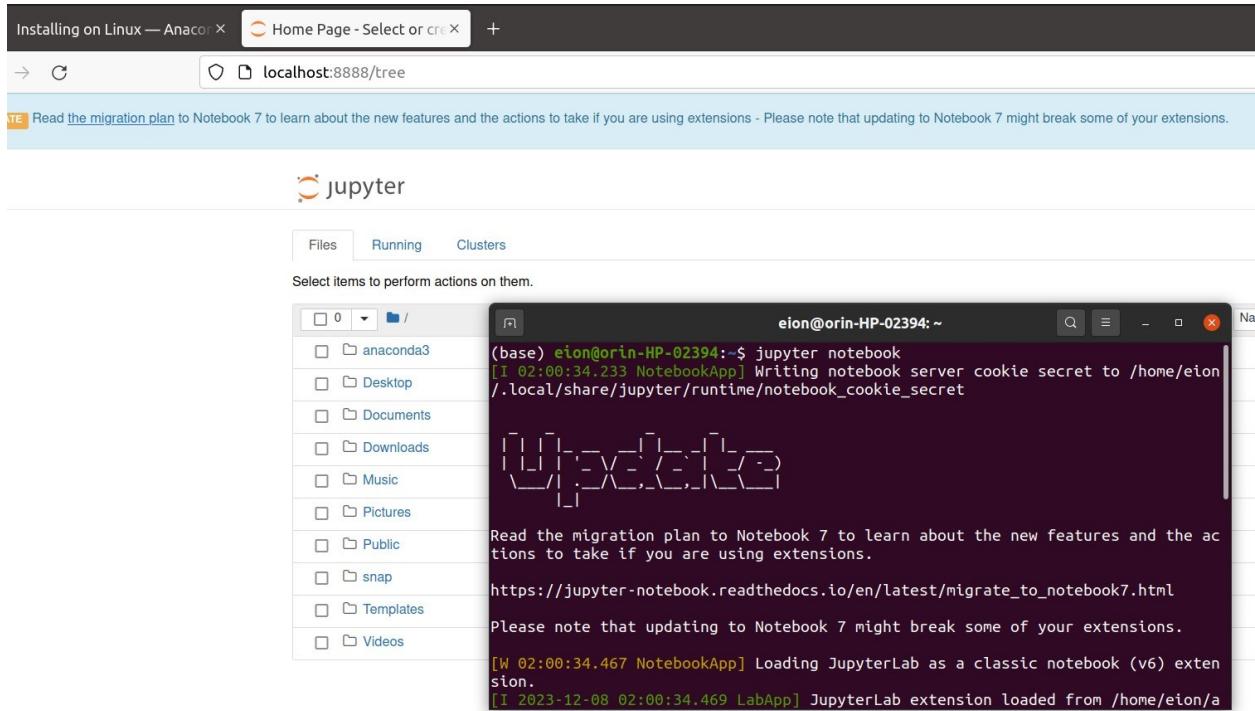


A screenshot of a terminal window titled "eion@orin-HP-02394: ~". The window shows the command "python --version" being run, with the output "Python 3.11.5". The terminal has a dark background with light-colored text and standard window controls at the top.

Then run

```
Unset
conda update -n base conda -y
conda clean --all --yes
conda install pip -y
conda install ipykernel -y
```

- Jupyter Notebook



FAQ:

1. Q: I installed anaconda, but I am not in the base environment and 'conda activate' does not work?
A: You may have told conda not to modify your shell scripts, try running the following in terminal

Unset

```
eval "$( /home/$USER/anaconda3/bin/conda shell.bash hook )"
conda init
```

Now restart your terminal and you should be in the conda base environment.

Note: If you wish to not have conda activate the base environment by default when launching a new terminal:

Unset

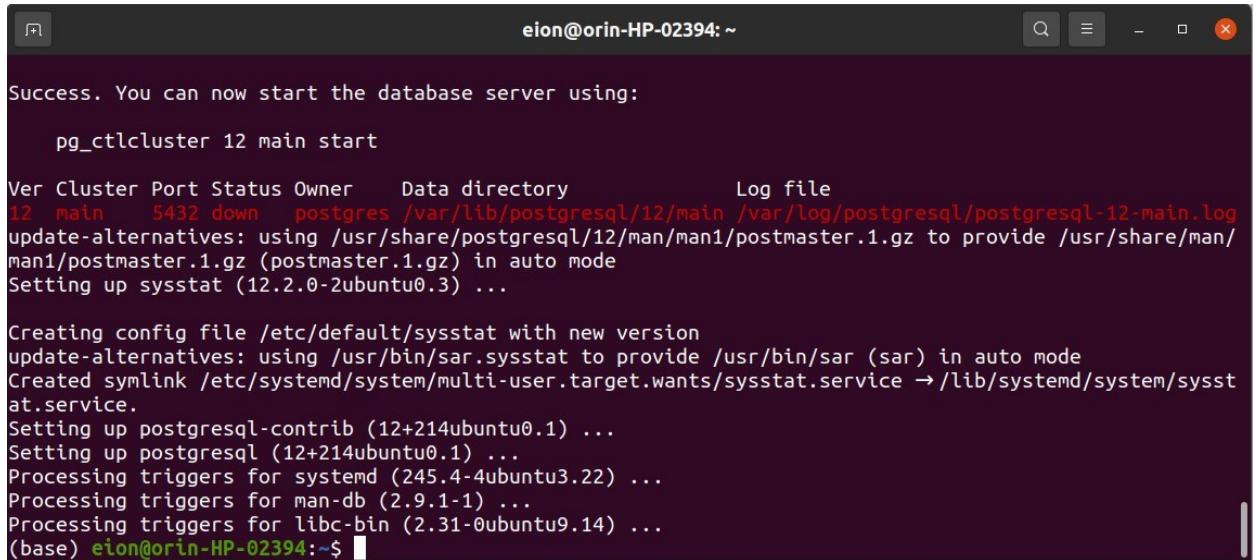
```
conda config --set auto_activate_base false
```

- Postgres and Pgadmin4 (on Ubuntu)

- PostgreSQL <https://www.postgresql.org/>
 - PgAdmin4 <https://www.pgadmin.org/download/>
-
- Install PostgreSQL + PgAdmin4 Guide
<https://tecatadmin.net/how-to-install-postgresql-in-ubuntu-20-04/>
 - PostgreSQL installation is the first 3 steps.
 - Note this is not the only guide and particular PostgreSQL versions can be specified when running the install line by specifying postgresql-12, postgresql-14, etc.
As additional confirmation in Step 1. of the guide, after you have added the PPA to your system and run the following lines in terminal:

```
sudo apt update
sudo apt-get install postgresql postgresql-contrib
```

You should see something like this:



The screenshot shows a terminal window titled "eion@orin-HP-02394: ~". The output of the command "sudo apt update" and "sudo apt-get install postgresql postgresql-contrib" is displayed. The output includes messages about success, starting the database server using pg_ctlcluster, and setting up various PostgreSQL components like data directories, log files, and system services. It also shows the creation of a config file for sysstat and the processing of triggers for systemd, man-db, and libc-bin.

```
Success. You can now start the database server using:
pg_ctlcluster 12 main start

Ver Cluster Port Status Owner      Data directory          Log file
12  main      5432 down    postgres /var/lib/postgresql/12/main /var/log/postgresql/postgresql-12-main.log
update-alternatives: using /usr/share/postgresql/12/man/man1/postmaster.1.gz to provide /usr/share/man/man1/postmaster.1.gz (postmaster.1.gz) in auto mode
Setting up sysstat (12.2.0-2ubuntu0.3) ...

Creating config file /etc/default/sysstat with new version
update-alternatives: using /usr/bin/sar.sysstat to provide /usr/bin/sar (sar) in auto mode
Created symlink /etc/systemd/system/multi-user.target.wants/sysstat.service → /lib/systemd/system/sysstat.service.

Setting up postgresql-contrib (12+214ubuntu0.1) ...
Setting up postgresql (12+214ubuntu0.1) ...
Processing triggers for systemd (245.4-4ubuntu3.22) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.14) ...
(base) eion@orin-HP-02394:~$
```

In Step 3, when setting the password, it is done from the Postgres system account and should look like the following

```
postgres@orin-HP-02394:~$ psql -c "ALTER USER postgres WITH PASSWORD 'bigdata';"
ALTER ROLE
postgres@orin-HP-02394:~$
```

Note that you are no longer at postgres=# or postgres-# .

To check that it worked, try logging back into postgres user with the new password.

```
(base) eion@orin-HP-02394:~$ su - postgres  
Password:  
postgres@orin-HP-02394:~$
```

PgAdmin4 Installation is Step 4. Since we would like to use the PgAdmin4 gui, be sure to use either pgadmin4 for both the web and desktop versions or just pgadmin4-desktop

```
sudo apt update  
sudo apt install pgadmin4 # or pgadmin4-desktop
```

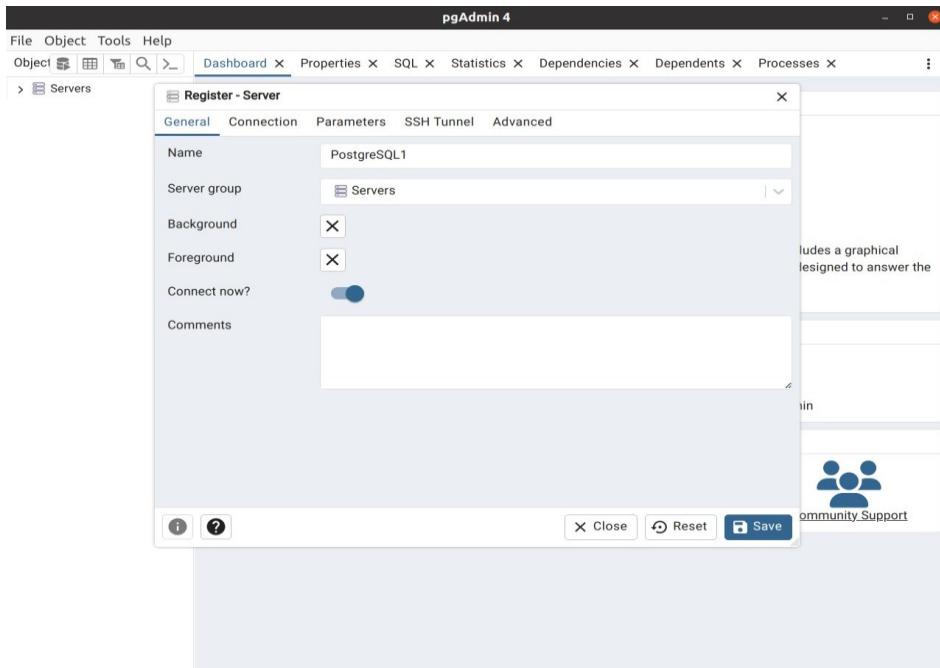
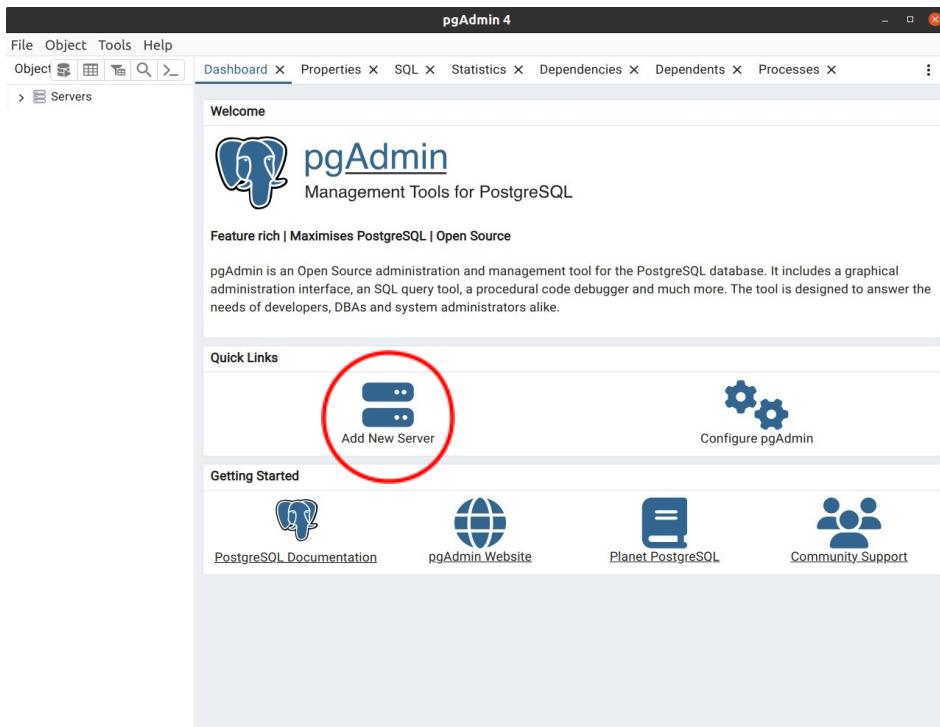
- If the previous steps to install postgres and pgadmin4 didn't work for you, you may try the following:
 1. Open terminal
 2. Run the following:
`sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'`
 3. Check if everything on your Ubuntu is up to date, you can run the following commands:
`sudo apt update`
`sudo apt upgrade`
 4. Install PostgreSQL: `sudo apt install postgresql`
 5. PostgreSQL creates a user called postgres by default. You can access this by using the following command `sudo -i -u postgres psql`
 6. Change your password by running this command `ALTER USER postgres WITH PASSWORD 'your_password';`
 7. You can exit postgres by running the commands:
`\q`
`exit`
 8. Install pgadmin4 by running the commands in the terminal:
`curl -fsS https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo gpg --dearmor -o /usr/share/keyrings/packages-pgadmin-org.gpg`

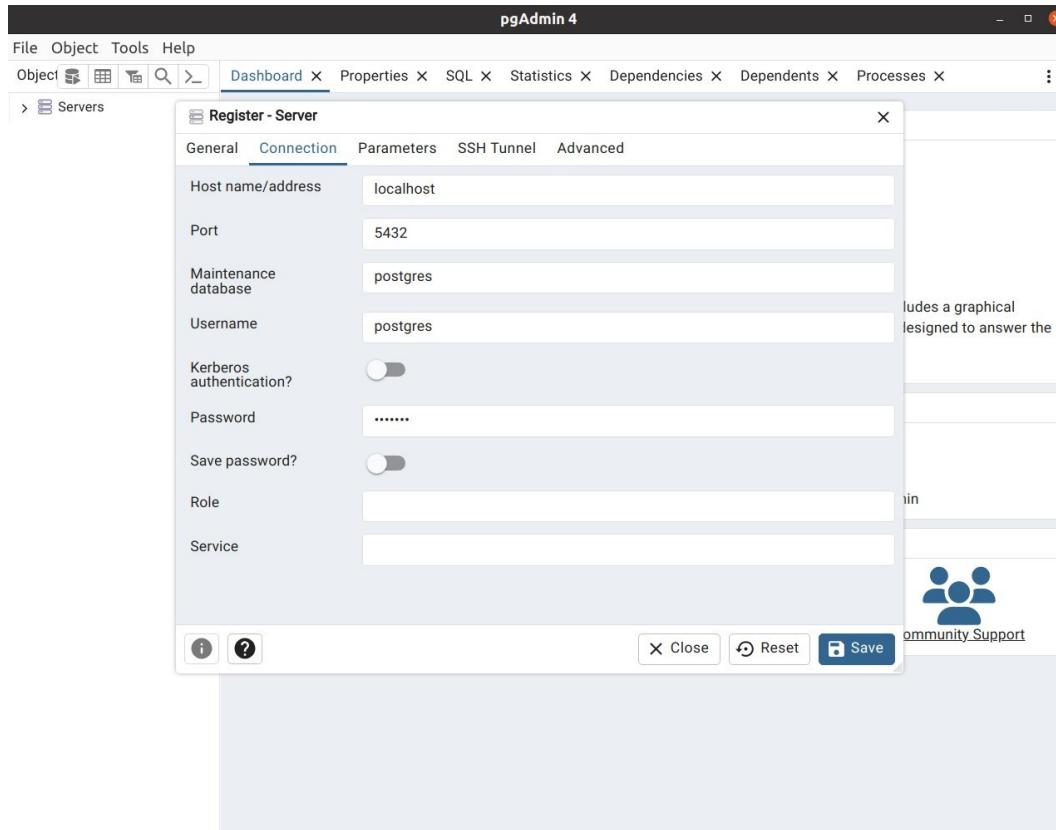
`sudo sh -c 'echo "[signed-by=/usr/share/keyrings/packages-pgadmin-org.gpg] https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/$(lsb_release -cs) pgadmin4 main" > /etc/apt/sources.list.d/pgadmin4.list && apt update'`

`sudo apt install pgadmin4-desktop`

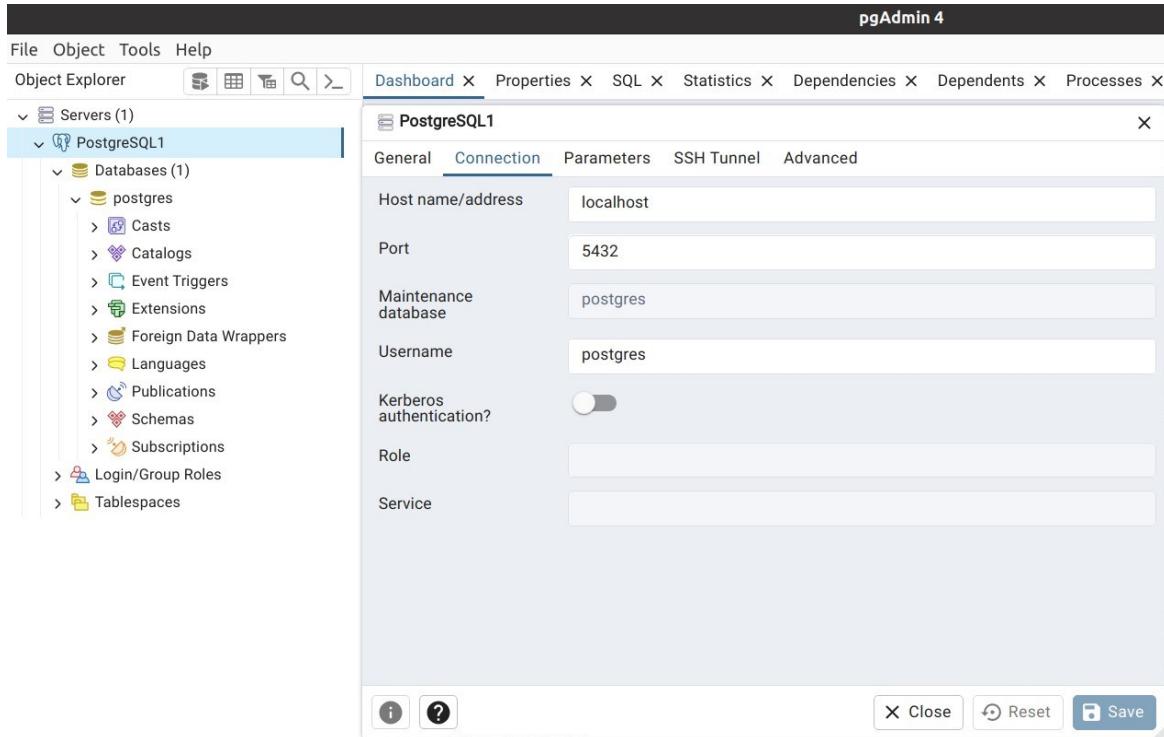
Now, let's make pgadmin4 work.

Now open the PgAdmin4 desktop application, Add New Server, update the Connection tab to connect to your PostgreSQL Database server and save. Follow the following screenshots.





Then right-click on your server, in my case PostgreSQL1, and view properties should look like this. Make sure you test the connection as well.



- Apache Spark <https://spark.apache.org/>

- For a detailed guide on installing Apache Spark and PySpark on Ubuntu 20.04, I recommend the following videos (Note this guide is using Python 3.7 for Apache Spark 2.4.4, the steps will be similar with other Python versions, however you will need to reference the apache docs for compatibility with your desired version/environment):
 - Apache Spark Docs [Latest]: <https://spark.apache.org/docs/latest/>
 - Remove latest from the url path to see all available docs
 - Part 1a <https://www.youtube.com/watch?v=7tDOUrI7Aoc>
 - Part 1b <https://www.youtube.com/watch?v=snZvQcl2HfQ>
 - For Python 3.8+ PySpark can also be installed using PyPi or Conda (see https://spark.apache.org/docs/latest/api/python/getting_started/install.html)
 - However, Java (openjdk) must be installed first and findspark after.
 - Since Conda was likely installed in step 0. above, a quick guide for PySpark installation in Conda with verification can be found here:
<https://medium.com/@divya.chandana/easy-install-pyspark-in-anacondaed427b3492f>

From your base Conda environment in terminal,

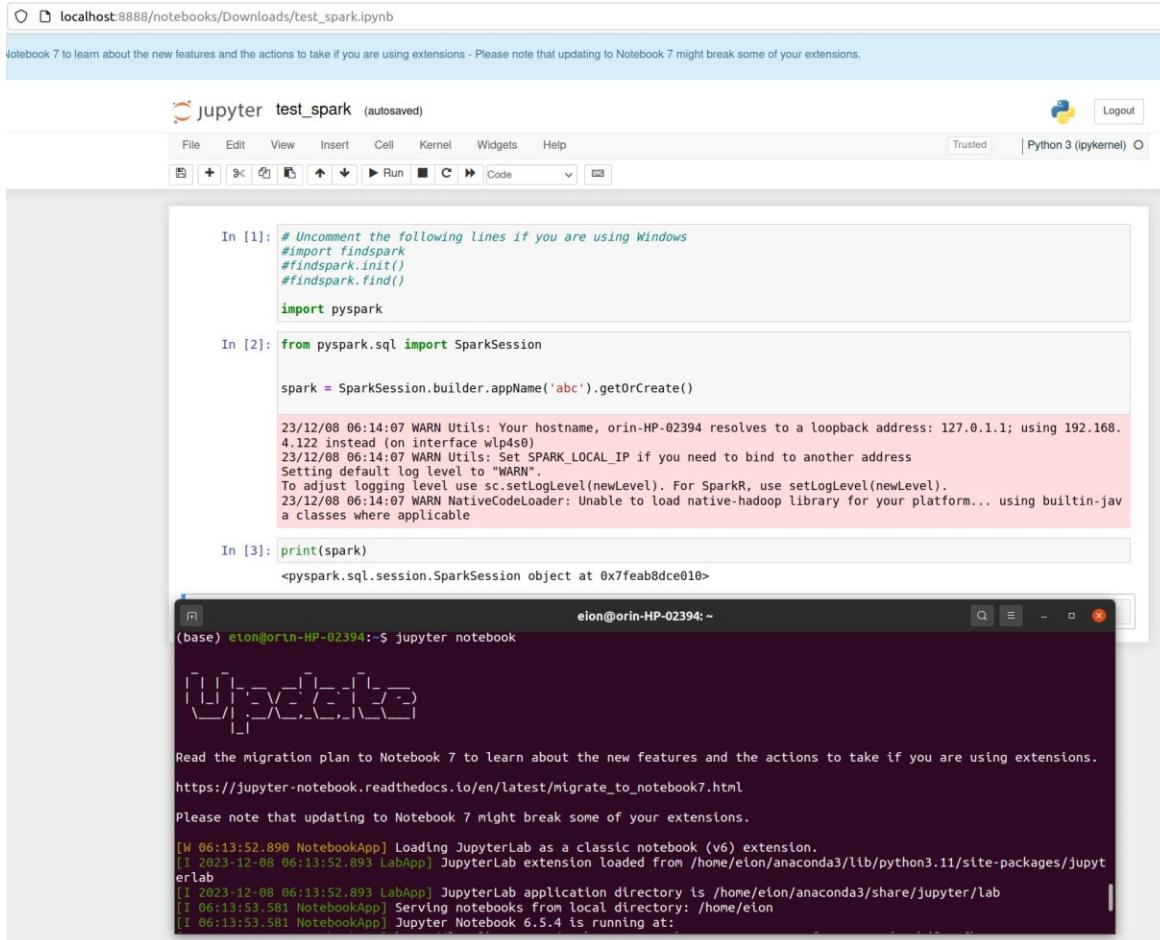
```
Unset
# From (base) Conda environment
conda install openjdk
conda install pyspark
conda install -c conda-forge findspark
```

Running ‘`pyspark`’ you should now see that you can start a `SparkSession`:

```
(base) eion@orin-HP-02394:~$ pyspark
Python 3.11.5 (main, Sep 11 2023, 13:54:46) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
23/12/08 06:07:56 WARN Utils: Your hostname, orin-HP-02394 resolves to a loopback address: 127.0.1.1; using 192.168.4.122 instead (on interface wlp4s0)
23/12/08 06:07:56 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).
23/12/08 06:07:57 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
    / \   / \ / \ / \
   /_ \ /_ \/_ \/_ \/_ \
  /_ / .\_ \_/_ / / \_ \_ \
 /_ /_ /_ /_ /_ /_ /_ /_ /_ \
version 3.4.1

Using Python version 3.11.5 (main, Sep 11 2023 13:54:46)
Spark context Web UI available at http://192.168.4.122:4040
Spark context available as 'sc' (master = local[*], app id = local-1702033678227).
SparkSession available as 'spark'.
>>> 
```

Once PySpark has been successfully installed, launch jupyter notebook and run the test spark.ipynb. You should be able to import the pyspark module and build a SparkSession.



- PyTorch & TensorFlow

When installing PyTorch and Tensorflow, if you intend on using the gpu versions, you have several options to consider. If you don't intend on using them together you may opt for installing them in individual conda/virtual environments/docker containers to avoid conflicting CUDA and cuDNN requirements. Alternatively, you could install PyTorch and TensorFlow which have the same CUDA and cuDNN requirements, by either installing both versions you want and using the appropriate version of CUDA and cuDNN for each library by specifying the correct environment variables, or you could run both with a CUDA version of the minimum requirement like installing TensorFlow with Cuda 11.2 and PyTorch for 11.6, but running entirely on 11.2, but this is not intended and can cause problems. For now I will demonstrate installing Pytorch and TensorFlow in separate conda environments. AMD GPU and CPU versions are also available for both.

Python, CUDA/cuDNN (Nvidia), ROCm (AMD) Compatibility:

- TensorFlow
 - (Nvidia)
https://www.tensorflow.org/install/source#tested_build_configurations
 - (ROCM + ROCm install)
https://github.com/ROCMSoftwarePlatform/tensorflow-upstream/blob/dev/lop-upstream/rocm_docs/tensorflow-rocm-release.md
- PyTorch
<https://github.com/pytorch/pytorch/blob/main/RELEASE.md>

Both installations can be performed with pip, conda, or built from source. pip/pip3 is often preferred.

CPU/GPU Installation Documentation:

- TensorFlow <https://www.tensorflow.org/install>
- https://www.tensorflow.org/install/pip#step-by-step_instructions
- PyTorch <https://pytorch.org/get-started/locally/>
 - <https://pytorch.org/get-started/previous-versions/>
 - CUDA <https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html>
 - Version Specific Documentation & Downloads
<https://developer.nvidia.com/cuda-toolkit-archive>
 - Select your desired package documentation and go to section 2. Perform Pre-installation Actions (Highly Recommended)
- cuDNN <https://docs.nvidia.com/cudnn/index.html>
 - Version Specific Documentation & Downloads
<https://developer.nvidia.com/rdp/cudnn-archive>

- CPU Version Installations

In the simplest case TensorFlow installation will look as follow:

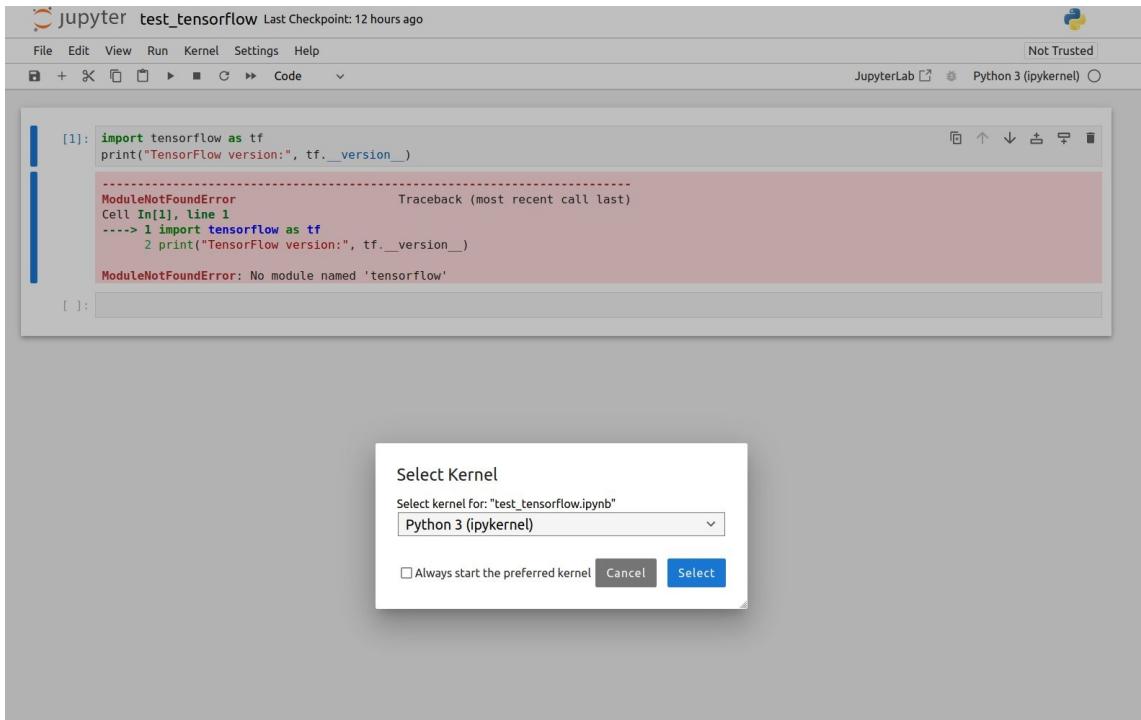
```
(base) eion@orin-HP-02394:~$ conda create -n tf_env python=3.11
Channels:
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

```
(tf_env) eion@orin-HP-02394:~$ pip install tensorflow
```

```
Installing collected packages: libclang, flatbuffers, wrapt, urllib3, typing-extensions, i
ow-io-gcs-filesystem, tensorflow-estimator, tensorboard-data-server, six, pyasn1, protobuf
lib, numpy, MarkupSafe, markdown, keras, idna, grpcio, gast, charset-normalizer, certifi,
y, werkzeug, rsa, requests, pyasn1-modules, opt-einsum, ml-dtypes, h5py, google-pasta, as
oauthlib, google-auth, google-auth-oauthlib, tensorboard, tensorflow
Successfully installed MarkupSafe-2.1.3 absl-py-2.0.0 astunparse-1.6.3 cachetools-5.3.2 c
charset-normalizer-3.3.2 flatbuffers-23.5.26 gast-0.5.4 google-auth-2.25.1 google-auth-oau
pasta-0.2.0 grpcio-1.60.0 h5py-3.10.0 idna-3.6 keras-2.15.0 libclang-16.0.6 markdown-3.5.
umpy-1.26.2 oauthlib-3.2.2 opt-einsum-3.3.0 packaging-23.2 protobuf-4.23.4 pyasn1-0.5.1 p
requests-2.31.0 requests-oauthlib-1.3.1 rsa-4.9 six-1.16.0 tensorboard-2.15.1 tensorboard
tensorflow-2.15.0.post1 tensorflow-estimator-2.15.0 tensorflow-io-gcs-filesystem-0.34.0 t
ing-extensions-4.8.0 urllib3-2.1.0 werkzeug-3.0.1 wrapt-1.14.1
```

You can now run the following to use this conda environment in your jupyter notebook,

```
Unset
conda install -c anaconda ipykernel
python -m ipykernel install --user --name=tf_env
```



Then you can launch jupyter notebook and select the ipykernel on the top right of the notebook and select the conda env associated with your tensorflow installation:

Jupyter test_tensorflow Last Checkpoint: 12 hours ago

File Edit View Run Kernel Settings Help Trusted JupyterLab tf_env

```
[1]: import tensorflow as tf
print("TensorFlow version:", tf.__version__)

2023-12-08 14:11:01.099480: I external/local_tsl/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2023-12-08 14:11:01.119757: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2023-12-08 14:11:01.119783: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2023-12-08 14:11:01.120361: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2023-12-08 14:11:01.123846: I external/local_tsl/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2023-12-08 14:11:01.124282: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-12-08 14:11:01.703945: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
TensorFlow version: 2.15.0
```

eion@orin-HP-02394:~

```
Installed kernelspec tf_env in /home/eion/.local/share/jupyter/kernels/tf_env
(base) eion@orin-HP-02394:~$ jupyter notebook
[I 2023-12-08 13:51:31.479 ServerApp] Package notebook took 0.0000s to import
[I 2023-12-08 13:51:31.483 ServerApp] Package jupyter_lsp took 0.0047s to import
[W 2023-12-08 13:51:31.484 ServerApp] A '_jupyter_server_extension_points' function was not found in jupyter_lsp. Instead, a '_jupyter_server_extension_paths' function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-12-08 13:51:31.486 ServerApp] Package jupyter_server_terminals took 0.0020s to import
[I 2023-12-08 13:51:31.486 ServerApp] Package jupyterlab took 0.0000s to import
[I 2023-12-08 13:51:31.504 ServerApp] Package notebook_shim took 0.0000s to import
[W 2023-12-08 13:51:31.504 ServerApp] A '_jupyter_server_extension_points' function was not found in notebook_shim. Instead, a '_jupyter_server_extension_paths' function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-12-08 13:51:31.909 ServerApp] Package io.jupyter_server_extension took 0.4046s to import
[I 2023-12-08 13:51:31.909 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2023-12-08 13:51:31.911 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2023-12-08 13:51:31.913 ServerApp] jupyterlab | extension was successfully linked.
[I 2023-12-08 13:51:31.915 ServerApp] notebook | extension was successfully linked.
[I 2023-12-08 13:51:32.055 ServerApp] notebook_shim | extension was successfully linked.
[I 2023-12-08 13:51:32.055 ServerApp] panel.io.jupyter_server_extension | extension was successfully linked.
[I 2023-12-08 13:51:32.062 ServerApp] notebook_shim | extension was successfully loaded.
[I 2023-12-08 13:51:32.063 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2023-12-08 13:51:32.064 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2023-12-08 13:51:32.065 LabApp] JupyterLab extension loaded from /home/eion/anaconda3/lib/python3.11/site-packages/jupyterlab
[I 2023-12-08 13:51:32.065 LabApp] JupyterLab application directory is /home/eion/anaconda3/share/jupyter/lab
```

Likewise PyTorch can simply be installed in the following manner,

```
(base) eion@orin-HP-02394:~$ conda create -n torch_env python=3.11
Channels:
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

```
(torch_env) eion@orin-HP-02394:~$ pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu
Installing collected packages: mpmath, urllib3, typing-extensions, sympy, pillow, numpy, networkx, MarkupSafe, idna, fsspec, filelock, charset-normalizer, certifi, requests, jinja2, torch, torchvision, torchaudio
Successfully installed MarkupSafe-2.1.3 certifi-2022.12.7 charset-normalizer-2.1.1 filelock-3.9.0 fsspec-2023.4.0 idna-3.4 jinja2-3.1.2 mpmath-1.3.0 networkx-3.0 numpy-1.24.1 pillow-9.3.0 requests-2.28.1 sympy-1.12.1 torch-2.1.1+cpu torchaudio-2.1.1+cpu torchvision-0.16.1+cpu typing-extensions-4.4.0 urllib3-1.26.13
(torch_env) eion@orin-HP-02394:~$
```

As before, the conda environment can be made accessible to the jupyter notebook,

```
Unset
conda install -c anaconda ipykernel
python -m ipykernel install --user --name=torch_env
```

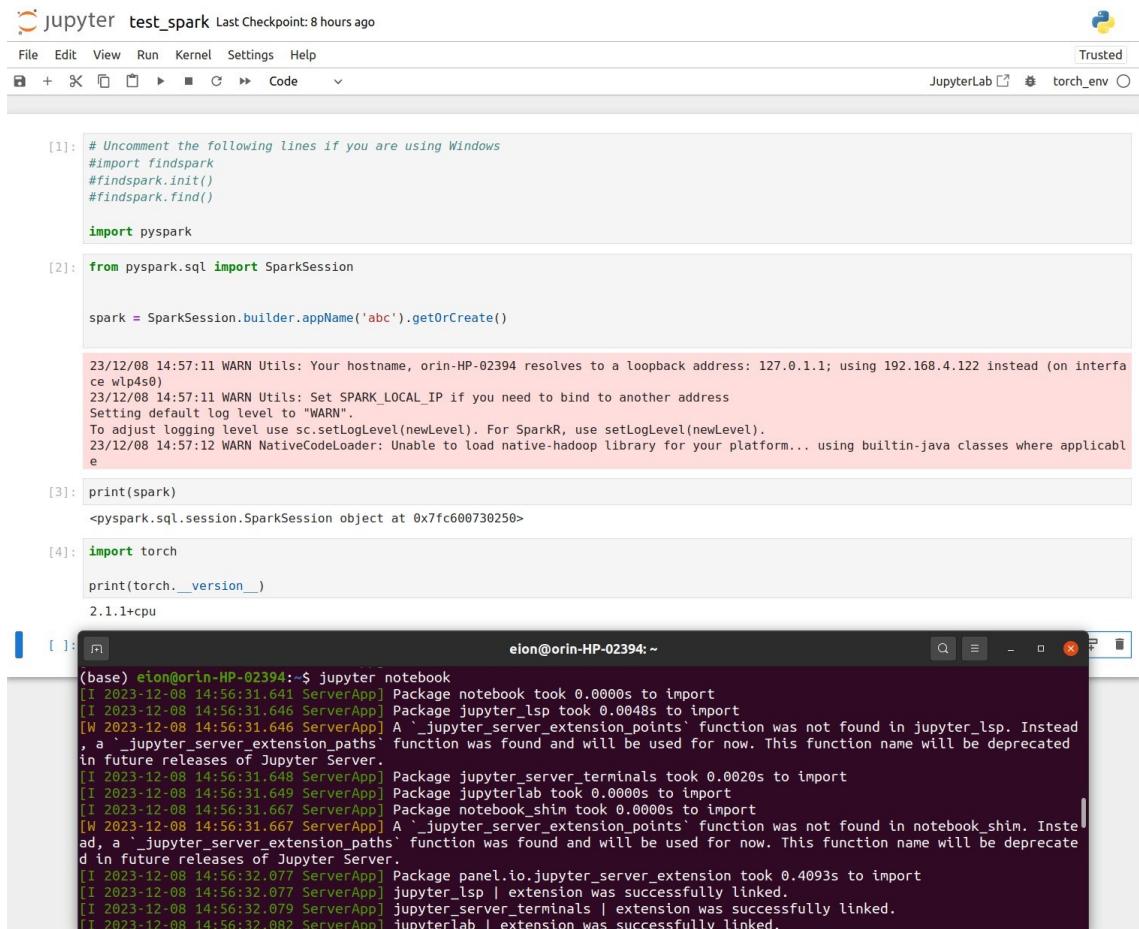
Then you can launch jupyter notebook and select the ipykernel on the top right of the notebook and select the conda env associated with your pytorch installation:



```
[2]: import torch  
print(torch.__version__)  
2.1.1+cpu
```

Note: If you require to have both PySpark and PyTorch in the same environment, you may choose to do so either in the (base) or another environment like the (torch) environment we have already made. The important thing to keep in mind here is that each environment has its own python and libraries, so if PySpark is installed in (base) and PyTorch is installed in (torch_env) they will not be able to work together.

After performing the PySpark installation in my torch_env, I can now use both PySpark and PyTorch in the same notebook:



```
[1]: # Uncomment the following lines if you are using Windows  
#import findspark  
#findspark.init()  
#findspark.find()  
  
import pyspark  
  
[2]: from pyspark.sql import SparkSession  
  
spark = SparkSession.builder.appName('abc').getOrCreate()  
  
23/12/08 14:57:11 WARN Utils: Your hostname, orin-HP-02394 resolves to a loopback address: 127.0.1.1; using 192.168.4.122 instead (on interface wlp4s0)  
23/12/08 14:57:11 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).  
23/12/08 14:57:12 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
  
[3]: print(spark)  
<pyspark.sql.session.SparkSession object at 0x7fc600730250>  
  
[4]: import torch  
  
print(torch.__version__)  
2.1.1+cpu  
  
[ ]: eion@orin-HP-02394:~$ jupyter notebook  
[I 2023-12-08 14:56:31.641 ServerApp] Package notebook took 0.000s to import  
[I 2023-12-08 14:56:31.646 ServerApp] Package jupyter_lsp took 0.0048s to import  
[W 2023-12-08 14:56:31.646 ServerApp] A `jupyter_server_extension_points` function was not found in jupyter_lsp. Instead, a `jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.  
[I 2023-12-08 14:56:31.648 ServerApp] Package jupyter_server_terminals took 0.0020s to import  
[I 2023-12-08 14:56:31.649 ServerApp] Package jupyterlab took 0.000s to import  
[I 2023-12-08 14:56:31.667 ServerApp] Package notebook_shim took 0.000s to import  
[W 2023-12-08 14:56:31.667 ServerApp] A `jupyter_server_extension_points` function was not found in notebook_shim. Instead, a `jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.  
[I 2023-12-08 14:56:32.077 ServerApp] Package panel.io.jupyter_server_extension took 0.4093s to import  
[I 2023-12-08 14:56:32.077 ServerApp] jupyter_lsp | extension was successfully linked.  
[I 2023-12-08 14:56:32.079 ServerApp] jupyter_server_terminals | extension was successfully linked.  
[I 2023-12-08 14:56:32.082 ServerApp] jupyterlab | extension was successfully linked.
```

- GPU Version Installations

Since I don't want to have to manage multiple CUDA versions I have Identified TensorFlow and PyTorch versions which share the same CUDA and cuDNN requirements:

Release Compatibility Matrix

Following is the Release Compatibility Matrix for PyTorch releases:

PyTorch version	Python	Stable CUDA	Experimental CUDA
2.2	>=3.8, <=3.11	CUDA 11.8, CUDNN 8.7.0.84	CUDA 12.1, CUDNN 8.9.2.26
2.1	>=3.8, <=3.11	CUDA 11.8, CUDNN 8.7.0.84	CUDA 12.1, CUDNN 8.9.2.26
2.0	>=3.8, <=3.11	CUDA 11.7, CUDNN 8.5.0.96	CUDA 11.8, CUDNN 8.7.0.84
1.13	>=3.7, <=3.10	CUDA 11.6, CUDNN 8.3.2.44	CUDA 11.7, CUDNN 8.5.0.96
1.12	>=3.7, <=3.10	CUDA 11.3, CUDNN 8.3.2.44	CUDA 11.6, CUDNN 8.3.2.44

GPU

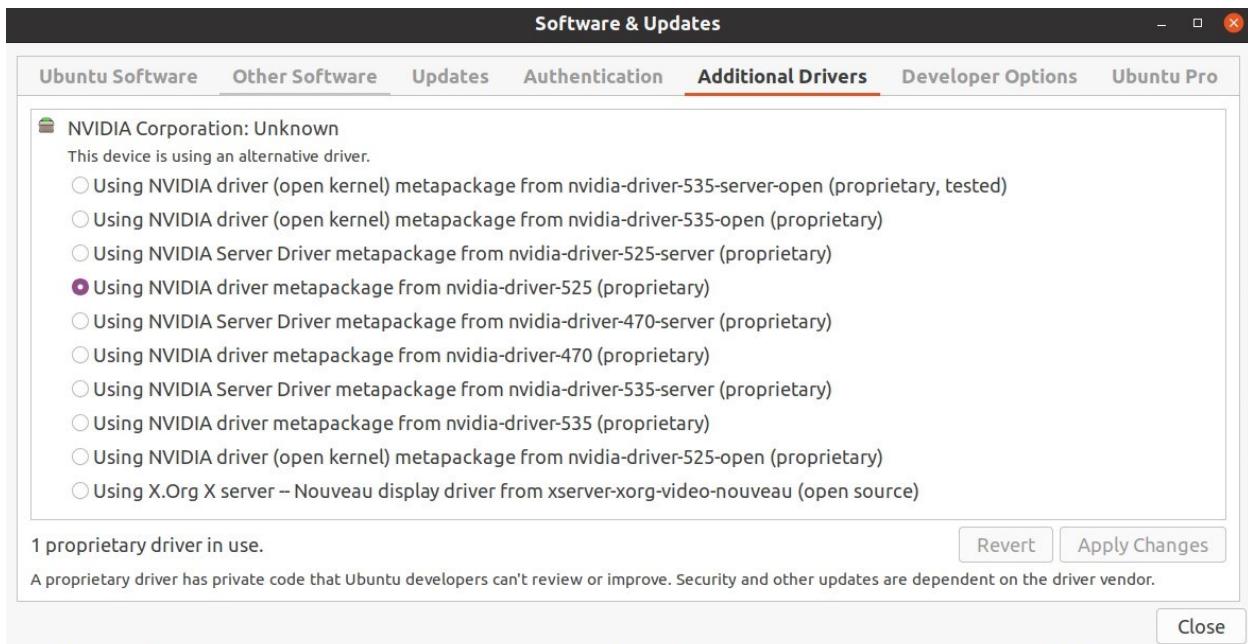
Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.15.0	3.9-3.11	Clang 16.0.0	Bazel 6.1.0	8.8	12.2
tensorflow-2.14.0	3.9-3.11	Clang 16.0.0	Bazel 6.1.0	8.7	11.8
tensorflow-2.13.0	3.8-3.11	Clang 16.0.0	Bazel 5.3.0	8.6	11.8
tensorflow-2.12.0	3.8-3.11	GCC 9.3.1	Bazel 5.3.0	8.6	11.8
tensorflow-2.11.0	3.7-3.10	GCC 9.3.1	Bazel 5.3.0	8.1	11.2
tensorflow-2.10.0	3.7-3.10	GCC 9.3.1	Bazel 5.1.1	8.1	11.2
tensorflow-2.9.0	3.7-3.10	GCC 9.3.1	Bazel 5.0.0	8.1	11.2

Once you have selected your desired framework versions you must first check whether your GPU driver is compatible with your selected framework's CUDA requirement.

Table 2: CUDA Toolkit and Minimum Required Driver Version for CUDA Minor Version Compatibility

CUDA Toolkit	Minimum Required Driver Version for CUDA Minor Version Compatibility*	
	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 12.3.x	>=525.60.13	>=527.41
CUDA 12.2.x	>=525.60.13	>=527.41
CUDA 12.1.x	>=525.60.13	>=527.41
CUDA 12.0.x	>=525.60.13	>=527.41
CUDA 11.8.x	>=450.80.02	>=452.39
CUDA 11.7.x	>=450.80.02	>=452.39
CUDA 11.6.x	>=450.80.02	>=452.39
CUDA 11.5.x	>=450.80.02	>=452.39
CUDA 11.4.x	>=450.80.02	>=452.39
CUDA 11.3.x	>=450.80.02	>=452.39
CUDA 11.2.x	>=450.80.02	>=452.39
CUDA 11.1 (11.1.0)	>=450.80.02	>=452.39
CUDA 11.0 (11.0.3)	>=450.36.06**	>=451.22**

If your Nvidia driver is not compatible, consider selecting a different framework version or updating your GPU driver. You can check Additional Drivers in Software & Updates to confirm that your driver supports the CUDA toolkit version you would like.



Now navigate to your desired release in the CUDA Toolkit archive:

<https://developer.nvidia.com/cuda-toolkit-archive>

Archived Releases

- CUDA Toolkit 12.3.0 (October 2023), [Versioned Online Documentation](#)
- CUDA Toolkit 12.2.2 (August 2023), [Versioned Online Documentation](#)
- CUDA Toolkit 12.2.1 (July 2023), [Versioned Online Documentation](#)
- CUDA Toolkit 12.2.0 (June 2023), [Versioned Online Documentation](#)
- CUDA Toolkit 12.1.1 (April 2023), [Versioned Online Documentation](#)
- CUDA Toolkit 12.1.0 (February 2023), [Versioned Online Documentation](#)
- CUDA Toolkit 12.0.1 (January 2023), [Versioned Online Documentation](#)
- CUDA Toolkit 12.0.0 (December 2022), [Versioned Online Documentation](#)
- CUDA Toolkit 11.8.0 (October 2022), [Versioned Online Documentation](#)**
- CUDA Toolkit 11.7.1 (August 2022), [Versioned Online Documentation](#)
- CUDA Toolkit 11.7.0 (May 2022), [Versioned Online Documentation](#)

By clicking your desired version, you can navigate to the correct download for your system:

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the [CUDA EULA](#).

Operating System	Linux	Windows							
Architecture	x86_64	ppc64le	arm64-sbsa	aarch64-jetson					
Distribution	CentOS	Debian	Fedora	KylinOS	OpenSUSE	RHEL	Rocky	SLES	Ubuntu
	WSL-Ubuntu								
Version	18.04	20.04	22.04						
Installer Type	deb (local)	deb (network)	runfile (local)						

Download Installer for Linux Ubuntu 20.04 x86_64

The base installer is available for download below.

> Base Installer

Installation Instructions:

```
$ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64/cuda-ubuntu2004.pin
$ sudo mv cuda-ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-pin-600
$ wget https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cuda-repo-ubuntu2004-11-8-local_11.8.0-520.61.05-1_amd64.deb
$ sudo dpkg -i cuda-repo-ubuntu2004-11-8-local_11.8.0-520.61.05-1_amd64.deb
$ sudo cp /var/cuda-repo-ubuntu2004-11-8-local/cuda-*keyring.gpg /usr/share/keyrings/
$ sudo apt-get update
$ sudo apt-get -y install cuda
```

Once installed, you will be prompted to reboot.

After rebooting add your CUDA version binaries to PATH and LD_LIBRARY_PATH in your '`~/.bashrc`'

```

unset
export PATH=/usr/local/cuda-11.8/bin${PATH:+:$PATH}
export
LD_LIBRARY_PATH=/usr/local/cuda-11.8/lib64${LD_LIBRARY_PATH:+:$LD_LIBRARY_PATH}
}}
```

Now run ‘source ~/.bashrc’ and check your CUDA version with ‘nvcc -V’ to confirm that CUDA was properly installed and matches the version you need:

```
(base) eion@orin-HP-02394:~$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

Now it is time to install our desired version of cuDNN corresponding to both the version necessary for your selected framework and compatible with your CUDA version from the cuDNN archive.

[Download cuDNN v8.7.0 \(November 28th, 2022\), for CUDA 11.x](#)

Local Installers for Windows and Linux, Ubuntu(x86_64, armsbsa)

[Local Installer for Windows \(Zip\)](#)
[Local Installer for Linux x86_64 \(Tar\)](#)
[Local Installer for Linux PPC \(Tar\)](#)
[Local Installer for Linux SBSA \(Tar\)](#)
[Local Installer for Debian 11 \(Deb\)](#)
[Local Installer for Ubuntu20.04 x86_64 \(Deb\)](#)
[Local Installer for Ubuntu22.04 x86_64 \(Deb\)](#)
[Local Installer for Ubuntu20.04 aarch64bsa \(Deb\)](#)
[Local Installer for Ubuntu22.04 aarch64bsa \(Deb\)](#)
[Local Installer for Ubuntu20.04 cross-sbsa \(Deb\)](#)
[Local Installer for Ubuntu22.04 cross-sbsa \(Deb\)](#)

Then, following section 3 of this guide <https://medium.com/@zhanwenchen/build-pytorch-from-source-with-cuda-11-8-565ab737bfc8> you should download the appropriate Deb file for your system, in my case this is Ubuntu20.04 x86_64 (Deb) .

Note: The file names seen below will differ based on your selection.

```
Unset
mkdir cudnn_install
mv cudnn-local-repo-ubuntu2004-8.7.0.84_1.0-1_amd64.deb cudnn_install
cd cudnn_install
ar -x cudnn-local-repo-ubuntu2004-8.7.0.84_1.0-1_amd64.deb
```

```
Unset
tar -xvf data.tar.xz
```

```
(base) eion@orin-HP-02394:~/cudnn_install$ ar -x cudnn-local-repo-ubuntu2004-8.7.0.84_1.0-1_amd64.deb
(base) eion@orin-HP-02394:~/cudnn_install$ ls
control.tar.gz  cudnn-local-repo-ubuntu2004-8.7.0.84_1.0-1_amd64.deb  data.tar.xz  debian-binary  _gpgbuilder
(base) eion@orin-HP-02394:~/cudnn_install$ tar -xvf data.tar.xz
./
./etc/
./etc/apt/
./etc/apt/sources.list.d/
./etc/apt/sources.list.d/cudnn-local-ubuntu2004-8.7.0.84.list
./usr/
./usr/share/
./usr/share/doc/
./usr/share/doc/cudnn-local-repo-ubuntu2004-8.7.0.84/
./usr/share/doc/cudnn-local-repo-ubuntu2004-8.7.0.84/changelog.Debian.gz
./var/
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/A3837CDF.pub
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/InRelease
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/Local.md5
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/Local.md5.gpg
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/Packages
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/Packages.gz
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/Release
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/Release.gpg
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/cudnn-local-A3837CDF-keyring.gpg
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/libcudnn8-dev_8.7.0.84-1+cuda11.8_amd64.deb
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/libcudnn8-samples_8.7.0.84-1+cuda11.8_amd64.deb
./var/cudnn-local-repo-ubuntu2004-8.7.0.84/libcudnn8 8.7.0.84-1+cuda11.8 amd64.deb
```

The following should be installed in order

```
Unset
cd var/cudnn-local-repo-ubuntu2004-8.7.0.84/
sudo dpkg -i libcudnn8_8.7.0.84-1+cuda11.8_amd64.deb
sudo dpkg -i libcudnn8-dev_8.7.0.84-1+cuda11.8_amd64.deb
sudo dpkg -i libcudnn8-samples_8.7.0.84-1+cuda11.8_amd64.deb
```

Then verify the cuDNN installation with the following:

```
Unset
cat /usr/include/x86_64-linux-gnu/cudnn_version_v8.h | grep CUDNN_MAJOR -A 2
```

```
(base) eion@orin-HP-02394:~$ cat /usr/include/x86_64-linux-gnu/cudnn_version_v8.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR 8
#define CUDNN_MINOR 7
#define CUDNN_PATCHLEVEL 0
```

Finally we can install our desired frameworks

- tensorflow 2.14

I will install TensorFlow in a new conda environment:

```
(base) eion@orin-HP-02394:~$ conda create -n tf_gpu_env python=3.11
Retrieving notices: ...working... done
Channels:
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

```
(tf_gpu_env) eion@orin-HP-02394:~$ pip install tensorflow==2.14.0
Successfully installed google-auth-oauthlib-1.0.0 keras-2.14.0 tensorboard-2.14.1
tensorflow-2.14.0 tensorflow-estimator-2.14.0
(tf_gpu_env) eion@orin-HP-02394:~$ █
```

You can now run the following to use this conda environment in your jupyter notebook,

```
Unset
conda install -c anaconda ipykernel
python -m ipykernel install --user --name=tf_gpu_env
```

Then you can launch jupyter notebook and select the ipykernel on the top right of the notebook and select the conda env associated with your tensorflow installation. In my case the environment is 'tf_gpu_env':

jupyter test_tensorflow Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab tf_gpu_env

```
[1]: import tensorflow as tf
print("TensorFlow version:", tf.__version__)

2023-12-12 00:03:15.164444: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2023-12-12 00:03:15.164472: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2023-12-12 00:03:15.164487: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2023-12-12 00:03:15.168728: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-12-12 00:03:15.770248: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
TensorFlow version: 2.14.0
```

```
[2]: from tensorflow.python.client import device_lib
device_lib.list_local_devices()

2023-12-12 00:05:53.302364: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:894] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2023-12-12 00:05:53.359767: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:894] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2023-12-12 00:05:53.360186: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:894] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2023-12-12 00:05:55.729680: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:894] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2023-12-12 00:05:55.729835: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:894] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2023-12-12 00:05:55.729944: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:894] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355
2023-12-12 00:05:55.730027: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1886] Created device /device:GPU:0 with 13156 MB memory: -> device: 0, name: NVIDIA GeForce RTX 3080 Laptop GPU, pci bus id: 0000:01:00.0, compute capability: 8.6

[2]: [name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 13602468855725817383
xla_global_id: -1,
name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 13795065856
locality {
  bus_id: 1
  links {
  }
}
incarnation: 17140210617857968207
physical_device_desc: "device: 0, name: NVIDIA GeForce RTX 3080 Laptop GPU, pci bus id: 0000:01:00.0, compute capability: 8.6"
xla_global_id: 416903419]
```

- torch 2.1

Likewise, I will install PyTorch in a new conda environment:

```
(base) eion@orin-HP-02394:~$ conda create -n torch_gpu_env python=3.11
Channels:
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

(torch_gpu_env) eion@orin-HP-02394:~$ pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu118
Successfully installed MarkupSafe-2.1.3 certifi-2022.12.7 charset-normalizer-2.1.1 fi
lelock-3.9.0 fsspec-2023.4.0 idna-3.4 jinja2-3.1.2 mpmath-1.3.0 networkx-3.0 numpy-1.
24.1 pillow-9.3.0 requests-2.28.1 sympy-1.12 torch-2.1.1+cu118 torchaudio-2.1.1+cu118
torchvision-0.16.1+cu118 triton-2.1.0 typing-extensions-4.4.0 urllib3-1.26.13
(torch_gpu_env) eion@orin-HP-02394:~$
```

As before, the conda environment can be made accessible to the jupyter notebook,

```
Unset
conda install -c anaconda ipykernel
python -m ipykernel install --user --name=torch_gpu_env
```

Then you can launch jupyter notebook and select the ipykernel on the top right of the notebook and select the conda env associated with your pytorch installation:



The screenshot shows a Jupyter Notebook interface with the following content:

```
[2]: import torch
print(torch.__version__)
2.1.1+cu118

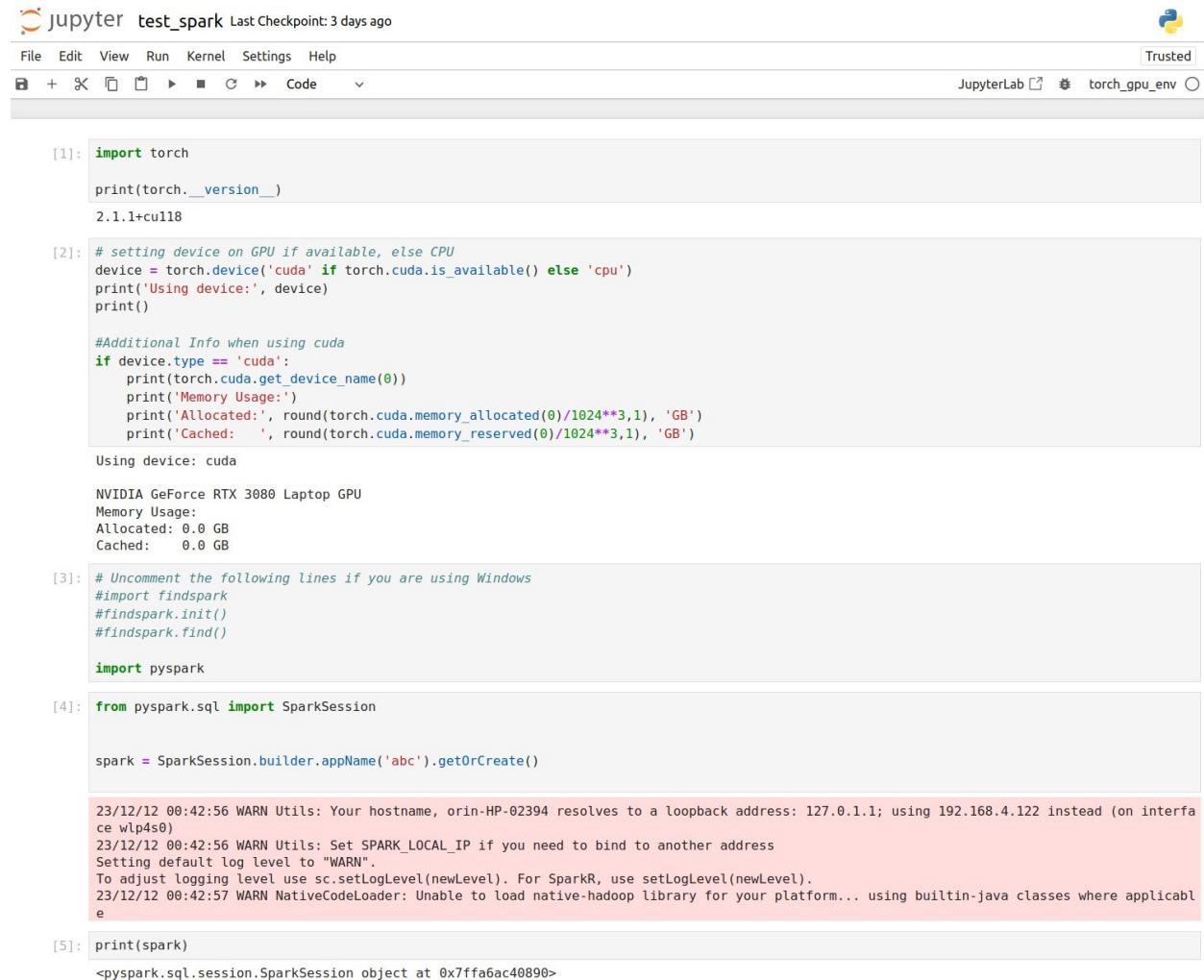
[3]: # setting device on GPU if available, else CPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Using device:', device)
print()

#Additional Info when using cuda
if device.type == 'cuda':
    print(torch.cuda.get_device_name(0))
    print('Memory Usage:')
    print('Allocated:', round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
    print('Cached:   ', round(torch.cuda.memory_reserved(0)/1024**3,1), 'GB')

Using device: cuda
NVIDIA GeForce RTX 3080 Laptop GPU
Memory Usage:
Allocated: 0.0 GB
Cached: 0.0 GB
```

Note: If you require to have both PySpark and PyTorch in the same environment, you may choose to do so either in the (base) or another environment like the (torch) environment we have already made. The important thing to keep in mind here is that each environment has its own python and libraries, so if PySpark is installed in (base) and PyTorch is installed in (torch_gpu_env) they will not be able to work together.

After performing the PySpark installation in my torch_gpu_env, I can now use both PySpark and PyTorch in the same notebook:



```
jupyter test_spark Last Checkpoint: 3 days ago
File Edit View Run Kernel Settings Help
Trusted
JupyterLab Trusted torch_gpu_env

[1]: import torch
print(torch.__version__)
2.1.1+cu118

[2]: # setting device on GPU if available, else CPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Using device:', device)
print()

#Additional Info when using cuda
if device.type == 'cuda':
    print(torch.cuda.get_device_name(0))
    print('Memory Usage:')
    print('Allocated:', round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
    print('Cached:   ', round(torch.cuda.memory_reserved(0)/1024**3,1), 'GB')

Using device: cuda
NVIDIA GeForce RTX 3080 Laptop GPU
Memory Usage:
Allocated: 0.0 GB
Cached: 0.0 GB

[3]: # Uncomment the following lines if you are using Windows
#import findspark
#findspark.init()
#findspark.find()

import pyspark

[4]: from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('abc').getOrCreate()

23/12/12 00:42:56 WARN Utils: Your hostname, orin-HP-02394 resolves to a loopback address: 127.0.1.1; using 192.168.4.122 instead (on interface wlp4s0)
23/12/12 00:42:56 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).
23/12/12 00:42:57 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

[5]: print(spark)
<pyspark.sql.session.SparkSession object at 0x7ffa6ac40890>
```

-Docker <https://docs.docker.com/>

Uninstall any conflicting packages from previous docker installations (Optional if no previous installations):

```
Unset  
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker  
containerd runc; do sudo apt-get remove $pkg; done
```

Then follow the 3 steps for Install using the apt repository

<https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository>

Once complete you should be able to check your docker version:

```
(base) eion@orin-HP-02394:~$ docker --version  
Docker version 24.0.7, build afdd53b  
(base) eion@orin-HP-02394:~$ █
```