If you want to install CUDA on your machine, and you're running Ubuntu 20.04 (Focal Fossa) OR Ubuntu 18.04, just follow these instructions, and you'll be set in 5 minutes.

If you already tried and failed to install CUDA and CuDNN use step 1 to clean up your system first. If you are trying to install a fresh installed system proceed to step 2.

1. Clean up

(a) Open a terminal window and type the following three commands to get rid of any NVIDIA/CUDA packages you may already have installed:

```
sudo rm /etc/apt/sources.list.d/cuda*
sudo apt remove --autoremove nvidia-cuda-toolkit
sudo apt remove --autoremove nvidia-*
```

(b) Purge any remaining NVIDIA configuration files and the associated dependencies that they may have been installed with.

```
sudo apt-get purge nvidia*
sudo apt-get autoremove
sudo apt-get autoclean
```

c) Remove any existing CUDA folders you may have in **/usr/local/**

This threw me for a loop the first several times I tried installing CUDA. There shouldn't be any folders with the name "**cuda**" or "**cuda-anything**" in **usr/local/** at this point!

```
sudo rm -rf /usr/local/cuda*
```

CUDA 10.1 and CuDNN 7.6.5

2. Install

(a) Setup your CUDA PPA. (Read here if you're curious about what a PPA is.)

Essentially, we're adding CUDA to our **sources.list**, which is the file that's referenced any time we use the **apt** package manager to download stuff in the terminal with a command like "`sudo apt update`"

```
sudo apt update
```

```
sudo add-apt-repository ppa:graphics-drivers
```

```
sudo apt-key adv --fetch-keys
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu180
4/x86_64/3bf863cc.pub
```

```
sudo bash -c 'echo "deb
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804
/x86_64 /" > /etc/apt/sources.list.d/cuda.list'
```

```
sudo bash -c 'echo "deb
http://developer.download.nvidia.com/compute/machine-learning/repo
s/ubuntu1804/x86_64 /" > /etc/apt/sources.list.d/cuda_learn.list'
```

(b) Install CUDA 10.1 packages, including the CuDNN library

```
sudo apt update

sudo apt install cuda-10-1

sudo apt install libcudnn7
```

3. Add CUDA to PATH

(a) After installing, we need to add CUDA to our PATH, so that the shell knows where to find CUDA. To edit our path, open up the ‘**.profile**’ file using your favorite text editor. For example, to edit using vim use the following command

```
sudo vim ~/.profile
```

(b) Finally, add these lines to the end of the file.

```
# set PATH for cuda 10.1 installation

if [ -d "/usr/local/cuda-10.1/bin/" ]; then

   export PATH=/usr/local/cuda-10.1/bin${PATH:+:${PATH}}

   export
LD_LIBRARY_PATH=/usr/local/cuda-10.1/lib64${LD_LIBRARY_PATH:+:${LD
_LIBRARY_PATH}}

fi
```

4. Reboot

(a) Reboot your computer (as you should after any driver install)

5. Final Check

Now, let’s check to make sure everything’s set up!

(a) Check NVIDIA Drivers:

```
nvidia-smi
```

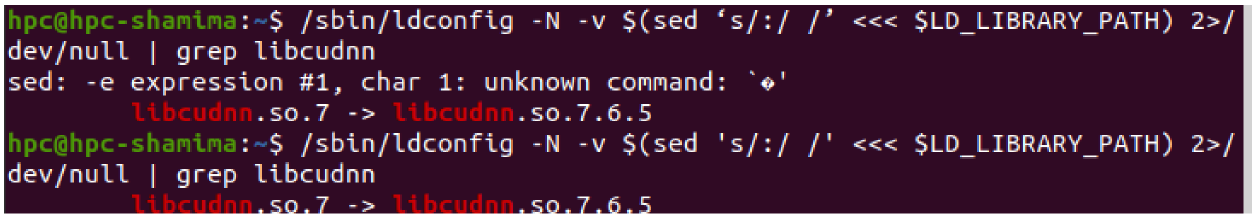
(b) Check CUDA:

```
nvcc --version
```

(c) Check CuDNN

```
/sbin/ldconfig -N -v $(sed ‘s/:/ /’ <<< $LD_LIBRARY_PATH)
2>/dev/null | grep libcudnn
```

If you get error like this replace single quote from the command



```
hpc@hpc-shamima:~$ /sbin/ldconfig -N -v $(sed ‘s/:/ /’ <<< $LD_LIBRARY_PATH) 2>/
dev/null | grep libcudnn
sed: -e expression #1, char 1: unknown command: `�'
        libcudnn.so.7 -> libcudnn.so.7.6.5
hpc@hpc-shamima:~$ /sbin/ldconfig -N -v $(sed 's/:/ /' <<< $LD_LIBRARY_PATH) 2>/
dev/null | grep libcudnn
        libcudnn.so.7 -> libcudnn.so.7.6.5
```

As long as these three checks didn’t throw you any nasty error messages, you’re all set!

Original Content: [Available here]()

CUDA 11.2 and CuDNN 8.1

Install Curl if you already haven't it in your ubuntu system. You can install curl using the following command

```
sudo apt install curl
```

1. Install Miniconda

Miniconda is the recommended approach for installing TensorFlow with GPU support. It creates a separate environment to avoid changing any installed software in your system. This is also the easiest way to install the required software especially for the GPU setup.

You can use the following command to install Miniconda. During installation, you may need to press enter and type "yes".

```
curl
https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64
.sh -o Miniconda3-latest-Linux-x86_64.sh
```

```
bash Miniconda3-latest-Linux-x86_64.sh
```

Restart your terminal. If you see your terminal is starting with (base) before username that means you have installed Miniconda successfully.

2. Create a conda environment

Create a new conda environment named tf with python version 3.9 the following command.

```
conda create --name tf python=3.9
```

Now, you can deactivate and activate it with the following commands.

```
conda activate tf
```

```
conda deactivate
```

Make sure it is activated for the rest of the installation.

4. GPU setup

You can use the following command to verify it is installed.

```
nvidia-smi
```

Then install CUDA and cuDNN with conda.

conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0

Configure the system paths. You can do it with following command everytime your start a new terminal after activating your conda environment.

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/

For your convenience it is recommended that you automate it with the following commands. The system paths will be automatically configured when you activate this conda environment.

mkdir -p $CONDA_PREFIX/etc/conda/activate.d

echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/' > $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh

5. Install TensorFlow

TensorFlow requires a recent version of pip, so upgrade your pip installation to be sure you're running the latest version.

```
pip install --upgrade pip
```

Then, install TensorFlow with pip.

```
pip install tensorflow
```

If you want to install any specific version of the tensorflow use (== tensorflow version). For example, to install tensorflow 2.6.0 use pip install tensorflow==2.6.0

6. Verify install

Verify the CPU setup:

```
python3 -c "import tensorflow as tf;
print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```

If a tensor is returned, you've installed TensorFlow successfully.

Verify the GPU setup:

```
python3 -c "import tensorflow as tf;
print(tf.config.list_physical_devices('GPU'))"
```

If a list of GPU devices is returned, you've installed TensorFlow successfully.

https://www.tensorflow.org/install/source#gpu

Some Tested GPU Builds

Tensorflow Ver	**Python version**	**Compiler**	**Build tools**	**cuDNN**	**CUDA**
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
tensorflow-2.8.0	3.7-3.10	GCC 7.3.1	Bazel 4.2.1	8.1	11.2
tensorflow-2.7.0	3.7-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.6.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.5.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1
tensorflow-2.1.0	2.7, 3.5-3.7	GCC 7.3.1	Bazel 0.27.1	7.6	10.1
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0

You can find more tested builds for GPU here

https://www.tensorflow.org/install/source#gpu