# Install GPU TensorFlow from Source on Ubuntu Server 16.04 LTS

This tutorial shows how to install GPU Tensorflow from source on Ubuntu Server 16.04.2 LTS, linux environment, with Nvidia Geforce GTX 1080 GPU.

TensorFlow supports using CUDA 8.0 & CUDNN 5.1. In order to use TensorFlow with GPU support you must have a NVIDIA graphic card with a minimum [compute capability](https://developer.nvidia.com/cuda-gpus) of 3.0.

### TensorFlow for GPU prerequisites:

The following NVIDIA hardware must be installed on your system:

* GPU card with CUDA Compute Capability 3.0 or higher. See [NVIDIA documentation](https://developer.nvidia.com/cuda-gpus)

<https://developer.nvidia.com/cuda-gpus> for a list of supported GPU cards.

The following NVIDIA software must be installed on your system:

* NVIDIA's Cuda Toolkit (>= 7.0). We recommend version 8.0. Ensure that you append the relevant Cuda pathnames to the LD\_LIBRARY\_PATH environment variable.
* The NVIDIA drivers associated with NVIDIA's Cuda Toolkit.
* cuDNN (>= v3). We recommend version 5.1. Make sure to append the appropriate path name to your LD\_LIBRARY\_PATH environment variable.

The following steps are required for installation, for more details check the official website:-

https://www.tensorflow.org/install/install\_sources

**1- Install Required Packages**

$ cd ~

$ sudo apt-get install openjdk-8-jdk git python-dev python3-dev python-numpy python3-numpy python-six python3-six build-essential python-pip python3-pip python-virtualenv swig python-wheel python3-wheel libcurl3-dev libcupti-dev

**2- Update and Install NVIDIA Drivers:**

First, update the system by the following code

$ sudo apt update

To install new driver just open the icon (search your computer) and type additional drivers, then, select a driver and apply the changes. Finally, restart the system. You can check the latest driver version according to your GPU info from Nvidia website

http://www.nvidia.com/Download/index.aspx

To make sure that Nvidia driver is installed, search in your computer (Nvidia X server setting), then you will see the details.

Alternatively, you may also obtain Nvidia info by the following code

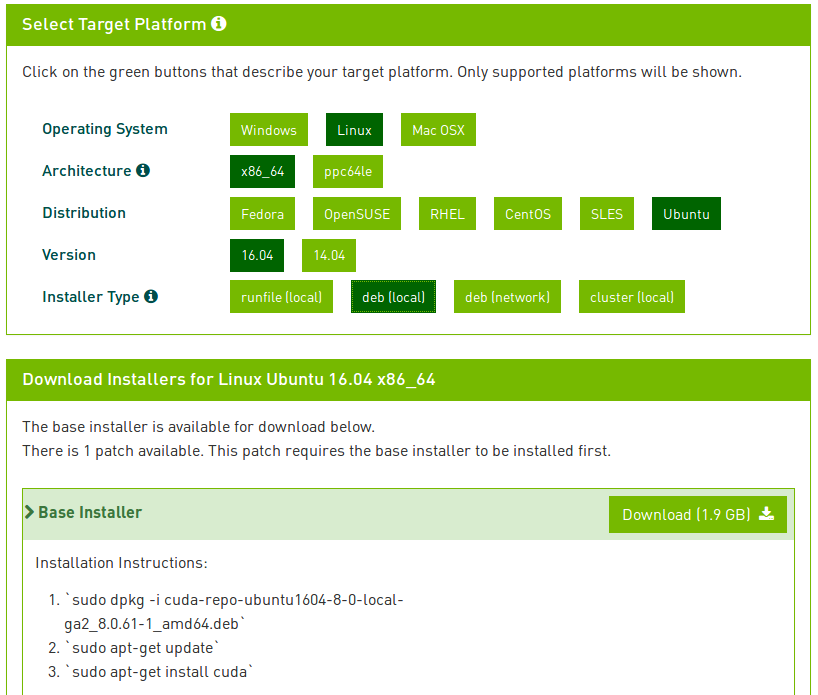
$ nvidia-smi

In my case, NVIDIA driver version 375.66

**3- Install NVIDIA CUDA Toolkit 8.0**

To install CUDA 8.0, open the Nvidia website and download CUDA based on your system, then proceed to install the CUDA file. As shown in the following figure.

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



After finish downloading, follow the installation instructions: - This will install cuda into: /usr/local/cuda-8.0

$ cd ~/Downloads # or directory to where you downloaded file

$ sudo dpkg -i cuda-repo-ubuntu1604-8-0-local-ga2\_8.0.61-1\_amd64.deb

$ sudo apt-get update

$ sudo apt-get install cuda

To make sure that CUDA was installed properly:

$ nvcc -V

4- Install NVIDIA cuDNN

Once the CUDA Toolkit is installed, open the following NVIDIA website (Note that you will be asked to register an NVIDIA developer account in order to download).

Download [cuDNN v5.1](https://developer.nvidia.com/cudnn) for Cuda 8.0 and choose [cuDNN v5.1 Library for Linux](https://developer.nvidia.com/compute/machine-learning/cudnn/secure/v5.1/prod_20161129/8.0/cudnn-8.0-linux-x64-v5.1-tgz).

<https://developer.nvidia.com/cudnn>

After downloading, go to the directory of cudnn to extract cudnn and make copy into /usr/local/cuda

$ cd ~/Downloads # or directory to where you downloaded file

$ sudo tar -xzvf cudnn-8.0-linux-x64-v5.1.tgz

$ sudo cp cuda/include/cudnn.h /usr/local/cuda/include

$ sudo cp cuda/lib64/libcudnn\* /usr/local/cuda/lib64

$ sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn\*

Then update your bash file:

$ gedit ~/.bashrc

This will open your [bash file](http://askubuntu.com/questions/540683/what-is-a-bashrc-file-and-what-does-it-do) in a text editor which you will scroll to the bottom and add these lines:

export LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:/usr/local/cuda/lib64:/usr/local/cuda/extras/CUPTI/lib64"

export CUDA\_HOME=/usr/local/cuda

Once you save and close the text file you can return to your original terminal and type this command to reload your .bashrc file:

$ source ~/.bashrc

To check cudnn version that you just installed:

$ cat /usr/local/cuda/include/cudnn.h | grep CUDNN\_MAJOR -A 2

5- Install Bazel

Installation instructions also on [Bazel](http://www.bazel.io/docs/install.html) website, <https://docs.bazel.build/versions/master/install.html>

$ echo "deb [arch=amd64] http://storage.googleapis.com/bazel-apt stable jdk1.8" | sudo tee /etc/apt/sources.list.d/bazel.list

$ curl https://storage.googleapis.com/bazel-apt/doc/apt-key.pub.gpg | sudo apt-key add -

$ sudo apt-get update

$ sudo apt-get install bazel

$ sudo apt-get upgrade bazel

To check Bazel version just type in the terminal

$ bazel version

**6- Install TensorFlow**

$ cd ~

$ git clone <https://github.com/tensorflow/tensorflow>

$ git checkout # To check where Branch is the desired branch

For example, to work with the r1.0 release instead of the master release, issue the following command:

$ git checkout r1.0

Next, you must prepare your environment for [Linux](https://www.tensorflow.org/install/install_sources#PrepareLinux) or [Mac OS](https://www.tensorflow.org/install/install_sources#PrepareMac)

## Prepare environment for Linux

**7- Configure TensorFlow Installation**

$ cd ~/tensorflow

$ ./configure

Then, you will be asked something like the following questions; the red bold words should be your answer. For questions about using python2 just press Enter to continue as a default. In my case, python2.

To build for Python 3 enter:

$ /usr/bin/python3.5

For the desired Python3 library path

$ /usr/local/lib/python3.5/dist-packages

The questions are as follow:-

Please specify the location of python. [Default is /usr/bin/python]:

Please specify optimization flags to use during compilation when bazel option "--config=opt" is specified [Default is -march=native]:

Do you wish to use jemalloc as the malloc implementation? [Y/n] **y**

jemalloc enabled

Do you wish to build TensorFlow with Google Cloud Platform support? [y/N] **n**

No Google Cloud Platform support will be enabled for TensorFlow

Do you wish to build TensorFlow with Hadoop File System support? [y/N] **n**

No Hadoop File System support will be enabled for TensorFlow

Do you wish to build TensorFlow with the XLA just-in-time compiler (experimental)? [y/N] **n**

No XLA JIT support will be enabled for TensorFlow

Found possible Python library paths:

/usr/local/lib/python2.7/dist-packages

Please input the desired Python library path to use. Default is [/usr/local/lib/python2.7/dist-packages]

Using python library path: /usr/local/lib/python2.7/dist-packages

Do you wish to build TensorFlow with OpenCL support? [y/N] **n**

No OpenCL support will be enabled for TensorFlow

Do you wish to build TensorFlow with CUDA support? [y/N] **y**

CUDA support will be enabled for TensorFlow

Please specify which gcc should be used by nvcc as the host compiler. [Default is /usr/bin/gcc]:

Please specify the Cuda SDK version you want to use, e.g. 7.0. [Leave empty to use system default]: **8.0**

Please specify the location where CUDA 8.0 toolkit is installed. Refer to README.md for more details. [Default is /usr/local/cuda]:

Please specify the cuDNN version you want to use. [Leave empty to use system default]: **5**

Please specify the location where cuDNN 5 library is installed. Refer to README.md for more details. [Default is /usr/local/cuda]:

Please specify a list of comma-separated Cuda compute capabilities you want to build with.

You can find the compute capability of your device at: https://developer.nvidia.com/cuda-gpus.

Please note that each additional compute capability significantly increases your build time and binary size.

[Default is: "3.5,5.2"]:

Setting up Cuda include

Setting up Cuda lib

Setting up Cuda bin

Setting up Cuda nvvm

Setting up CUPTI include

Setting up CUPTI lib64

Configuration finished

8- Build TensorFlow

Warning Resource Intensive I recommend having at least 8GB of computer memory.

To build TensorFlow with GPU support: this will take some minutes

$ bazel build --config=opt --config=cuda //tensorflow/tools/pip\_package:build\_pip\_package

9- Build & Install Pip Package

(Note that your current path in terminal is ~/tensorflow)

This will build the pip package required for installing TensorFlow in your ~/tensorflow\_pkg [you can change this directory as the one you like]

$ bazel-bin/tensorflow/tools/pip\_package/build\_pip\_package ~/tensorflow\_pkg

Now you can cd into the directory where you build your tensorflow, for example my case is ~/tensorflow\_pkg

$ sudo pip install tensorflow-1.0.1-cp27-cp27mu-linux\_x86\_64.whl

# with no spaces after tensorflow hit tab before hitting enter to fill in blanks

Remember that, at any time, you can manually force the project to be reconfigured (run the ./configure file in step 7 above to reconfigure) and built from scratch by emptying the directory ~/tensorflow\_pkg with:

$ rm -rf ./\*

10- Test Your Installation

Finally, time to test our installation. Close all your terminals and open a new terminal to test. Also, make sure your Terminal is not in the ‘tensorflow’ directory.

$ cd ~

$ python # or python3 for python 3

After you entered python shell type the following

import tensorflow as tf

hello = tf.constant('Hello, TensorFlow!')

sess = tf.Session()

print(sess.run(hello))

Hello, TensorFlow!

print sess.run(tf.constant(12)\*tf.constant(3))

36

To check tensorflow version that you installed type:-

$ python -c 'import tensorflow as tf; print(tf.\_\_version\_\_)'