Installing Tensorflow from source on Ubuntu 16.04 (CPU,GPU)

Prerequisites:

1.1 Python : Install python from miniconda. Avoid anaconda since it comes with too many redundant packages.

1.1.1 Steps to install python:
1. Download the required python from the above link. (Python 3.6 Recommended)
2. In terminal run:

```
$ bash <yourpythonfile.sh>
```

Note: Everytime it asks about updating path and where to install leave it at its default. (Just press enter.)

1.2 Changing the system python (in order to use sudo with miniconda python):

1. First add all the pythons in a list like this:

   ```
   $ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
   $ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.5 2
   $ sudo update-alternatives --install /usr/bin/python python /home/rakshith/miniconda3/bin/python3.6 3
   ```

   (Enter your corresponding miniconda python path, the last number in each command is basically a priority)

2. Everytime you want to give sudo permission to minconda python you can do so by:

   ```
   $ sudo update-alternatives --config python
   ```

3. Then type the corresponding index number of the python you want to give sudo permission to and press enter.

4. Make sure to change the permission back to system python2.7 once your done.

1.3 Assigning root permissions to conda environment: [Warning: Be very careful while doing this. One mistake can destroy your system.]
Conda sometimes throws permission error in order to fix this. Run this in terminal:

```
$ sudo chown -R USERNAME /home/USERNAME/miniconda3
```

Here replace USERNAME with your username and the path with the path of your miniconda.

BE VERY CAREFULL WITH THE PATH. YOU WILL HAVE TO REINSTALL OS IF SOMETHING GOES WRONG.

1.4 Installing Bazel
The steps to install bazel is described very well in the official website.

1.5 Installing tensorflow prequisite packages
If you are using miniconda python make sure you are giving sudo permission by using the method in step 1.2 .

Then run in terminal:

```
$ sudo apt-get install python-numpy python-dev python-pip python-wheel
```

Make sure its installed in conda enviroment, to check this run in terminal:

```
$ conda list
```

If you see these packages then everything worked fine. If its missing you probably installed these packages for a different python in your system.

1.6 GPU Prequisites [Optional, only if you want GPU support]:
The official NVIDIA Guide is very messy and complicated. These steps are taken from there and are arranged in sequence:

Firstly Check if your GPU is supported for cuda library in this link.

Second if you want to follow every step in NVIDIA's documentation for cuda toolkit here is the link. Please make sure the cuda version is compatible with the current Tensorflow release.

This is the official link for cuDNN.

Simplified steps:

1.6.1 Prerequisite steps:

i. Check if you have a CUDA-Capable GPU. (its better to do this even if your GPU is listed in that website.):

```
$ lspci | grep -i nvidia
```

If you do not see any settings, update the PCI hardware database that Linux maintains by entering `update-pciids` (generally found in `/sbin`) at the command line and rerun the previous `lspci` command.

ii. Verify you have a supported version of linux:

```
$ uname -m && cat /etc/*release
```

iii. Check if you have the correct version of gcc:

```
$ gcc --version
```

iv. Verify the system has the correct kernel headers and development packages installed:

```
$ uname -r
```

v. Then run:

```
$ sudo apt-get install linux-headers-$(uname -r)
```

1.6.2 Install cuda sdk:

Download the correct version in this link. Make sure you are downloading `deb (local)`. The file will be about 1.2 GB in size.

Installation:
i. Change the file name to the downloaded version:

```
$ sudo dpkg -i <yourdownloadedfilename>.deb
```

ii. `$ sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub`
iii. `$ sudo apt-get update`
iv. `$ sudo apt-get install cuda`
v. `$ sudo apt-get install cuda-drivers`
vi. `$ /usr/bin/nvidia-persistenced --verbose`
vii. Optional Step:

```
sudo apt-get install g++ freeglut3-dev build-essential libx11-dev \ libxmu-d
ev libxi-dev libglu1-mesa libglu1-mesa-dev
```

Finally update your system `PATH` like this:

```
$ sudo nano /etc/environment
```

Add this: `/usr/local/cuda-<yourversion>/bin` towards the end (replace < yourversion > with the one you installed, provided you installed cuda at its default path. So it would look something like this:

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/game
s:/usr/local/games:/usr/local/cuda-9.1/bin"
```

`ctrl+O` and then `ctrl+X`

Reboot the system and check if everything was installed properly. To check run:

```
$ nvidia-smi
```

This should display your graphic card with all its details and cuda driver version.
Also check:

```
$ nvcc --version
```

If this fails to give any output even after rebooting. Reinstall cuda toolkit:

```
$ sudo apt-get install cuda-toolkit
```

Check with `nvcc` command again once this is done. It should be fixed.

1.6.3 Installing cuDNN

The Official Documentation is good for this just follow it. You need to sign up with NVIDIA as a developer in order to install this. You can download the 3 files here.

Just download the `.deb` files and follow step `2.3.2` in the Installation Guide.

Finally run in terminal:

```
$ sudo apt-get install libcupti-dev
```

Reboot once everything is installed.

Installing Tensorflow

This is the official documentation for source instalation. Follow these steps at your own risk:

1. In your home directory create a folder and name it `Tensorflow`. Open it and open a terminal window from the directory.
2. `$ git clone https://github.com/tensorflow/tensorflow`
3. Checkout the latest stable build (change it to r followed by the number):

   ```
   $ cd tensorflow
   $ git checkout r1.4
   ```

4. `./configure`
5. Make sure everything is left at no except for `jemalloc` and `cuda` support (you can disable cuda if you dont want GPU support) . Disable `clang compiler` for cuda and make sure you enter the right cuda versions and cuDNN versions along with their paths. Leave optimisations also at its default (just press enter). It will set it to - `march=native`.
6. For CPU only:

   ```
   $ bazel build --config=opt //tensorflow/tools/pip_package:build_pip_pack
   age
   ```

   For GPU+CPU:

   ```
   $ bazel build --config=opt --config=cuda //tensorflow/tools/pip_package:
   build_pip_package
   ```

If it fails due to bazel version problems you can try appending `--incompatible_load_argument_is_label=false` this flag to the build command, so it would look like this:

```
$ bazel build --config=opt --incompatible_load_argument_is_label=false /
  /tensorflow/tools/pip_package:build_pip_package
```

7. If everything in successful, then build the wheel file:

```
$ bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorfl
ow_pkg
```

8. Finally install tensorflow: (if you want to install with `sudo` make sure you give your corresponding python `sudo` permission by following step `1.2`)

```
$ sudo pip install /tmp/tensorflow_pkg/<yourgeneratedwheelfilename>.whl
```

(Replace the filename)

Debugging:

1. NVCC Not Found
2. System Python change

References:

1. pradeepadiga blog