Installing TensorFlow on Ubuntu

This guide explains how to install TensorFlow on Ubuntu. Although these instructions might also work on other Linux variants, we have only tested (and we only support) these instructions on machines meeting the following requirements:

- 64-bit desktops or laptops
- Ubuntu 14.04 or higher

Determine which TensorFlow to install

You must choose one of the following types of TensorFlow to install:

- **TensorFlow with CPU support only**. If your system does not have a NVIDIA® GPU, you must install this version. Note that this version of TensorFlow is typically much easier to install (typically, in 5 or 10 minutes), so even if you have an NVIDIA GPU, we recommend installing this version first.
- **TensorFlow with GPU support**. TensorFlow programs typically run significantly faster on a GPU than on a CPU. Therefore, if your system has a NVIDIA® GPU meeting the prerequisites shown below and you need to run performance-critical applications, you should ultimately install this version.

NVIDIA requirements to run TensorFlow with GPU support

If you are installing TensorFlow with GPU support using one of the mechanisms described in this guide, then the following NVIDIA software must be installed on your system:

- CUDA® Toolkit 8.0. For details, see NVIDIA's documentation (http://docs.nvidia.com/cuda/cuda-installation-guide-linux/#axzz4VZnqTJ2A). Ensure that you append the relevant Cuda pathnames to the `LD_LIBRARY_PATH` environment variable as described in the NVIDIA documentation.
- The NVIDIA drivers associated with CUDA Toolkit 8.0.
- cuDNN v6.0. For details, see NVIDIA's documentation (https://developer.nvidia.com/cudnn). Ensure that you create the `CUDA_HOME` environment variable as described in the NVIDIA documentation.
- GPU card with CUDA Compute Capability 3.0 or higher. See NVIDIA documentation

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

- The libcupti-dev library, which is the NVIDIA CUDA Profile Tools Interface. This library provides advanced profiling support. To install this library, issue the following command for CUDA Toolkit >= 8.0:

  ```
  $ sudo apt-get install cuda-command-line-tools
  ```

  and add its path to your `LD_LIBRARY_PATH` environment variable:

  ```
  $ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/extras/CU
  ```

  For CUDA Toolkit <= 7.5 do:

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

If you have an earlier version of the preceding packages, please upgrade to the specified versions. If upgrading is not possible, then you may still run TensorFlow with GPU support, but only if you do the following:

- Install TensorFlow from sources as documented in Installing TensorFlow from Sources (https://www.tensorflow.org/install/install_sources).
- Install or upgrade to at least the following NVIDIA versions:
  - CUDA toolkit 7.0 or greater
  - cuDNN v3 or greater
  - GPU card with CUDA Compute Capability 3.0 or higher.

Determine how to install TensorFlow

You must pick the mechanism by which you install TensorFlow. The supported choices are as follows:

- Virtualenv (#InstallingVirtualenv)
- "native" pip (#InstallingNativePip)
- Docker (#InstallingDocker)
- Anaconda (#InstallingAnaconda)
- installing from sources, which is documented in a separate guide (https://www.tensorflow.org/install/install_sources).

We recommend the Virtualenv installation. Virtualenv (https://virtualenv.pypa.io/en/stable/) is a virtual Python environment isolated from other Python development, incapable of interfering with or being affected by other Python programs on the same machine. During the Virtualenv installation process, you will install not only TensorFlow but also all the packages that TensorFlow requires. (This is actually pretty easy.) To start working with TensorFlow, you simply need to "activate" the virtual environment. All in all, Virtualenv provides a safe and reliable mechanism for installing and running TensorFlow.

Native pip installs TensorFlow directly on your system without going through any container system. **We recommend the native pip install for system administrators aiming to make TensorFlow available to everyone on a multi-user system.** Since a native pip installation is not walled-off in a separate container, the pip installation might interfere with other Python-based installations on your system. However, if you understand pip and your Python environment, a native pip installation often entails only a single command.

Docker completely isolates the TensorFlow installation from pre-existing packages on your machine. The Docker container contains TensorFlow and all its dependencies. Note that the Docker image can be quite large (hundreds of MBs). You might choose the Docker installation if you are incorporating TensorFlow into a larger application architecture that already uses Docker.

In Anaconda, you may use conda to create a virtual environment. However, within Anaconda, we recommend installing TensorFlow with the `pip install` command, not with the `conda install` command.

NOTE: The conda package is community supported, not officially supported. That is, the TensorFlow team neither tests nor maintains the conda package. Use that package at your own risk.

Installing with Virtualenv

Take the following steps to install TensorFlow with Virtualenv:

1. Install pip and Virtualenv by issuing one of the following commands:

```
$ sudo apt-get install python-pip python-dev python-virtualenv # fo
$ sudo apt-get install python3-pip python3-dev python-virtualenv # for
```

2. Create a Virtualenv environment by issuing one of the following commands:

```
$ virtualenv --system-site-packages targetDirectory # for Python 2.
```

```
$ virtualenv --system-site-packages -p python3 targetDirectory # for P
```

where ***targetDirectory*** specifies the top of the Virtualenv tree. Our instructions assume that ***targetDirectory*** is `~/tensorflow`, but you may choose any directory.

3. Activate the Virtualenv environment by issuing one of the following commands:

```
$ source ~/tensorflow/bin/activate # bash, sh, ksh, or zsh
$ source ~/tensorflow/bin/activate.csh  # csh or tcsh
```

The preceding `source` command should change your prompt to the following:

```
(tensorflow)$
```

4. Ensure pip ≥8.1 is installed:

```
(tensorflow)$ easy_install -U pip
```

5. Issue one of the following commands to install TensorFlow in the active Virtualenv environment:

```
(tensorflow)$ pip install --upgrade tensorflow      # for Python 2.
(tensorflow)$ pip3 install --upgrade tensorflow     # for Python 3.n
(tensorflow)$ pip install --upgrade tensorflow-gpu  # for Python 2.7 a
(tensorflow)$ pip3 install --upgrade tensorflow-gpu # for Python 3.n a
```

If the above command succeeds, skip Step 6. If the preceding command fails, perform Step 6.

6. (Optional) If Step 5 failed (typically because you invoked a pip version lower than 8.1), install TensorFlow in the active Virtualenv environment by issuing a command of the following format:

```
(tensorflow)$ pip install --upgrade tfBinaryURL   # Python 2.7
(tensorflow)$ pip3 install --upgrade tfBinaryURL  # Python 3.n
```

where ***tfBinaryURL*** identifies the URL of the TensorFlow Python package. The appropriate value of ***tfBinaryURL***depends on the operating system, Python version, and GPU support. Find the appropriate value for ***tfBinaryURL*** for your system here (#the_url_of_the_tensorflow_python_package). For example, if you are installing TensorFlow for Linux, Python 3.4, and CPU-only support, issue the following command to install TensorFlow in the active Virtualenv environment:

```
(tensorflow)$ pip3 install --upgrade \
```

```
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.5.0-
```

If you encounter installation problems, see Common Installation Problems (#common_installation_problems).

Next Steps

After installing TensorFlow, validate the installation (#ValidateYourInstallation).

Note that you must activate the Virtualenv environment each time you use TensorFlow. If the Virtualenv environment is not currently active, invoke one of the following commands:

```
 $ source ~/tensorflow/bin/activate      # bash, sh, ksh, or zsh
$ source ~/tensorflow/bin/activate.csh  # csh or tcsh
```

When the Virtualenv environment is active, you may run TensorFlow programs from this shell. Your prompt will become the following to indicate that your tensorflow environment is active:

```
(tensorflow)$
```

When you are done using TensorFlow, you may deactivate the environment by invoking the **deactivate** function as follows:

```
(tensorflow)$ deactivate
```

The prompt will revert back to your default prompt (as defined by the PS1 environment variable).

Uninstalling TensorFlow

To uninstall TensorFlow, simply remove the tree you created. For example:

```
$ rm -r targetDirectory
```

Installing with native pip

You may install TensorFlow through pip, choosing between a simple installation procedure or a more complex one.

Note: The REQUIRED_PACKAGES section of setup.py (https://github.com/tensorflow/tensorflow/blob/master/tensorflow/tools/pip_package/setup.py) lists the TensorFlow packages that pip will install or upgrade.

Prerequisite: Python and Pip

Python is automatically installed on Ubuntu. Take a moment to confirm (by issuing a **`python -V`** command) that one of the following Python versions is already installed on your system:

- Python 2.7
- Python 3.4+

The pip or pip3 package manager is *usually* installed on Ubuntu. Take a moment to confirm (by issuing a **`pip -V`** or **`pip3 -V`** command) that pip or pip3 is installed. We strongly recommend version 8.1 or higher of pip or pip3. If Version 8.1 or later is not installed, issue the following command, which will either install or upgrade to the latest pip version:

```
$ sudo apt-get install python-pip python-dev   # for Python 2.7
$ sudo apt-get install python3-pip python3-dev # for Python 3.n
```

Install TensorFlow

Assuming the prerequisite software is installed on your Linux host, take the following steps:

1. Install TensorFlow by invoking **one** of the following commands:

   ```
   $ pip install tensorflow      # Python 2.7; CPU support (no GPU sup
   $ pip3 install tensorflow     # Python 3.n; CPU support (no GPU suppor
   $ pip install tensorflow-gpu  # Python 2.7;  GPU support
   $ pip3 install tensorflow-gpu # Python 3.n; GPU support
   ```

   If the preceding command runs to completion, you should now validate your installation (#ValidateYourInstallation).

2. (Optional.) If Step 1 failed, install the latest version of TensorFlow by issuing a command of the following format:

   ```
   $ sudo pip  install --upgrade tfBinaryURL   # Python 2.7
   $ sudo pip3 install --upgrade tfBinaryURL   # Python 3.n
   ```

   where ***tfBinaryURL*** identifies the URL of the TensorFlow Python package. The appropriate value of ***tfBinaryURL*** depends on the operating system, Python version,

and GPU support. Find the appropriate value for ***tfBinaryURL*** here (#the_url_of_the_tensorflow_python_package). For example, to install TensorFlow for Linux, Python 3.4, and CPU-only support, issue the following command:

```
$ sudo pip3 install --upgrade \
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.5.0-
```

If this step fails, see Common Installation Problems (#common_installation_problems).

Next Steps

After installing TensorFlow, validate your installation (#ValidateYourInstallation).

Uninstalling TensorFlow

To uninstall TensorFlow, issue one of following commands:

```
$ sudo pip uninstall tensorflow  # for Python 2.7
$ sudo pip3 uninstall tensorflow # for Python 3.n
```

Installing with Docker

Take the following steps to install TensorFlow through Docker:

1. Install Docker on your machine as described in the Docker documentation (http://docs.docker.com/engine/installation/).
2. Optionally, create a Linux group called **docker** to allow launching containers without sudo as described in the Docker documentation (https://docs.docker.com/engine/installation/linux/linux-postinstall/). (If you don't do this step, you'll have to use sudo each time you invoke Docker.)
3. To install a version of TensorFlow that supports GPUs, you must first install nvidia-docker (https://github.com/NVIDIA/nvidia-docker), which is stored in github.
4. Launch a Docker container that contains one of the TensorFlow binary images (https://hub.docker.com/r/tensorflow/tensorflow/tags/).

The remainder of this section explains how to launch a Docker container.

CPU-only

To launch a Docker container with CPU-only support (that is, without GPU support), enter a command of the following format:

```
$ docker run -it -p hostPort:containerPort TensorFlowCPUImage
```

where:

- *`-p hostPort:containerPort`* is optional. If you plan to run TensorFlow programs from the shell, omit this option. If you plan to run TensorFlow programs as Jupyter notebooks, set both *`hostPort`* and *`containerPort`* to 8888. If you'd like to run TensorBoard inside the container, add a second **`-p`** flag, setting both *hostPort* and *containerPort* to 6006.
- *`TensorFlowCPUImage`* is required. It identifies the Docker container. Specify one of the following values:
  - `gcr.io/tensorflow/tensorflow`, which is the TensorFlow CPU binary image.
  - `gcr.io/tensorflow/tensorflow:latest-devel`, which is the latest TensorFlow CPU Binary image plus source code.
  - `gcr.io/tensorflow/tensorflow:`*`version`*, which is the specified version (for example, 1.1.0rc1) of TensorFlow CPU binary image.
  - `gcr.io/tensorflow/tensorflow:`*`version`*`-devel`, which is the specified version (for example, 1.1.0rc1) of the TensorFlow GPU binary image plus source code.

  `gcr.io` is the Google Container Registry. Note that some TensorFlow images are also available at dockerhub (https://hub.docker.com/r/tensorflow/tensorflow/).

For example, the following command launches the latest TensorFlow CPU binary image in a Docker container from which you can run TensorFlow programs in a shell:

```
$ docker run -it gcr.io/tensorflow/tensorflow bash
```

The following command also launches the latest TensorFlow CPU binary image in a Docker container. However, in this Docker container, you can run TensorFlow programs in a Jupyter notebook:

```
$ docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow
```

Docker will download the TensorFlow binary image the first time you launch it.

GPU support

Prior to installing TensorFlow with GPU support, ensure that your system meets all NVIDIA software requirements (#NVIDIARequirements). To launch a Docker container with NVidia GPU support, enter a command of the following format:

```
$ nvidia-docker run -it -p hostPort:containerPort TensorFlowGPUImage
```

where:

- *-p hostPort:containerPort* is optional. If you plan to run TensorFlow programs from the shell, omit this option. If you plan to run TensorFlow programs as Jupyter notebooks, set both *hostPort* and ***containerPort*** to **8888**.
- *TensorFlowGPUImage* specifies the Docker container. You must specify one of the following values:
  - `gcr.io/tensorflow/tensorflow:latest-gpu`, which is the latest TensorFlow GPU binary image.
  - `gcr.io/tensorflow/tensorflow:latest-devel-gpu`, which is the latest TensorFlow GPU Binary image plus source code.
  - `gcr.io/tensorflow/tensorflow:`*`version`*`-gpu`, which is the specified version (for example, 0.12.1) of the TensorFlow GPU binary image.
  - `gcr.io/tensorflow/tensorflow:`*`version`*`-devel-gpu`, which is the specified version (for example, 0.12.1) of the TensorFlow GPU binary image plus source code.

We recommend installing one of the **`latest`** versions. For example, the following command launches the latest TensorFlow GPU binary image in a Docker container from which you can run TensorFlow programs in a shell:

```
$ nvidia-docker run -it gcr.io/tensorflow/tensorflow:latest-gpu bash
```

The following command also launches the latest TensorFlow GPU binary image in a Docker container. In this Docker container, you can run TensorFlow programs in a Jupyter notebook:

```
$ nvidia-docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow:latest
```

The following command installs an older TensorFlow version (0.12.1):

```
$ nvidia-docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow:0.12.
```

Docker will download the TensorFlow binary image the first time you launch it. For more details see the TensorFlow docker readme (https://github.com/tensorflow/tensorflow/tree/master/tensorflow/tools/docker).

Next Steps

You should now validate your installation (#ValidateYourInstallation).

Installing with Anaconda

Take the following steps to install TensorFlow in an Anaconda environment:

1. Follow the instructions on the Anaconda download site (https://www.continuum.io/downloads) to download and install Anaconda.
2. Create a conda environment named `tensorflow` to run a version of Python by invoking the following command:

   ```
   $ conda create -n tensorflow pip python=2.7 # or python=3.3, etc.
   ```

3. Activate the conda environment by issuing the following command:

   ```
   $ source activate tensorflow
    (tensorflow)$  # Your prompt should change
   ```

4. Issue a command of the following format to install TensorFlow inside your conda environment:

   ```
   (tensorflow)$ pip install --ignore-installed --upgrade tfBinaryURL
   ```

   where ***tfBinaryURL*** is the URL of the TensorFlow Python package (#the_url_of_the_tensorflow_python_package). For example, the following command installs the CPU-only version of TensorFlow for Python 3.4:

   ```
    (tensorflow)$ pip install --ignore-installed --upgrade \
    https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.5.0-
   ```

Validate your installation

To validate your TensorFlow installation, do the following:

1. Ensure that your environment is prepared to run TensorFlow programs.
2. Run a short TensorFlow program.

Prepare your environment

If you installed on native pip, Virtualenv, or Anaconda, then do the following:

1. Start a terminal.
2. If you installed with Virtualenv or Anaconda, activate your container.
3. If you installed TensorFlow source code, navigate to any directory *except* one containing TensorFlow source code.

If you installed through Docker, start a Docker container from which you can run bash. For example:

```
$ docker run -it gcr.io/tensorflow/tensorflow bash
```

Run a short TensorFlow program

Invoke python from your shell as follows:

```
$ python
```

Enter the following short program inside the python interactive shell:

```
# Python
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

If the system outputs the following, then you are ready to begin writing TensorFlow programs:

```
Hello, TensorFlow!
```

If you are new to TensorFlow, see Getting Started with TensorFlow (https://www.tensorflow.org/get_started/premade_estimators).

If the system outputs an error message instead of a greeting, see Common installation problems (#common_installation_problems).

Common installation problems

We are relying on Stack Overflow to document TensorFlow installation problems and their remedies. The following table contains links to Stack Overflow answers for some common installation problems. If you encounter an error message or other installation problem not listed in the following table, search for it on Stack Overflow. If Stack Overflow doesn't show the error message, ask a new question about it on Stack Overflow and specify the **`tensorflow`** tag.

Stack Overflow Link	Error Message
36159194 (https://stackoverflow.com/q/36159194)	`ImportError: libcudart.so.Version: cannot open shared No such file or directory`
41991101 (https://stackoverflow.com/q/41991101)	`ImportError: libcudnn.Version: cannot open shared obje No such file or directory`
36371137 (http://stackoverflow.com/q/36371137) and here (#Protobuf31)	`libprotobuf ERROR google/protobuf/src/google/protobuf/ protocol message was rejected because it was too big To increase the limit (or to disable these warnings) CodedInputStream::SetTotalBytesLimit() in google/pro`
35252888 (https://stackoverflow.com/q/35252888)	`Error importing tensorflow. Unless you are using bazel not try to import tensorflow from its source directo tensorflow source tree, and relaunch your python int there.`
33623453 (https://stackoverflow.com/q/33623453)	`IOError: [Errno 2] No such file or directory: '/tmp/pip-o6Tpui-build/setup.py'`
42006320 (http://stackoverflow.com/q/42006320)	`ImportError: Traceback (most recent call last): File ".../tensorflow/core/framework/graph_pb2.py", l from google.protobuf import descriptor as _descripto ImportError: cannot import name 'descriptor'`

35190574 (https://stackoverflow.com/questions/35190574)	`SSLError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate failed`

42009190 (http://stackoverflow.com/q/42009190)	`Installing collected packages: setuptools, protobuf,`
	`Found existing installation: setuptools 1.1.6`
	`Uninstalling setuptools-1.1.6:`
	`Exception:`
	`...`
	`[Errno 1] Operation not permitted:`
	`'/tmp/pip-a1DXRT-uninstall/.../lib/python/_markerlib`

36933958 (http://stackoverflow.com/questions/36933958)	`...`
	`Installing collected packages: setuptools, protobuf,`
	`Found existing installation: setuptools 1.1.6`
	`Uninstalling setuptools-1.1.6:`
	`Exception:`
	`...`
	`[Errno 1] Operation not permitted:`
	`'/tmp/pip-a1DXRT-uninstall/System/Library/Frameworks`
	`Versions/2.7/Extras/lib/python/_markerlib'`

The URL of the TensorFlow Python package

A few installation mechanisms require the URL of the TensorFlow Python package. The value you specify depends on three factors:

- operating system
- Python version
- CPU only vs. GPU support

This section documents the relevant values for Linux installations.

Python 2.7

CPU only:

```
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.5.0-cp2
```

GPU support:

```
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.5.0
```

Note that GPU support requires the NVIDIA hardware and software described in NVIDIA requirements to run TensorFlow with GPU support (#NVIDIARequirements).

Python 3.4

CPU only:

```
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.5.0-cp3
```

GPU support:

```
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.5.0
```

Note that GPU support requires the NVIDIA hardware and software described in NVIDIA requirements to run TensorFlow with GPU support (#NVIDIARequirements).

Python 3.5

CPU only:

```
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.5.0-cp3
```

GPU support:

```
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.5.0
```

Note that GPU support requires the NVIDIA hardware and software described in NVIDIA requirements to run TensorFlow with GPU support (#NVIDIARequirements).

Python 3.6

CPU only:

```
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.5.0-cp3
```

GPU support:

```
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.5.0
```

Note that GPU support requires the NVIDIA hardware and software described in NVIDIA

requirements to run TensorFlow with GPU support (#NVIDIARequirements).

Protobuf pip package 3.1

You can skip this section unless you are seeing problems related to the protobuf pip package.

NOTE: If your TensorFlow programs are running slowly, you might have a problem related to the protobuf pip package.

The TensorFlow pip package depends on protobuf pip package version 3.1. The protobuf pip package downloaded from PyPI (when invoking `pip install protobuf`) is a Python-only library containing Python implementations of proto serialization/deserialization that can run **10x-50x slower** than the C++ implementation. Protobuf also supports a binary extension for the Python package that contains fast C++ based proto parsing. This extension is not available in the standard Python-only pip package. We have created a custom binary pip package for protobuf that contains the binary extension. To install the custom binary protobuf pip package, invoke one of the following commands:

- for Python 2.7:

```
$ pip install --upgrade \
https://storage.googleapis.com/tensorflow/linux/cpu/protobuf-3.1.0-cp27-
```

- for Python 3.5:

```
$ pip3 install --upgrade \
https://storage.googleapis.com/tensorflow/linux/cpu/protobuf-3.1.0-cp35-
```

Installing this protobuf package will overwrite the existing protobuf package. Note that the binary pip package already has support for protobufs larger than 64MB, which should fix errors such as these:

```
[libprotobuf ERROR google/protobuf/src/google/protobuf/io/coded_stream.
A protocol message was rejected because it was too big (more than 67108864
To increase the limit (or to disable these warnings), see
CodedInputStream::SetTotalBytesLimit() in google/protobuf/io/coded_stream.
```

Last updated January 27, 2018.