Installing TensorFlow from Sources

This guide explains how to build TensorFlow sources into a TensorFlow binary and how to install that TensorFlow binary. Note that we provide well-tested, pre-built TensorFlow binaries for Ubuntu, macOS, and Windows systems. In addition, there are pre-built TensorFlow docker images (https://hub.docker.com/r/tensorflow/tensorflow/). So, don't build a TensorFlow binary yourself unless you are very comfortable building complex packages from source and dealing with the inevitable aftermath should things not go exactly as documented.

If the last paragraph didn't scare you off, welcome. This guide explains how to build TensorFlow on 64-bit desktops and laptops running either of the following operating systems:

- Ubuntu
- macOS X

Note: Some users have successfully built and installed TensorFlow from sources on non-supported systems. Please remember that we do not fix issues stemming from these attempts.

We **do not support** building TensorFlow on Windows. That said, if you'd like to try to build TensorFlow on Windows anyway, use either of the following:

- Bazel on Windows (https://bazel.build/versions/master/docs/windows.html)
- TensorFlow CMake build (https://github.com/tensorflow/tensorflow/tree/r0.12/tensorflow/contrib/cmake)

Determine which TensorFlow to install

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

- **TensorFlow with CPU support only**. If your system does not have a NVIDIA® GPU, build and install this version. Note that this version of TensorFlow is typically easier to build and install, so even if you have an NVIDIA GPU, we recommend building and 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 and you need to run performance-critical applications, you should ultimately build and install this version. Beyond the NVIDIA GPU itself, your system must also fulfill the NVIDIA software requirements described in one of the following documents:
- Installing TensorFlow on Ubuntu (https://www.tensorflow.org/versions/master/install/install_linux#NVIDIARequirements)
- Installing TensorFlow on macOS (https://www.tensorflow.org/versions/master/install/install_mac#NVIDIARequirements)

Clone the TensorFlow repository

Start the process of building TensorFlow by cloning a TensorFlow repository.

To clone **the latest** TensorFlow repository, issue the following command:

```
$ git clone https://github.com/tensorflow/tensorflow
```

The preceding `git clone` command creates a subdirectory named `tensorflow`. After cloning, you may optionally build a **specific branch** (such as a release branch) by invoking the following commands:

```
$ cd tensorflow
$ git checkout Branch # 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 (#PrepareLinux) or macOS (#PrepareMac)

Prepare environment for Linux

Before building TensorFlow on Linux, install the following build tools on your system:

- bazel
- TensorFlow Python dependencies
- optionally, NVIDIA packages to support TensorFlow for GPU.

Install Bazel

If bazel is not installed on your system, install it now by following these directions (https://bazel.build/versions/master/docs/install.html).

Install TensorFlow Python dependencies

To install TensorFlow, you must install the following packages:

- `numpy`, which is a numerical processing package that TensorFlow requires.
- `dev`, which enables adding extensions to Python.
- `pip`, which enables you to install and manage certain Python packages.
- `wheel`, which enables you to manage Python compressed packages in the wheel (.whl) format.

To install these packages for Python 2.7, issue the following command:

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

To install these packages for Python 3.n, issue the following command:

```
$ sudo apt-get install python3-numpy python3-dev python3-pip python3-wheel
```

Optional: install TensorFlow for GPU prerequisites

If you are building TensorFlow without GPU support, skip this section.

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) 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. 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 NVIDIA's Cuda Toolkit.
- cuDNN (>= v3). We recommend version 5.1. For details, see NVIDIA's documentation (https://developer.nvidia.com/cudnn), particularly the description of appending the appropriate pathname to your `LD_LIBRARY_PATH` environment variable.

Finally, you must also install `libcupti` which for Cuda Toolkit >= 8.0 you do via

```
$ 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/CUPTI/lib64
```

For Cuda Toolkit <= 7.5, you install `libcupti-dev` by invoking the following command:

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

Next

After preparing the environment, you must now configure the installation (#ConfigureInstallation).

Prepare environment for macOS

Before building TensorFlow, you must install the following on your system:

- bazel
- TensorFlow Python dependencies.
- optionally, NVIDIA packages to support TensorFlow for GPU.

Install bazel

If bazel is not installed on your system, install it now by following these directions (https://bazel.build/versions/master/docs/install.html#mac-os-x).

Install python dependencies

To install TensorFlow, you must install the following packages:

- six
- numpy, which is a numerical processing package that TensorFlow requires.
- wheel, which enables you to manage Python compressed packages in the wheel (.whl) format.

You may install the python dependencies using pip. If you don't have pip on your machine, we recommend using homebrew to install Python and pip as documented here (http://docs.python-guide.org/en/latest/starting/install/osx/). If you follow these instructions, you will not need to disable SIP.

After installing pip, invoke the following commands:

```
$ sudo pip install six numpy wheel
```

Optional: install TensorFlow for GPU prerequisites

If you do not have brew installed, install it by following these instructions (http://brew.sh/).

After installing brew, install GNU coreutils by issuing the following command:

```
$ brew install coreutils
```

If you want to compile tensorflow and have XCode 7.3 and CUDA 7.5 installed, note that Xcode 7.3 is not yet compatible with CUDA 7.5. To remedy this problem, do either of the following:

- Upgrade to CUDA 8.0.
- Download Xcode 7.2 and select it as your default by issuing the following command:

  ```
  $ sudo xcode-select -s /Application/Xcode-7.2/Xcode.app
  ```

NOTE: Your system must fulfill the NVIDIA software requirements described in one of the following documents:

- Installing TensorFlow on Linux (https://www.tensorflow.org/versions/master/install/install_linux#NVIDIARequirements)
- Installing TensorFlow on Mac OS (https://www.tensorflow.org/versions/master/install/install_mac#NVIDIARequirements)

Configure the installation

The root of the source tree contains a bash script named **`configure`**. This script asks you to identify the pathname of all relevant TensorFlow dependencies and specify other build configuration options such as compiler flags. You must run this script *prior* to creating the pip package and installing TensorFlow.

If you wish to build TensorFlow with GPU, **`configure`** will ask you to specify the version numbers of Cuda and cuDNN. If several versions of Cuda or cuDNN are installed on your system, explicitly select the desired version instead of relying on the default.

One of the questions that **`configure`** will ask is as follows:

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

This question refers to a later phase in which you'll use bazel to build the pip package (#build_the_pip_package). We recommend accepting the default (**`-march=native`**), which will optimize the generated code for your local machine's CPU type. However, if you are building TensorFlow on one CPU type but will run TensorFlow on a different CPU type, then consider specifying a more specific optimization flag as described in the gcc documentation (https://gcc.gnu.org/onlinedocs/gcc-4.5.3/gcc/i386-and-x86_002d64-Options.html).

Here is an example execution of the **`configure`** script. Note that your own input will likely differ from our sample input:

```
$ cd tensorflow  # cd to the top-level directory created
$ ./configure
Please specify the location of python. [Default is /usr/bin/python]: /usr/bin/python2.7
Found possible Python library paths:
  /usr/local/lib/python2.7/dist-packages
  /usr/lib/python2.7/dist-packages
Please input the desired Python library path to use.  Default is [/usr/lib/python2.7/dist-packages]

Using python library path: /usr/local/lib/python2.7/dist-packages
Do you wish to build TensorFlow with MKL support? [y/N]
No MKL support will be enabled for TensorFlow
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]
jemalloc enabled
Do you wish to build TensorFlow with Google Cloud Platform support? [y/N]
No Google Cloud Platform support will be enabled for TensorFlow
Do you wish to build TensorFlow with Hadoop File System support? [y/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]
No XLA support will be enabled for TensorFlow
Do you wish to build TensorFlow with VERBS support? [y/N]
No VERBS support will be enabled for TensorFlow
Do you wish to build TensorFlow with OpenCL support? [y/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
Do you want to use clang as CUDA compiler? [y/N]
nvcc will be used as CUDA compiler
Please specify the Cuda SDK version you want to use, e.g. 7.0. [Leave empty to default to CUDA 8.0]: 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 which gcc should be used by nvcc as the host compiler. [Default is /usr/bin/gcc]:
Please specify the cuDNN version you want to use. [Leave empty to default to cuDNN 6.0]: 6
Please specify the location where cuDNN 6 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"]: 3.0
Do you wish to build TensorFlow with MPI support? [y/N]
MPI support will not be enabled for TensorFlow
Configuration finished
```

If you told **`configure`** to build for GPU support, then **`configure`** will create a canonical set of symbolic links to the Cuda libraries on your system. Therefore, every time you change the Cuda library paths, you must rerun the **`configure`** script before re-invoking the **`bazel build`** command.

Note the following:

- Although it is possible to build both Cuda and non-Cuda configs under the same source tree, we recommend running **`bazel clean`** when switching between these two configurations in the same source tree.
- If you don't run the **`configure`** script *before* running the **`bazel build`** command, the **`bazel build`** command will fail.

Build the pip package

To build a pip package for TensorFlow with CPU-only support, you would typically invoke the following command:

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

To build a pip package for TensorFlow with GPU support, invoke the following command:

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

NOTE on gcc 5 or later: the binary pip packages available on the TensorFlow website are built with gcc 4, which uses the older ABI. To make your build compatible with the older ABI, you need to add `--cxxopt="-D_GLIBCXX_USE_CXX11_ABI=0"` to your **`bazel build`** command. ABI compatibility allows custom ops built against the TensorFlow pip package to continue to work against your built package.

Tip: By default, building TensorFlow from sources consumes a lot of RAM. If RAM is an issue on your system, you may limit RAM usage by specifying `--local_resources 2048,.5,1.0` while invoking **`bazel`**.

The **`bazel build`** command builds a script named **`build_pip_package`**. Running this script as follows will build a **`.whl`** file within the **`/tmp/tensorflow_pkg`** directory:

```
$ bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg
```

Install the pip package

Invoke **`pip install`** to install that pip package. The filename of the **`.whl`** file depends on your platform. For example, the following command will install the pip package

for TensorFlow 1.4.0 on Linux:

```
$ sudo pip install /tmp/tensorflow_pkg/tensorflow-1.4.0-py2-none-any.whl
```

Validate your installation

Validate your TensorFlow installation by doing the following:

Start a terminal.

Change directory (**`cd`**) to any directory on your system other than the **`tensorflow`** subdirectory from which you invoked the **`configure`** command.

Invoke python:

```
$ 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 @{$get_started/get_started$Getting Started with TensorFlow}.

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

Common installation problems

The installation problems you encounter typically depend on the operating system. See the "Common installation problems" section of one of the following guides:

- Installing TensorFlow on Linux (https://www.tensorflow.org/versions/master/install/install_linux#CommonInstallationProblems)
- Installing TensorFlow on Mac OS (https://www.tensorflow.org/versions/master/install/install_mac#CommonInstallationProblems)
- Installing TensorFlow on Windows (https://www.tensorflow.org/versions/master/install/install_windows#CommonInstallationProblems)

Beyond the errors documented in those two guides, the following table notes additional errors specific to building TensorFlow. Note that we are relying on Stack Overflow as the repository for build and installation problems. If you encounter an error message not listed in the preceding two guides or in the following table, search for it on Stack Overflow. If Stack Overflow doesn't show the error message, ask a new question on Stack Overflow and specify the **`tensorflow`** tag.

Stack Overflow Link	Error Message
41293077 (https://stackoverflow.com/questions/41293077/how-to-compile-tensorflow-with-sse4-2-and-avx-instructions)	`W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on your machine and could speed up CPU computations.`
42013316 (http://stackoverflow.com/q/42013316)	`ImportError: libcudart.so.8.0: cannot open shared object file: No such file or directory`
42013316 (http://stackoverflow.com/q/42013316)	`ImportError: libcudnn.5: cannot open shared object file: No such file or directory`
35953210 (http://stackoverflow.com/q/35953210)	Invoking `python` or `ipython` generates the following error: `ImportError: cannot import name pywrap_tensorflow`

Tested source configurations

Linux

Version:	CPU/GPU:	Python Version:	Compiler:	Build Tools:	cuDNN:	CUDA:
tensorflow-1.4.0	CPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.5.4	N/A	N/A
tensorflow_gpu-1.4.0	GPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.5.4	6	8
tensorflow-1.3.0	CPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	N/A	N/A
tensorflow_gpu-1.3.0	GPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	6	8
tensorflow-1.2.0	CPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	N/A	N/A
tensorflow_gpu-1.2.0	GPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	5.1	8

tensorflow-1.1.0	CPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	N/A	N/A
tensorflow_gpu-1.1.0	GPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8
tensorflow-1.0.0	CPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	N/A	N/A
tensorflow_gpu-1.0.0	GPU	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8

Mac

Version:	CPU/GPU:	Python Version:	Compiler:	Build Tools:	cuDNN:	CUDA:
tensorflow-1.4.0	CPU	2.7, 3.3-3.6	Clang from xcode	Bazel 0.5.4	N/A	N/A
tensorflow-1.3.0	CPU	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.5	N/A	N/A
tensorflow-1.2.0	CPU	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.5	N/A	N/A
tensorflow-1.1.0	CPU	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.2	N/A	N/A
tensorflow_gpu-1.1.0	GPU	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.2	5.1	8
tensorflow-1.0.0	CPU	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.2	N/A	N/A
tensorflow_gpu-1.0.0	GPU	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.2	5.1	8

Windows

Version:	CPU/GPU:	Python Version:	Compiler:	Build Tools:	cuDNN:	CUDA:
tensorflow-1.4.0	CPU	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	N/A	N/A
tensorflow_gpu-1.4.0	GPU	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	6	8
tensorflow-1.3.0	CPU	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	N/A	N/A
tensorflow_gpu-1.3.0	GPU	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	6	8
tensorflow-1.2.0	CPU	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	N/A	N/A
tensorflow_gpu-1.2.0	GPU	3.5-3.6	MSVC 2015 update 3	Cmake v3.6.3	5.1	8
tensorflow-1.1.0	CPU	3.5	MSVC 2015 update 3	Cmake v3.6.3	N/A	N/A
tensorflow_gpu-1.1.0	GPU	3.5	MSVC 2015 update 3	Cmake v3.6.3	5.1	8
tensorflow-1.0.0	CPU	3.5	MSVC 2015 update 3	Cmake v3.6.3	N/A	N/A
tensorflow_gpu-1.0.0	GPU	3.5	MSVC 2015 update 3	Cmake v3.6.3	5.1	8

Last updated November 8, 2017.