Build from source

Build a TensorFlow *pip* package from source and install it on Ubuntu Linux and macOS. While the instructions might work for other systems, it is only tested and supported for Ubuntu and macOS.

We already provide well-tested, pre-built TensorFlow packages (https://www.tensorflow.org/install/pip) fo and macOS systems.

Setup for Linux and macOS

Install the following build tools to configure your development environment.

Install Python and the TensorFlow package dependencies

Ubuntumac OS (#mac-os)

```
$ sudo apt install python-dev python-pip  # or python3-dev python3-pip
```

Install the TensorFlow *pip* package dependencies (if using a virtual environment, omit the `--user` argument):

```
 install -U --user pip six numpy wheel setuptools mock 'future>=0.17.1'
 install -U --user keras_applications --no-deps
 install -U --user keras_preprocessing --no-deps
```

A **`pip`** version >19.0 is required to install the TensorFlow 2.0 **`.whl`** package. Additional required dependenc ed in the **`setup.py`** s://github.com/tensorflow/tensorflow/blob/master/tensorflow/tools/pip_package/setup.py) file under **RED_PACKAGES**.

Install Bazel

To build TensorFlow, you will need to install Bazel. Bazelisk (https://github.com/bazelbuild/bazelisk) is an easy way to install Bazel and automatically downloads the correct Bazel version for TensorFlow. For ease of use, add Bazelisk as the `bazel` executable in your `PATH`.

If Bazelisk is not available, you can manually install Bazel (https://docs.bazel.build/versions/master/install.html). Make sure to install a supported Bazel version: any version between `_TF_MIN_BAZEL_VERSION` and `_TF_MAX_BAZEL_VERSION` as specified in `tensorflow/configure.py`.

Install GPU support (optional, Linux only)

There is *no* GPU support for macOS.

Read the GPU support (https://www.tensorflow.org/install/gpu) guide to install the drivers and additional software required to run TensorFlow on a GPU.

t is easier to set up one of TensorFlow's GPU-enabled Docker images (#docker_linux_builds).

Download the TensorFlow source code

Use Git (https://git-scm.com/) to clone the TensorFlow repository (https://github.com/tensorflow/tensorflow):

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

The repo defaults to the `master` development branch. You can also checkout a release branch (https://github.com/tensorflow/tensorflow/releases) to build:

```
 checkout branch_name  # r1.9, r1.10, etc.
```

Configure the build

Configure your system build by running the **`./configure`** at the root of your TensorFlow source tree. This script prompts you for the location of TensorFlow dependencies and asks for additional build configuration options (compiler flags, for example).

```
onfigure
```

Sample session

The **`./configure`** script The following shows a sample run of **`./configure`** (your session may differ):

View sample configuration session

```
$ ./configure
You have bazel 0.15.0 installed.
Please specify the location of python. [Default is /usr/bin/python]: /usr/bin/

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/py

Do you wish to build TensorFlow with jemalloc as malloc support? [Y/n]:
jemalloc as malloc support will be enabled for TensorFlow.

Do you wish to build TensorFlow with Google Cloud Platform support? [Y/n]:
Google Cloud Platform support will be enabled for TensorFlow.

Do you wish to build TensorFlow with Hadoop File System support? [Y/n]:
Hadoop File System support will be enabled for TensorFlow.

Do you wish to build TensorFlow with Amazon AWS Platform support? [Y/n]:
Amazon AWS Platform support will be enabled for TensorFlow.

Do you wish to build TensorFlow with Apache Kafka Platform support? [Y/n]:
Apache Kafka Platform support will be enabled for TensorFlow.

Do you wish to build TensorFlow with XLA JIT support? [y/N]:
No XLA JIT support will be enabled for TensorFlow.

Do you wish to build TensorFlow with GDR support? [y/N]:
No GDR 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 SYCL support? [y/N]:
No OpenCL SYCL 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 the CUDA SDK version you want to use. [Leave empty to default t

Please specify the location where CUDA 9.0 toolkit is installed. Refer to REAI

Please specify the cuDNN version you want to use. [Leave empty to default to c

Please specify the location where cuDNN 7 library is installed. Refer to READI

Do you wish to build TensorFlow with TensorRT support? [y/N]:
No TensorRT support will be enabled for TensorFlow.

Please specify the NCCL version you want to use. If NCLL 2.2 is not installed

Please specify a list of comma-separated Cuda compute capabilities you want to
You can find the compute capability of your device at: https://developer.nvidi
Please note that each additional compute capability significantly increases yo
build time and binary size. [Default is: 3.5,7.0] 6.1

Do you want to use clang as CUDA compiler? [y/N]:
nvcc will be used as CUDA compiler.

Please specify which gcc should be used by nvcc as the host compiler. [Defaul

Do you wish to build TensorFlow with MPI support? [y/N]:
No MPI support will be enabled for TensorFlow.

Please specify optimization flags to use during compilation when bazel option

Would you like to interactively configure ./WORKSPACE for Android builds? [y/N
Not configuring the WORKSPACE for Android builds.

Preconfigured Bazel build configs. You can use any of the below by adding "--c
    --config=mkl            # Build with MKL support.
    --config=monolithic     # Config for mostly static monolithic build.
Configuration finished
```

Configuration options

GPU support

For GPU support (https://www.tensorflow.org/install/gpu), set `cuda=Y` during configuration and specify the versions of CUDA and cuDNN. If your system has multiple versions of CUDA or cuDNN installed, explicitly set the version instead of relying on the default. `./configure` creates symbolic links to your system's CUDA libraries—so if you update your CUDA library paths, this configuration step must be run again before building.

Optimizations

For compilation optimization flags, the default (`-march=native`) optimizes the generated code for your machine's CPU type. However, if building TensorFlow for a different CPU type, consider a more specific optimization flag. See the GCC manual (https://gcc.gnu.org/onlinedocs/gcc-4.5.3/gcc/i386-and-x86_002d64-Options.html) for examples.

Preconfigured configurations

There are some preconfigured build configs available that can be added to the `bazel build` command, for example:

- `--config=mkl` —Support for the Intel® MKL-DNN (https://github.com/intel/mkl-dnn).
- `--config=monolithic` —Configuration for a mostly static, monolithic build.
- `--config=v1` —Build TensorFlow 1.x instead of 2.x.

Starting with TensorFlow 1.6, binaries use AVX instructions which may not run on older CPUs.

Build the pip package

TensorFlow 2.x

tensorflow:master repo has been updated to build 2.x by default. Install Bazel (https://docs.bazel.build/versions/master/install.html) and use `bazel build` to create the TensorFlow package.

```
el build //tensorflow/tools/pip_package:build_pip_package
```

For GPU support, enable CUDA with **`cuda=Y`** during the **`./configure`** stage.

TensorFlow 1.x

To build the 1.x version of TensorFlow from master, use **`bazel build --config=v1`** to create a TensorFlow 1.x package.

```
el build --config=v1 //tensorflow/tools/pip_package:build_pip_package
```

CPU-only

Use **`bazel`** to make the TensorFlow package builder with CPU-only support:

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

GPU support

To make the TensorFlow package builder with GPU support:

```
el build --config=opt --config=cuda //tensorflow/tools/pip_package:build_pip_
```

Bazel build options

See the Bazel command-line reference (https://docs.bazel.build/versions/master/command-line-reference.html) for build options (https://docs.bazel.build/versions/master/command-line-reference.html#build-options).

Building TensorFlow from source can use a lot of RAM. If your system is memory-constrained, limit Bazel's RAM usage with: **`--local_ram_resources=2048`**.

The official TensorFlow packages (https://www.tensorflow.org/install/pip) are built with GCC 4 and use the older ABI. For GCC 5 and later, make your build compatible with the older ABI using: **`--cxxopt="-D_GLIBCXX_USE_CXX11_ABI=0"`**. ABI compatibility ensures that custom

ops built against the official TensorFlow package continue to work with the GCC 5 built package.

Build the package

The **`bazel build`** command creates an executable named **`build_pip_package`**—this is the program that builds the **`pip`** package. Run the executable as shown below to build a **`.whl`** package in the **`/tmp/tensorflow_pkg`** directory.

To build from a release branch:

```
azel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg
```

To build from master, use **`--nightly_flag`** to get the right dependencies:

```
azel-bin/tensorflow/tools/pip_package/build_pip_package --nightly_flag /tmp/t
```

Although it is possible to build both CUDA and non-CUDA configurations under the same source tree, it's recommended to run **`bazel clean`** when switching between these two configurations in the same source tree.

Install the package

The filename of the generated **`.whl`** file depends on the TensorFlow version and your platform. Use **`pip install`** to install the package, for example:

```
 install /tmp/tensorflow_pkg/tensorflow-version-tags.whl
```

ss: TensorFlow is now installed.

Docker Linux builds

TensorFlow's Docker development images are an easy way to set up an environment to build Linux packages from source. These images already contain the source code and dependencies required to build TensorFlow. See the TensorFlow Docker guide (https://www.tensorflow.org/install/docker) for installation and the list of available image tags (https://hub.docker.com/r/tensorflow/tensorflow/tags/).

CPU-only

The following example uses the **`:devel`** image to build a CPU-only Python 2 package from the latest TensorFlow source code. See the Docker guide (https://www.tensorflow.org/install/docker) for available TensorFlow **`-devel`** tags.

Download the latest development image and start a Docker container that we'll use to build the *pip* package:

```
ker pull tensorflow/tensorflow:devel
ker run -it -w /tensorflow -v $PWD:/mnt -e HOST_PERMS="$(id -u):$(id -g)" \
ensorflow/tensorflow:devel bash

t pull  # within the container, download the latest source code
```

The above **`docker run`** command starts a shell in the **`/tensorflow`** directory—the root of the source tree. It mounts the host's current directory in the container's **`/mnt`** directory, and passes the host user's information to the container through an environmental variable (used to set permissions—Docker can make this tricky).

Alternatively, to build a host copy of TensorFlow within a container, mount the host source tree at the container's **`/tensorflow`** directory:

```
ker run -it -w /tensorflow -v /path/to/tensorflow:/tensorflow -v $PWD:/mnt \
e HOST_PERMS="$(id -u):$(id -g)" tensorflow/tensorflow:devel bash
```

With the source tree set up, build the TensorFlow package within the container's virtual environment:

1. Configure the build—this prompts the user to answer build configuration questions.
2. Build the tool used to create the *pip* package.
3. Run the tool to create the *pip* package.

4. Adjust the ownership permissions of the file for outside the container.

```
configure  # answer prompts or use defaults

zel build --config=opt //tensorflow/tools/pip_package:build_pip_package

bazel-bin/tensorflow/tools/pip_package/build_pip_package /mnt  # create packa

own $HOST_PERMS /mnt/tensorflow-version-tags.whl
```

Install and verify the package within the container:

```
p uninstall tensorflow  # remove current version

p install /mnt/tensorflow-version-tags.whl
 /tmp  # don't import from source directory
thon -c "import tensorflow as tf; print(tf.__version__)"
```

ss: TensorFlow is now installed.

On your host machine, the TensorFlow *pip* package is in the current directory (with host user permissions): **`./tensorflow-version-tags.whl`**

GPU support

Docker is the easiest way to build GPU support for TensorFlow since the *host* machine only requires the NVIDIA® driver (https://github.com/NVIDIA/nvidia-docker/wiki/Frequently-Asked-Questions#how-do-i-install-the-nvidia-driver) (the *NVIDIA® CUDA® Toolkit* doesn't have to be installed). See the GPU support guide (https://www.tensorflow.org/install/gpu) and the TensorFlow Docker guide (https://www.tensorflow.org/install/docker) to set up nvidia-docker (https://github.com/NVIDIA/nvidia-docker) (Linux only).

The following example downloads the TensorFlow **`:devel-gpu-py3`** image and uses **`nvidia-docker`** to run the GPU-enabled container. This development image is configured to build a Python 3 *pip* package with GPU support:

```
ker pull tensorflow/tensorflow:devel-gpu-py3
ker run --runtime=nvidia -it -w /tensorflow -v $PWD:/mnt -e HOST_PERMS="$(id
ensorflow/tensorflow:devel-gpu-py3 bash
```

Then, within the container's virtual environment, build the TensorFlow package with GPU support:

```
configure  # answer prompts or use defaults

zel build --config=opt --config=cuda //tensorflow/tools/pip_package:build_pip

bazel-bin/tensorflow/tools/pip_package/build_pip_package /mnt  # create packa

own $HOST_PERMS /mnt/tensorflow-version-tags.whl
```

Install and verify the package within the container and check for a GPU:

```
p uninstall tensorflow  # remove current version

p install /mnt/tensorflow-version-tags.whl
 /tmp  # don't import from source directory
thon -c "import tensorflow as tf; print(tf.contrib.eager.num_gpus())"
```

ss: TensorFlow is now installed.

Tested build configurations

Linux

CPU

Version	Python version	Compiler	Build tools
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1
tensorflow-1.14.0	2.7, 3.3-3.7	GCC 4.8	Bazel 0.24.1

tensorflow-1.13.1	2.7, 3.3-3.7	GCC 4.8	Bazel 0.19.2
tensorflow-1.12.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0
tensorflow-1.11.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0
tensorflow-1.10.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0
tensorflow-1.9.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.11.0
tensorflow-1.8.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.10.0
tensorflow-1.7.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.10.0
tensorflow-1.6.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0
tensorflow-1.5.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.8.0
tensorflow-1.4.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.5.4
tensorflow-1.3.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5
tensorflow-1.2.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5
tensorflow-1.1.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2
tensorflow-1.0.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2

GPU

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0
tensorflow_gpu-1.14.0	2.7, 3.3-3.7	GCC 4.8	Bazel 0.24.1	7.4	10.0
tensorflow_gpu-1.13.1	2.7, 3.3-3.7	GCC 4.8	Bazel 0.19.2	7.4	10.0
tensorflow_gpu-1.12.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9
tensorflow_gpu-1.11.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9
tensorflow_gpu-1.10.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9
tensorflow_gpu-1.9.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.11.0	7	9
tensorflow_gpu-1.8.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.10.0	7	9
tensorflow_gpu-1.7.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0	7	9

tensorflow_gpu-1.6.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0	7	9
tensorflow_gpu-1.5.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.8.0	7	9
tensorflow_gpu-1.4.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.5.4	6	8
tensorflow_gpu-1.3.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	6	8
tensorflow_gpu-1.2.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	5.1	8
tensorflow_gpu-1.1.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8
tensorflow_gpu-1.0.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8

macOS

CPU

Version	Python version	Compiler	Build tools
tensorflow-2.0.0	2.7, 3.3-3.7	Clang from xcode 10.1	Bazel 0.26.1
tensorflow-1.14.0	2.7, 3.3-3.7	Clang from xcode	Bazel 0.24.1
tensorflow-1.13.1	2.7, 3.3-3.7	Clang from xcode	Bazel 0.19.2
tensorflow-1.12.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.15.0
tensorflow-1.11.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.15.0
tensorflow-1.10.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.15.0
tensorflow-1.9.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.11.0
tensorflow-1.8.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.10.1
tensorflow-1.7.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.10.1
tensorflow-1.6.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.8.1
tensorflow-1.5.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.8.1
tensorflow-1.4.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.5.4
tensorflow-1.3.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.5
tensorflow-1.2.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.5
tensorflow-1.1.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.2

tensorflow-1.0.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.2

GPU

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow_gpu-1.1.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.2	5.1	8
tensorflow_gpu-1.0.0	2.7, 3.3-3.6	Clang from xcode	Bazel 0.4.2	5.1	8