# TensorFlow 正式版中文文档

书栈(BookStack.CN)

# 目　录

# 致谢

当前文档 《TensorFlow 正式版中文文档》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-06-05。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址: http://www.bookstack.cn/books/tensorflow_documents_zh

书栈官网: http://www.bookstack.cn

书栈开源: https://github.com/TruthHun

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

# 介绍



# 说明

TensorFlow 正式版本 V1.0 已经发布，api 较之前 V0.xx 版本发生了较大变化，Tutorial、HowTo 等文档也发生了很大变化。新的文档更加合理，对TensorFlow甚至机器学习的新手更加友好，更适合循序渐进的学习。

网络上流传较广的TensorFlow中文文档大多为 TensorFlow中文社区 的文档，翻译自 V0.5。

在此，选取版本 r1.1 的文档进行翻译，r1.1与r1.0的文档内容区别不大，结构做了一些调整。

---

具体内容查看 目录。

---

本项目同时更新于 GitBook 与 GitHub。

GitHub 项目地址: https://github.com/efeiefei/tensorflow_documents_zh/

GitBook 阅读地址: https://efeiefei.gitbooks.io/tensorflow_documents_zh/

欢迎联系，一起翻译！

# 来源(书栈小编注)

> *https://github.com/efeiefei/tensorflow_documents_zh/*

# 安装



# 安装TensorFlow

如下指南描述了如何安装TensorFlow的不同版本。

- 在 Ubuntu 上安装 TensorFlow
- 在 Mac OS X 上安装 TensorFlow
- 在 Windows 上安装 TensorFlow
- 从源码安装 TensorFlow

Python TensorFlow API 自版本 0.n 到 1.0 变化花了很多。如下指南描述了如何从老旧 TensorFlow 应用迁移到1.0版本。

迁移到 TensorFlow 1.0

如下指南描述了如何安装其他语言的TensorFlow库。这些API是为了在应用中使用TensorFlow模型，所以并不如Python API一样具有扩展性。

- 为 Java 安装 TensorFlow
- 为 C 安装 TensorFlow
- 为 GO 安装 TensorFlow

# 在 Ubuntu 上安装 TensorFlow



# 在 Ubuntu 上安装 TensorFlow

这篇指南描述了如何在 Ubuntu 上安装 TensorFlow。这些实例也可能在其他 Linux 版本生效，但我们只在 Ubuntu 14.04 及更高的版本上测试过。

## 确定 TensorFlow 版本

如下之中选择一种来安装：

- 只支持 **CPU** 的 **TensorFlow**。如果你的系统不支持 NVIDIA® GPU，你必须安装这个版本。这个版本的 TensorFlow 通常安装起来比较简单（一般 5 到 10分钟），所以即使你拥有 NVIDIA GPU，我们也推荐首先安装这个版本。
- 支持 **GPU** 的 **TensorFlow**. TensorFlow 在 GPU 上通常比在 CPU 上的执行的更快。所以如果你有符合如下要求的 NVIDIA® GPU 并且需要注重性能，可以随后安装这个版本。

### GPU support TensorFlow 的 NVIDIA 需求

需要事先安装如下 NVIDIA 软件。

- CUDA® Toolkit 8.0. 详见 NVIDIA's documentation。确保按照文档中描述的将 Cuda 相关路径加入到 `LD_LIBRARY_PATH` 环境变量中。
- CUDA Toolkit 8.0 相关的 NVIDIA 驱动。

- cuDNN v5.1。详见 NVIDIA's documentation。确保创建了 `CUDA_HOME` 环境变量。
- CUDA Compute Capability 3.0 或更高的 GPU 芯片。支持的 GPU 芯片详见 NVIDIA documentation 。

- libcupti-dev 库，该库提供了高级的性能支持，按如下命令安装：

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

如果含有上述库但版本较老，先升级。如果不能升级，操作如下：

- 从源码安装 TensorFlow
- 安装或升级至少如下版本：
    - CUDA toolkit 7.0 或更高
    - cuDNN v3 或更高
    - CUDA Compute Capability 3.0 或更高的 GPU 芯片。

# 确定如何安装 TensorFlow

有如下选择：

- virtualenv
- "native" pip
- Docker
- Anaconda

推荐 **virtualenv** 安装
（略过四种方法的说明，自行查找）

# virtualenv 安装

步骤如下：

1. 安装 pip 及 virtualenv：

```
$ sudo apt-get install python-pip python-dev python-virtualenv
```

2. 建立 virtualenv 环境：

```
$ virtualenv --system-site-packages targetDirectory
```

`targetDirectory` 指明了 virtualenv 的位置。

3. 激活 virtualenv 环境：

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

如上操作会将提示符更改为如下：

```
(tensorflow)$
```

4. 如下命令中选取一个安装 TensorFlow:

```
(tensorflow)$ pip install –upgrade tensorflow      # for Python 2.7
(tensorflow)$ pip3 install –upgrade tensorflow     # for Python 3.n
(tensorflow)$ pip install –upgrade tensorflow-gpu  # for Python 2.7 and GPU
(tensorflow)$ pip3 install –upgrade tensorflow-gpu # for Python 3.n and GPU
```

上述命令成功则跳过步骤5，否则执行步骤5。

5. (可选) 如果步骤4失败 (通常因为 pip 版本小于 8.1):

```
(tensorflow)$ pip install --upgrade TF_PYTHON_URL   # Python 2.7
(tensorflow)$ pip3 install --upgrade TF_PYTHON_URL  # Python 3.N
```

`TF_PYTHON_URL` 指定了 python tensorflow 的包的地址。 `TF_PYTHON_URL` 依赖于操作系统、Python 版本、GPU 支持，从 这里 找到合适的URL。如，安装 TensorFlow for Linux, Python 2.7、CPU-only，使用如下命令:

```
(tensorflow)$ pip3 install --upgrade \
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.1.0-cp34-cp34m-linux_x86_64.whl
```

如果遇到安装问题，详见 常见安装问题.

## 下一步

安装完毕之后： 验证安装结果。

注意，每次使用 TensorFlow 前需要先激活 virtualenv 环境，使用如下命令:

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

激活之后可从当前 shell 执行命令，此时提示符变成如下:

```
(tensorflow)$
```

使用完毕 TensorFlow 可用 `deactivate` 命令退出当前 virtualenv 环境:

```
(tensorflow)$ deactivate
```

此时提示符将会返回到默认提示符 （在 `PS1` 环境变量中定义）。

## 卸载 TensorFlow

卸载 TensorFlow，删除相关目录即可:

```
$ rm -r targetDirectory
```

# 原生 pip 安装

注意： setup.py 的 REQUIRED_PACKAGES 部分 列出了 pip 将会安装或者升级的 TensorFlow 包。

## 前提: Python 及 Pip

Python 已经在 Ubuntu 中自动安装了。花谢时间确定（ `python -v` ）你的操作系统中含有如下 Python 版本中的一个:

- Python 2.7
- Python 3.3+

Pip 或 pip3 通常已经在 Ubuntu 中安装。花些时间确定（ `pip -V` 或 `pip3 -V` ）已经安装。强烈建议使用 8.1 或更高的版本。如果 8.1 或更高的版本没有安装，使用如下命令安装或升级到最新 pip 版本:

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

## 安装 TensorFlow

假定已经安装了如上必要软件，如下步骤安装 TensorFlow:

1. 如下命令之一安装:

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

   如上命令执行完毕，可验证安装结果.

2. （可选）如果步骤1失败，使用如下命令安装:

   ```
   $ sudo pip  install --upgrade TF_PYTHON_URL   # Python 2.7
   $ sudo pip3 install --upgrade TF_PYTHON_URL   # Python 3.N
   ```

   `TF_PYTHON_URL` 指定了 python tensorflow 的包的地址。 `TF_PYTHON_URL` 依赖于操作系统、Python 版本、GPU 支持，从 这里 找到合适的URL。
   例如，安装 TensorFlow for Linux, Python 3.4、CPU-only，使用如下命令:

   ```
   $ sudo pip3 install --upgrade \
   https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.1.0-cp34-cp34m-linux_x86_64.whl
   ```

   该步骤失败查询 常见安装问题.

## 下一步

安装完毕之后： 验证安装结果。

## 卸载 TensorFlow

如下命令卸载:

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

# Docker 安装

如下步骤通过 Docker 安装 TensorFlow:

1. 按描述在你的机器上安装 Docker
   Docker documentation.
2. 可选，建立名为 `docker` 的用户组以便不通过 sudo 来登陆 container，
   Docker documentation.
   （如果省略该步骤，每次启动 Docker 都需要使用 sudo。）
3. 为安装支持 GPU 的 TensorFlow 版本，需要首先
   安装 nvidia-docker
4. 启动含有
   TensorFlow binary images.
   之一的 Docker 容器。

本章节的其余部分描述了如何启动一个 Docker 容器。

## CPU-only

使用如下命令启动一个 CPU-only Docker 容器:

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

其中:

- *-p hostPort:containerPort* 可选。
  如果计划从 shell 执行 TensorFlow 程序，忽略该选项。
  如果计划作为 `Jupyter notebooks` 执行 TensorFlow 程序，设定
  *hostPort* 及 *containerPort*
  均为 8888。如果计划在容器中启动 TensorBoard，
  添加第二个 `-p` 参数，设定 *hostPort* 及 *containerPort*
  均为 6006.

- *TensorFlowCPUImage* 是必须的. 它指定了使用的容器，如下选项中选取一个:

  - gcr.io/tensorflow/tensorflow，TensorFlow CPU 镜像。
  - gcr.io/tensorflow/tensorflow:latest-devel，最新的
    TensorFlow CPU 镜像外加源代码。
  - gcr.io/tensorflow/tensorflow:*version*，指定版本（如1.0.1）。
  - gcr.io/tensorflow/tensorflow:*version*-devel，指定版本外加源代码。

  gcr.io 是 Google 的容器仓库。注意一些镜像同样可从
  dockerhub获取。

例如，如下命令在一个容器中启动最新的 TensorFlow CPU 镜像，你可以在 shell 中执行 TensorFlow 程序:

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

如下命令同样在一个容器中启动最新的 TensorFlow CPU 镜像。
但是在该容器中，你可以在 `Jupyter notebook` 中执行 TensorFlow 程序：

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

Docker 会在你第一次启动 TensorFlow 镜像时下载它。

## GPU support

安装支持 GPU 的 TensorFlow 之前，确保你的系统满足
NVIDIA software requirements。To launch a Docker container
通过如下命令，启动一个支持 GPU 的 TensorFlow 的 Docker 容器

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

其中：

- *-p hostPort:containerPort* 可选。
  如果计划从 shell 执行 TensorFlow 程序，忽略该选项。
  如果计划作为 `Jupyter notebooks` 执行 TensorFlow 程序，设定
  *hostPort* 及 *containerPort*
  均为 8888。
- *TensorFlowCPUImage* 是必须的. 它指定了使用的容器，如下选项中选取一个：
  - gcr.io/tensorflow/tensorflow:latest-gpu，最新 TensorFlow GPU 镜像。
  - gcr.io/tensorflow/tensorflow:latest-devel-gpu，最新 TensorFlow GPU 镜像外加源代码。
  - gcr.io/tensorflow/tensorflow:*version*-gpu，指定版本的 TensorFlow GPU 镜像。
  - gcr.io/tensorflow/tensorflow:*version*-devel-gpu，指定版本的 TensorFlow GPU 镜像外加源代码。

我们推荐安装一个 `最新` 版。如下命令可以在 Docker 容器中启动一个最新版本 TensorFlow GPU 镜像，
你可以在其 shell 中执行TensorFlow 程序：

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

如下命令同样在一个容器中启动最新的 TensorFlow GPU 镜像。
但是在该容器中，你可以在 `Jupyter notebook` 中执行 TensorFlow 程序：

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

如下命令启动一个老版本的 TensorFlow:

```
$ nvidia-docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow:0.12.1-gpu
```

Docker 会在你第一次启动 TensorFlow 镜像时下载它。详情

TensorFlow docker readme.

## 下一步

安装完毕之后： 验证安装结果。

# Anaconda 安装

按照如下步骤在 Anaconda 环境中安装 TensorFlow:

1. 按说明下载并安装 Anaconda:
   Anaconda download site

2. 建立一个 conda 环境，命名为 tensorflow，以便运行某个 Python 版本:

```
$ conda create -n tensorflow
```

3. 激活 anaconda 环境:

```
$ source activate tensorflow
(tensorflow)$  # 你的提示符应变化
```

4. 在你的 conda 环境中安装 TensorFlow:

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

   其中 `TF_PYTHON_URL` 是
   TensorFlow Python 包地址.
   比如: 如下命令可以为 Python 3.4 安装 CPU-only 版本的 TensorFlow:

```
(tensorflow)$ pip install --ignore-installed --upgrade \
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.1.0-cp34-cp34m-linux_x86_64.whl
```

# 验证安装结果

按如下操作验证 TensorFlow 安装结果:

1. 确保准备环境完备
2. 执行一个简短的 TensorFlow 程序

## 准备环境

If you installed on native pip, virtualenv, or Anaconda, then
do the following:
如果通过原生 pip、virtualenv、Anaconda 安装，做如下操作:

1. 启动一个 terminal
2. 如果通过 virtualenv 或 Anaconda 安装，激活容器
3. 如果你安装了 TensorFlow 源码，定位到不含源码的任一目录中

如果通过 Docker 安装，启动一个可以通过 bash 操作的 Docker 容器：

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

## 执行一个简短的 TensorFlow 程序

在 shell 中调用 Python：

```
$ python
```

在 Python 交互式环境中输入如下命令：

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

如果系统输出如下，则安装成功：

```
Hello, TensorFlow!
```

如果你新接触 TensorFlow，参考初识 TensorFlow进行下一步学习。

如果系统输出错误信息而非欢迎信息，查看常见安装问题。

## 常见安装问题

我们依靠 Stack Overflow 来编写 TensorFlow 安装问题及解决方案的文档。
如下表格包含了 Stack Overflow 上比较常见的安装问题的连接。
如果你遇到了不在列表中的新的错误信息或者其他安装问题，请在 Stack Overflow 上搜索。
如果搜索不到，请在 Stack Overflow 上提出一个新的问题，并打上 `tensorflow` 的标签。

| Stack Overflow Link | Error Message |
| --- | --- |
| 36159194 | `ImportError: libcudart.so.Version: cannot open shared object file: No such file or directory` |
| 41991101 | `ImportError: libcudnn.Version: cannot open shared object file: No such file or directory` |
| 36371137 and here | `libprotobuf ERROR google/protobuf/src/google/protobuf/io/coded_stream.cc:207] A protocol message was rejected because it was too big (more than 67108864 bytes). To increase the limit (or to disable these warnings), see CodedInputStream::SetTotalBytesLimit() in google/protobuf/io/coded_stream.h.` |
| 35252888 | `Error importing tensorflow. Unless you are using bazel, you should not try to import tensorflow from its source directory; please exit the tensorflow source tree, and relaunch your python interpreter from there.` |
| 33623453 | `IOError: [Errno 2] No such file or directory: '/tmp/pip-o6Tpui-build/setup.py'` |

| | |
|---|---|
| 42006320 | `ImportError: Traceback (most recent call last):`<br>`File "…/tensorflow/core/framework/graph_pb2.py", line 6, in`<br>`from google.protobuf import descriptor as _descriptor`<br>`ImportError: cannot import name 'descriptor'` |
| 35190574 | `SSLError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed` |
| 42009190 | `Installing collected packages: setuptools, protobuf, wheel, numpy, tensorflow`<br>`Found existing installation: setuptools 1.1.6`<br>`Uninstalling setuptools-1.1.6:`<br>`Exception:`<br>`…`<br>`[Errno 1] Operation not permitted:`<br>`'/tmp/pip-a1DXRT-uninstall/…/lib/python/_markerlib'` |
| 36933958 | `…`<br>`Installing collected packages: setuptools, protobuf, wheel, numpy, tensorflow`<br>`Found existing installation: setuptools 1.1.6`<br>`Uninstalling setuptools-1.1.6:`<br>`Exception:`<br>`…`<br>`[Errno 1] Operation not permitted:`<br>`'/tmp/pip-a1DXRT-uninstall/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python/_markerlib'` |

## TensorFlow Python 包地址（`TF_PYTHON_URL`）

一些安装方法需要 TensorFlow Python 包，它的地址依赖于几个方面：

- 操作系统
- Python 版本
- CPU-only 还是 GPU support

### Python 2.7

CPU only:

```
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.1.0-cp27-none-linux_x86_64.whl
```

GPU support:

```
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.1.0-cp27-none-linux_x86_64.whl
```

注意 GPU 支持需要满足 NVIDIA 硬件需求以及在
GPU support TensorFlow 的 NVIDIA 需求
中描述的软件。

## Python 3.4

CPU only:

```
1. https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.1.0-cp34-cp34m-linux_x86_64.whl
```

GPU support:

```
1. https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.1.0-cp34-cp34m-linux_x86_64.whl
```

注意 GPU 支持需要满足 NVIDIA 硬件需求以及在
GPU support TensorFlow 的 NVIDIA 需求
中描述的软件。

## Python 3.5

CPU only:

```
1. https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.1.0-cp35-cp35m-linux_x86_64.whl
```

GPU support:

```
1. https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.1.0-cp35-cp35m-linux_x86_64.whl
```

注意 GPU 支持需要满足 NVIDIA 硬件需求以及在
GPU support TensorFlow 的 NVIDIA 需求
中描述的软件。

## Python 3.6

CPU only:

```
1. https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.1.0-cp36-cp36m-linux_x86_64.whl
```

GPU support:

```
1. https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.1.0-cp36-cp36m-linux_x86_64.whl
```

注意 GPU 支持需要满足 NVIDIA 硬件需求以及在
GPU support TensorFlow 的 NVIDIA 需求
中描述的软件。

# Protobuf pip 包 3.1

如果没有遇到和 protobuf pip 包相关的问题，你可以跳过这个部分。

注意： 如果你的 TensorFlow 运行缓慢，你可能遇到了一个 protobuf pip 包相关的问题。

TensorFlow pip 包依赖于 protobuf pip 包 3.1。
从 PyPI 下载的 protobuf 是 proto 序列化/反序列化的纯 Python 实现的库，
其速度比 C++ 实现慢**10-50**倍。Protobuf 支持二进制扩展，速度更快，基于 C++。
但该扩展无法在纯Python实现的 pip 包中获取。我们制作了包含该二进制扩展的protobuf pip 包。
如下命令可安装该自定义的 protobuf pip 包：

- for Python 2.7:

```
$ pip install --upgrade \
https://storage.googleapis.com/tensorflow/linux/cpu/protobuf-3.1.0-cp27-none-linux_x86_64.whl
```

- for Python 3.5:

```
$ pip3 install --upgrade \
https://storage.googleapis.com/tensorflow/linux/cpu/protobuf-3.1.0-cp35-none-linux_x86_64.whl
```

安装这个 protobuf 包会覆盖已经存在的 protobuf 包。
注意该二进制 pip 包已经支持大于 64MB 的问题，修复了如下问题：

```
[libprotobuf ERROR google/protobuf/src/google/protobuf/io/coded_stream.cc:207]
Protocol message 被拒绝，因为太大（大于 67108864 字节）。
为增大限制或禁用报警，
在 google/protobuf/io/coded_stream.h 中查看 CodedInputStream::SetTotalBytesLimit()
```

# 在 Windows 上安装 TensorFlow



## 在 Windows 上安装 TensorFlow

这篇指南描述了如何在 Windows 上安装 TensorFlow。

## 确定 TensorFlow 版本

如下之中选择一种来安装：

- 只支持 **CPU** 的 **TensorFlow**。如果你的系统不支持 NVIDIA® GPU，你必须安装这个版本。这个版本的 TensorFlow 通常安装起来比较简单（一般 5 到 10分钟），所以即使你拥有 NVIDIA GPU，我们也推荐首先安装这个版本。
- 支持 **GPU** 的 **TensorFlow**. TensorFlow 在 GPU 上通常比在 CPU 上的执行的更快。所以如果你有符合如下要求的 NVIDIA® GPU 并且需要注重性能，可以随后安装这个版本。

### GPU support TensorFlow 的 NVIDIA 需求

需要事先安装如下软件：

- CUDA® Toolkit 8.0。详见 NVIDIA's documentation。确保按照文档中描述的将 Cuda 相关路径加入到 `%PATH%` 环境变量中。
- CUDA Toolkit 8.0 相关的 NVIDIA 驱动。
- cuDNN v5.1。详见 NVIDIA's documentation。注意：cuDNN 通常与其他 CUDA DLLs 安装的位置不同。确保将 cuDNN 库的安装目录加入到了 `%PATH%` 中。
- CUDA Compute Capability 3.0 或更高的 GPU 芯片。支持的 GPU 芯片详见 NVIDIA documentation 。

如果上述软件版本较老，请将其升级到指定版本。

## 确定如何安装 TensorFlow

有如下选择：

- "native" pip
- Anaconda

原生 pip 直接在系统中安装 TensorFlow，而不使用虚拟环境。
因为原生 pip 安装没有使用独立的容器隔离开，所以可能干扰其他基于Python的安装。
不过，如果你理解 pip 和 Python 环境，原生 pip 安装通常只需要一个命令！
如果使用原生 pip 安装，用户可在任何目录中执行 TensorFlow 程序。

在 Anaconda 中，你可以通过 conda 创建一个虚拟环境。
然而，我们推荐使用 `pip install` 安装 TensorFlow，而非 `conda install` 。

注意：conda 包是社区支持而非官方支持。也就是说 TensorFlow 团队没有测试也没有管理过 conda 包。
使用这个包需要自行承担风险。

# 原生 pip 安装

如果如下版本的 Python 没有安装，先安装：

- Python 3.5.x from python.org

TensorFlow 在 Windows 上支持 Python 3.5.x。
注意 Python 3.5.x 使用 pip3，我们用 pip3 来安装 TensorFlow。

在 terminal 中输入如下命令安装只支持 CPU 的 TensorFlow：

```
C:\> pip3 install --upgrade tensorflow
```

安装支持 GPU 的 TensorFlow，使用如下命令：

```
C:\> pip3 install --upgrade tensorflow-gpu
```

# Anaconda 安装

**Anaconda** 安装是社区支持，而非官方支持

按照如下步骤在 Anaconda 环境中安装 TensorFlow：

1. 按说明下载并安装 Anaconda：
   Anaconda download site

2. 建立一个 conda 环境，命名为 tensorflow，以便运行某个 Python 版本：

   ```
   C:\> conda create -n tensorflow
   ```

3. 激活 anaconda 环境：

   ```
   C:\> activate tensorflow
   (tensorflow)C:\>  # 你的提示符应该发生变化
   ```

4. 在你的 conda 环境中安装只支持 CPU 的 TensorFlow（写在一行）：

   ```
   (tensorflow)C:\> pip install --ignore-installed --upgrade
   https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow-1.1.0-cp35-cp35m-win_amd64.whl
   ```

   安装支持 GPU 的 TensorFlow（写在一行）：

   ```
   (tensorflow)C:\> pip install --ignore-installed --upgrade
   https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-1.1.0-cp35-cp35m-win_amd64.whl
   ```

# 验证安装结果

启动 terminal。

如果通过 Anaconda 安装，激活 Anaconda 环境。

启动 Python:

```
$ python
```

在 Python 交互式环境中输入

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

如果系统输出如下，则安装成功:

```
Hello, TensorFlow!
```

如果你新接触 TensorFlow，参考初识 TensorFlow进行下一步学习。

如果系统输出错误信息而非欢迎信息，查看常见安装问题。

# 常见安装问题

我们依靠 Stack Overflow 来编写 TensorFlow 安装问题及解决方案的文档。
如下表格包含了 Stack Overflow 上比较常见的安装问题的连接。
如果你遇到了不在列表中的新的错误信息或者其他安装问题，请在 Stack Overflow 上搜索。
如果搜索不到，请在 Stack Overflow 上提出一个新的问题，并打上 `tensorflow` 的标签。

| Stack Overflow Link | Error Message |
| --- | --- |
| 41007279 | `[…\stream_executor\dso_loader.cc] Couldn't open CUDA library nvcuda.dll` |
| 41007279 | `[…\stream_executor\cuda\cuda_dnn.cc] Unable to load cuDNN DSO` |
| 42006320 | `ImportError: Traceback (most recent call last):`<br>`File "…\tensorflow\core\framework\graph_pb2.py", line 6, in`<br>`from google.protobuf import descriptor as _descriptor`<br>`ImportError: cannot import name 'descriptor'` |
| 42011070 | `No module named "pywrap_tensorflow"` |
| 42217532 | `OpKernel ('op: "BestSplits" device_type: "CPU"') for unknown op: BestSplits` |
| 43134753 | `The TensorFlow library wasn't compiled to use SSE instructions` |

# 部署



# Deploy

This section focuses on deploying real-world models. It contains the following documents:

- @{$distributed$Distributed TensorFlow}, which explains how to create a cluster of TensorFlow servers.
- @{$tfserve$TensorFlow Serving}, which describes TensorFlow Serving—an open-source serving system for machine learning models. This document provides a short introduction to TensorFlow Serving; the bulk of the documentation about TensorFlow Serving is in a separate website.
- @{$hadoop$How to run TensorFlow on Hadoop}, which has a highly self-explanatory title.

# 延伸



# Extend

This section explains how developers can add functionality to TensorFlow's capabilities. Begin by reading the following architectural overview:

- @{$architecture$TensorFlow Architecture}

The following guides explain how to extend particular aspects of TensorFlow:

- @{$adding_an_op$Adding a New Op}, which explains how to create your own operations.
- @{$add_filesys$Adding a Custom Filesystem Plugin}, which explains how to add support for your own shared or distributed filesystem.
- @{$new_data_formats$Custom Data Readers}, which details how to add support for your own file and record formats.
- @{$estimators$Creating Estimators in tf.contrib.learn}, which explains how to write your own custom Estimator. For example, you could build your own Estimator to implement some variation on standard linear regression.

Python is currently the only language supported by TensorFlow's API stability promises. However, TensorFlow also provides functionality in C++, Java, and Go, plus community support for Haskell and Rust. If you'd like to create or develop TensorFlow features in a language other than these languages, read the following guide:

- @{$language_bindings$TensorFlow in Other Languages}

To create tools compatible with TensorFlow's model format, read the following guide:

- @{$tool_developers$A Tool Developer's Guide to TensorFlow Model Files}