

# Linux 服务器安装笔记

---

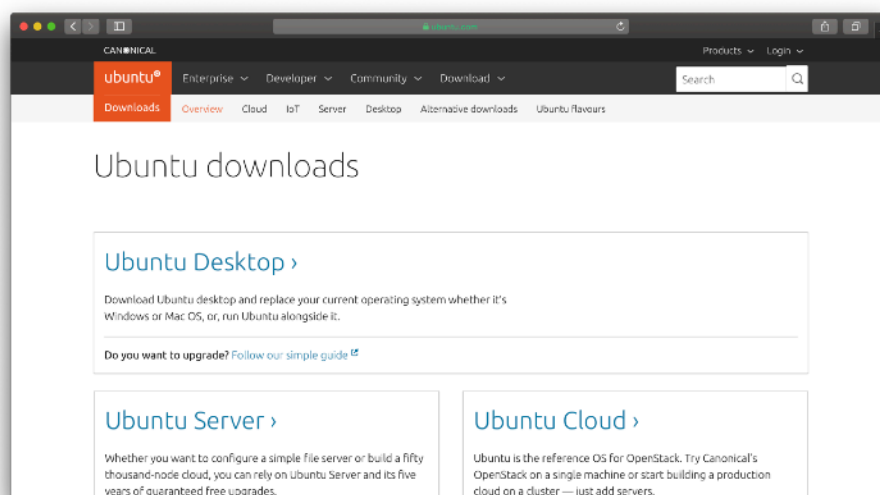
安装机型：曙光服务器；安装系统：Ubuntu-18.04-Server版

硬件准备：服务器，U盘（存储大于8G）

软件准备：Windows系统下的 UltraISO（制作启动盘，也可以用其他刻录工具，例如老毛桃等）

## - 下载Ubuntu镜像文件、制作启动盘

- 官网 <https://www.ubuntu.com/download> 可以选择下载桌面版或服务器版，服务器版没有图形界面。



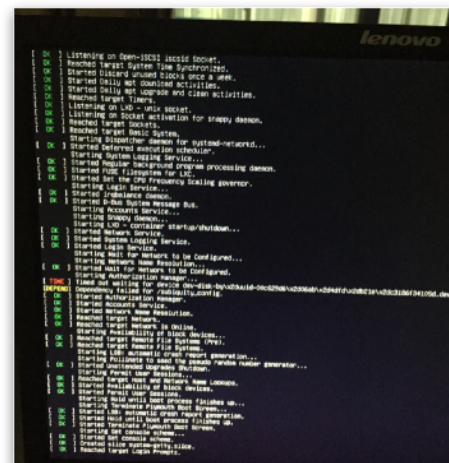
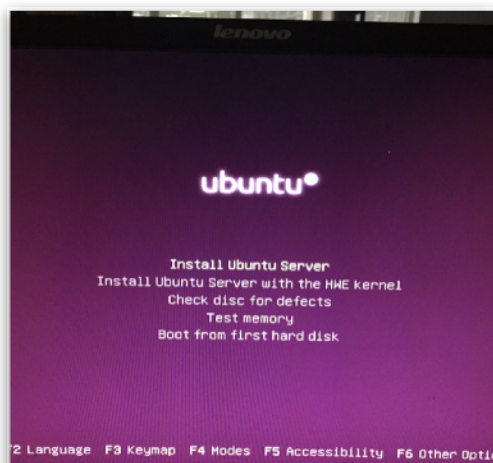
- 下载完所需的版本后，将得到一个.iso文件。这里我下载了Ubuntu-18.04.2-live-amd64.iso 其中live表示安装时需要外接网络，安装比较方便。
- 下载好之后，在Windows电脑上插入空的U盘，我们打开刻录软件UltraISO，点击打开按钮，选择我们下载好的镜像文件，并选择我们要刻录的U盘，点击启动，选择写入硬盘映像，在之后的对话框中我们选择写入方式为RAW，还有其他的写入方式，但对于理仁楼的老机器而言必须选择RAW方式，否则安装时将会提示“Load liblinux32.so Failure” 具体各种写入方式有何不同可以参见

<https://blog.csdn.net/bjkuailcabc/article/details/79913456>

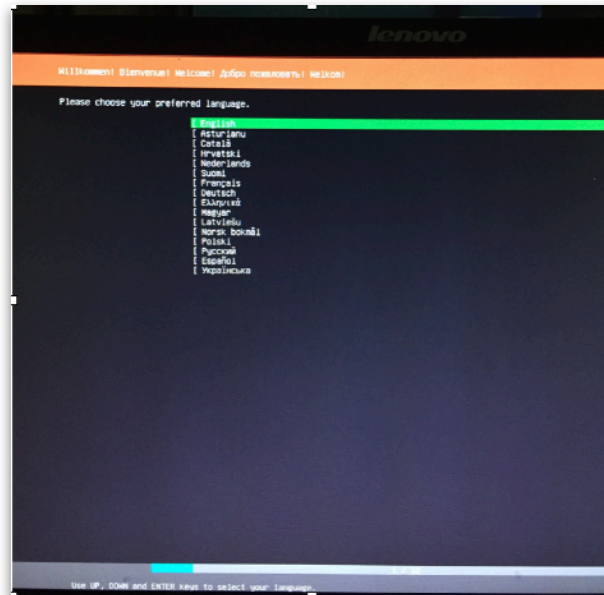
- 最后点击写入，显示刻录成功！此时我们就制作好了U盘启动盘。

## - 安装系统

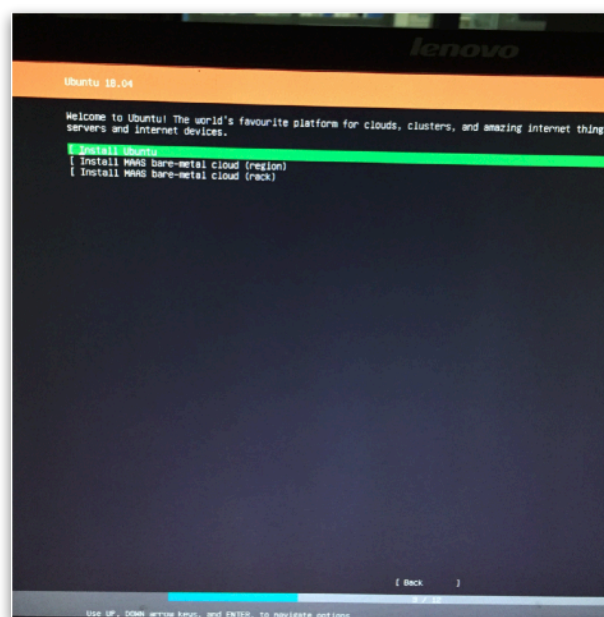
- 将U盘启动盘插入服务器的USB接口，并检查键盘，鼠标，网线以及点路线是否连接正确，确认后开机。
- 进入BIOS界面，这一步是针对服务器有系统的情况，如果服务器本身没有任何系统则无需进入BIOS界面，而是直接进入系统安装界面。对于曙光服务器，进入BIOS界面一般是在开机时持续按住ESC，F2，和Delete键，知道出现BIOS界面，进入界面后用键盘上的上下左右选择BOOT选项，将选项卡下方的优先选项选为我们刚刚插入的启动盘的名称，例如我用的是KINGSTON，那么选择带有KINGSTON字样的盘就可以了。选好后按F4保存退出，系统将以U盘为启动盘重启，进而进入安装界面。
- 当进入安装界面后，Ubuntu为下图所示出现此界面后，我们通过键盘选择Install Ubuntu Server，进行安装。屏幕将会出现各种安装进程的启动条目滚动。



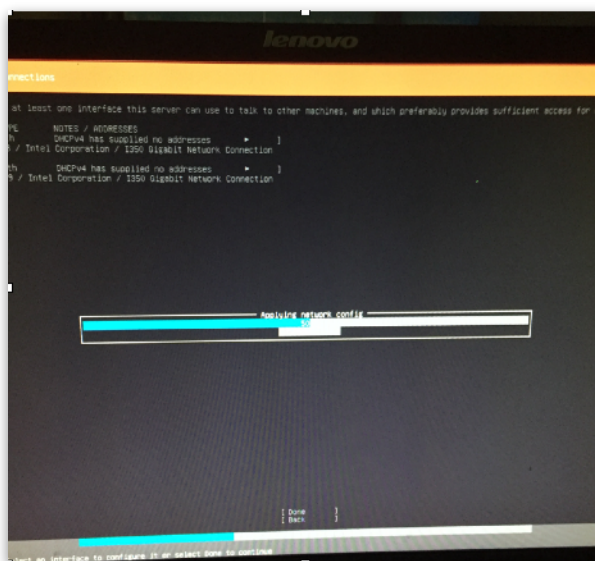
- 当滚动结束屏幕将会提示语言选择，我们可以通过键盘选择我们想要的语言，这里我选择了英文：



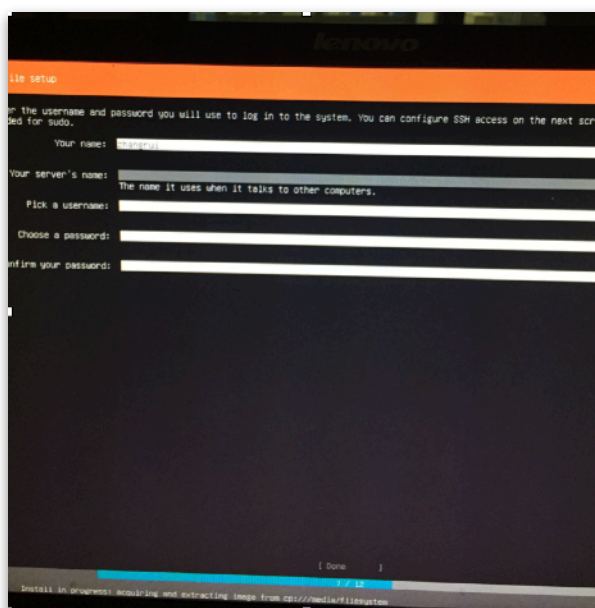
- 接下来选择所需的平台服务，我们选择“Install Ubuntu”即可：



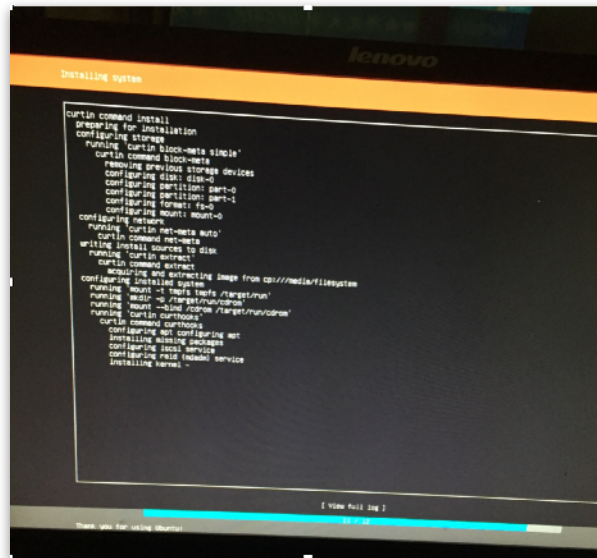
- 确认连接网络，我们直接按回车键即可：



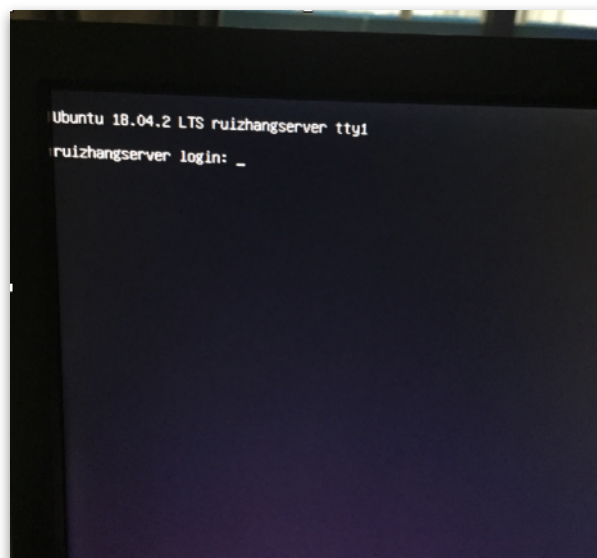
- 待安装程序联网认证后，将会提示我们创建用户名和密码，注意此时创建的用户默认具有管理员权限，但不是root，只是用户具有了sudoer权限。



- 创建完用户后，按回车键，将会启动内核安装程序：



- 安装完毕后系统将提示“Install Successfully !”然后自动重启，便进入到登陆界面，需要我们输入刚才创建好的用户名和密码，注意我安装的是Server版，因此只有黑色的界面，如果需要炫酷的玫红色用户图形界面，需要下载安装桌面版Ubuntu。



- 输入口令后，我们就进入到了刚才创建的用户界面了，我们可以在其中输入 `ifconfig` 命令来查看当前机器的网卡，一般曙光机器有两个网卡，我们要使用其



中能用的那个网卡，这些信息就用ifconfig命令来完成，可以看到现在有两个网卡，我们使用的是enp5sf0号网卡，且此时的IP地址为 10.10.15.137。

```
zhangrui@ruizhangserver:~$ ifconfig
enp5sf0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.15.137 netmask 255.255.255.0 broadcast 10.10.15.255
    inet6 fe80::225:90ff:fec4:2b28 prefixlen 64 scopeid 0x20<link>
    ether 00:25:90:c4:2b:28 txqueuelen 1000 (Ethernet)
    RX packets 47596 bytes 48274862 (48.2 MB)
    RX errors 0 dropped 541 overruns 0 frame 0
    TX packets 20606 bytes 1644118 (1.6 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device memory 0xdf920000-df93ffff

enp5sf1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:25:90:c4:2b:29 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device memory 0xdf900000-df91ffff

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 226 bytes 19060 (19.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 226 bytes 19060 (19.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

zhangrui@ruizhangserver:~$
```

- 至此，系统安装完成！

## - 系统配置

- 修改root密码。在服务器终端键入`sudo passwd`，此时提示输入新root密码，输入结束后回车，提示再次确认密码，输入后便修改成功。注意，root权限非常危险，密码只能是少部分超级管理员知道，这也是Unix的一条不成文的规定。添加用户是也应谨慎为其分配sudo权限，越少越好。一般是一人最多两人。
- SSH远程配置。对于这一版本的Ubuntu，在安装过程中默认已经开启SSH服务。我们需要在终端进行远程测试，打开Terminal，键入

`ssh yourusername@10.10.15.137`

- 进行远程连接，若提示输入password：则代表远程成功，ssh服务正常。
- 若提示“Connection refused！”或“Operation time out！”则证明ssh服务出现问题。这时候我们要做的是

- 第一，在终端“ping”网络，看能否“ping”通。例如在终端键入

`ping 10.10.15.137`

若发现丢包率为100%，则代表网络连接出现问题，需要排查网络，若接收包正常，丢包率几乎为零，则网络畅通，远程失败并非网络问题，进行下一步排查；

- 第二，在服务器终端检查ssh服务开启状态。在终端键入

```
sudo ps -e |grep ssh
```

来检查是否安装了ssh服务，若未安装则使用`sudo apt-get`命令安装，若已经安装，则键入`sudo service sshd stop`关闭ssh，稍后几秒钟再键入 `sudo service sshd start`重新开启ssh服务，不建议直接使用`service sshd restart`来重启ssh。操作结束后可以在pc终端再次远程测试，如果还不行，则需要检查服务器的防火墙是否关闭，由于我们并非商业用途，我们可以直接关闭防火墙，或者只开启对终端22号端口信任，但这样做有一定局限性，我们一般直接关闭防火墙。

- 第三，关闭防火墙。在服务器终端键入 `sudo ufw disable`，关闭后可以用命令 `sudo ufw status` 来查看是否关闭成功。如果还是远程失败，那可能是sshd\_config文件的配置问题。
- 第四，修改sshd\_config 配置文件（万不得已，不要进行这一步）。在服务器终端命令行`su`到root用户，注意当前处于root用户，具有最高权限，可以对系统进行任何操作，异常危险，一定要谨慎操作。`cd`进入到/etc/ssh/目录下，`vim`查看sshd\_config文件，找到PermitRootLogin参数，如果后面是no，则改为yes，（最好是不要在原来语句上直接修改，应将原来的那一行语句前面加上#注释掉，再另起一行键入更改后的语句！）`ctrl+c`退出修改，`:wq`保存退出，重启ssh服务，再次尝试远程，如果还是失败，则仍需仔细修改配置文件，详细配置文件可参考我Github上的修改好的配置文件。

<https://github.com/RuiZhangJerry/Install-Linux-public>

- 当能够成功远程服务器后，我们在机房的工作就结束了，现在可以在pc端的Terminal上进行后续软件包的安装。
- 软件包安装。
- 编译环境配置。我们需要安装gcc以及g++，python环境系统已经自带默认安装了，此版本Ubuntu安装的是python3. 安装gcc以及g++均可以使用Ubuntu包管理器：`apt-get`。
    - gcc/g++: `sudo apt-get install gcc; sudo apt-get install g++.`
    - numpy: `sudo apt-get install numpy;`
    - gdb Debug: `sudo apt-get install gdb;`
    - gsl: GNU科学计算包也可以用apt安装: `sudo apt-get install libgsl0-dev.`

- gfortran安装：在后续编译lapack等包时需要用gfortran编译，因此我们需要使用：`sudo apt-get install gfortran`.
- 测试。
  - 在用户主目录下 `mkdir` 创建一个文件夹，例如c++，然后 `cd` 进去，`touch` 创建一个.c文件，例如helloworld.c，`vi` 进入helloworld.c 键入显示hello word！代码，保存退出后，使用 `g++ helloworld.c`，编译结束 `ls` 查看，不出意外应该会有一个可执行文件a.out，然后 `./a.out` 运行，应该能在终端看到helloworld！此时g++/gcc安装成功。
  - numpy可以在python3中使用import numpy命令来测试。
  - 测试gdb时，我们可以在刚才的.c文件中定义一个指针，例如：`int *p`；然后直接访问：`cout<<*p<<endl`；此时运行时将会提示Segmentation Fault！然后我们键入 `gdb ./a.out` 回车，`run` 回车，`bt` 回车，应该能定位到我们访问未初始化的指针那条语句处。
  - 在我的GitHub中有gsl的测试代码，我们在编译该文件时要添加gsl的库文件链接指令，键入 `g++ -I/usr/include -L/usr/lib gsl_test.c -lgsl -lgslcblas` 即可，运行后如果报错，则说明环境变量未配置好。需要手动配置环境变量：`sudo vim /etc/profile` 最下面加入：`export LD_LIBRARY_PATH=/usr/local/lib: $LD_LIBRARY_PATH` 退出后使用 `source /etc/profile` 刷新即可。此时再编译运行gsl测试代码，就正常输出结果了。
- Lapack, Blas软件包的安装。可参见

<https://blog.csdn.net/wusiyuan163/article/details/49779353>

- Umfpack 计算包的安装。可参见

<http://faculty.cse.tamu.edu/davis/suitesparse.html>

在安装Lapack，Blas，Umfpack时最重要的一步是，在局部编译后生成了多个.o目标文件，我们需要手动用 `ar` 命令将这些.o文件链接成.a静态库文件，用于静态链接，即库文件liblapack.a, libblas.a, libumfpack.a我们一定要将这几个手动复制到usr/local/lib中，否则后续make程序的时候就要报未找到-llapack, -lblas, -lumfpack的链接错误. 关于Linux的文件系统以及编译原理可以参见

<https://www.cnblogs.com/ziyunlong/p/6023121.html>



- 本地计算包lib的编译。在上述包都安装好后，将本地的lib包上传至服务器上的某个文件夹中，例如我是放在c++中，然后cd到文件夹中，`make`编译包即可。注意，如果编译完成后没有报错，但后期测试时，编译报错umfpack包出现连接错误（如下图），则在lib包的外部Makefile中的`all`：后面加上`(cd interface; make)`，然后在

```
Undefined symbols for architecture x86_64:
  "UMFPACK_INTERFACE::umfpack_di_solve(Sparse*, Vector*, Vector*)", referenced from:
    SE_VariableCoefficientHELMHLOTZProblem::solve()      in cch2i8Vv.o
ld: symbol(s) not found for architecture x86_64
collect2: error: ld returned 1 exit status
make: *** [main] Error 1
```

`clean`：后面加上`(cd interface; make clean)`之后再编译整个包，完成后再来测试程序，问题解决。

- 测试。在我的GitHub中有两个代码一个是二维另一个是三维，可以全面测试lib能否正常使用。在编译之前需要对Makefile进行细微的调整，主要是lib的路径，比如我现在的是`WORK_DIR=/home/zhangrui/c++/lib`。其他的链接库不需改变。这台机器我已经编译好所有的包，可直接将我的账号下的Makefile复制到自己目录下的程序中，`make` 编译 `./main` 运行即可，不需要用户自己再去重新编译包安装软件了！

- python 及其相关软件包的安装：

- python 安装：一般的服务器均默认安装python，但有少数服务器版本的系统未安装，在终端键入 `python` 如果显示python版本号并进入python界面后，则已安装有python，反之终端会提示需要使用ubuntu的包管理器 `apt-get` 来安装python，此时我们需要先安装python。直接使用命令 `sudo apt-get install python`，输入管理员口令后便开始自动安装，如果安装过程为报错则安装成功。这时我们再键入`python`后便会显示所安装的版本号并进入python界面。

- pip安装：现在我们的系统中只是安装了基本的python，但python有非常多优秀的软件包供开发者使用，为了更高效的使用python我们需要安装一些相关的python源码包。为了方便管理和安装python第三方库，我们首先安装pip以及其依赖包。在终端键入 `sudo apt-get install python-pip python-dev`，安装完成后我们更新pip，键入 `sudo pip install -u pip`。等待安装结束即可。

- 科学计算包安装：主要使用numpy (矩阵计算)、scipy (科学计算)、matplotlib (2D, 3D 绘图) 以及 pandas等，这些包我们用 X 来统一表示，则其安装就在终端键入 `sudo pip install X` 即可。

Sunday, April 14, 2019

- 机器学习包，我们主要安装 TensorFlow 包。首先我们需要更改pip源码地址，以使下载速度更快，进入根目录: `cd ~`, `mkdir` 创建隐藏文件夹 `.pip`, (注: 文件夹前面带有 `.` 表示隐藏文件夹) `cd` 进入 `.pip` 文件夹, `touch` 创建文件 `pip.conf`, `vi` 查看 `pip.conf`, `i` 进入修改模式, 在其中输入

```
[global]
```

```
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
```

```
[install]
```

```
trust-host=mirrors.aliyun.com
```

然后`ctrl+c`退出修改模式, `:wq` 保存退出文档即可。接下来便可以安装tensorflow包。在终端键入 `sudo pip install tensorflow` 这是cpu版, 还可以安装 gpu 版本, 即键入: `sudo pip install tensorflow-gpu` 等待安装即可, 若无报错即为安装成功。

- 注: 上述安装均基于python 2.7 版本。如果安装python 3.x 版本可参见

<https://blog.csdn.net/y1250056491/article/details/78670710/>

- python 包测试。在上述包安装完成后, 我们对其进行测试, 首先我们进入到python界面, 即在终端键入 `python` 回车, 在python界面中我们首先检测科学计算包 X, 键入 `import X` 回车, 如果没有报错即可。再测试tensorflow包, 在python界面键入 如下代码进行测试:

```
import tensorflow as tf

session = tf.Session()

a = tf.constant(111)

b = tf.constant(222)

print(session.run(a+b))
```

在终端能看到正确运行返回结果 333 即可。

```
shangrui@ruishangserver: $ python
Python 2.7.15rc1 (default, Nov 12 2018, 14:31:15)
[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> session = tf.Session()
2019-05-18 01:52:18.243282: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2000045000 Hz
2019-05-18 01:52:18.249076: I tensorflow/compiler/xla/service/service.cc:150] XLA service 0x558f9e6e5e40 executing
computations on platform Host. Devices:
2019-05-18 01:52:18.249174: I tensorflow/compiler/xla/service/service.cc:158] StreamExecutor device (0): <undefin
ed>, <undefined>
>>> a = tf.constant(111)
>>> b = tf.constant(222)
>>> print(session.run(a+b))
333
>>>
```