# 安装anaconda

**查看版本 python -V (大写的)**

1.确定服务器是32位的还是64位的

$ file /bin/ls

* 1

输出为:

/bin/ls: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked(uses shared libs), for GNU/Linux 2.6.18, stripped

* 1

根据服务器的位数选择Anaconda的版本，我选择了Linux 64位的。   
2.下载Anaconda   
从[https://www.continuum.io/downloads/](https://www.continuum.io/downloads/" \t "_blank)下载自己需要的Anaconda的版本。   
3.安装   
进入Anaconda所在的目录，执行下面的命令：

$ bash Anaconda2-4.4.0-Linux-x86\_64.sh（根据具体安装包修改）

* 1

此后依照提示操作即可（最简便的方式就是，让ENTER就ENTER，问yes或no，输入yes，三个yes分别代表同意license、使用默认的安装路径、自动向.bashrc写入路径)   
4. 使.bashrc生效   
此时Anaconda并未安装完成，若在终端输入python将会发现依然是Centos自带的python版本，这是因为.bashrc的更新还没有生效，执行下述命令使其生效即可。

**$ source ~/.bashrc**

# **2. 在系统级修改 Python 版本**

我们可以使用 update-alternatives 来为整个系统更改 Python 版本。以 root 身份登录，首先罗列出所有可用的 python 替代版本信息：

**[python]** [view plain](https://blog.csdn.net/u011534057/article/details/51615193) [copy](https://blog.csdn.net/u011534057/article/details/51615193)

1. # update-alternatives --list python
2. update-alternatives: error: no alternatives **for** python

如何查看当前python的工作目录呢，只需要按如下操作就可以

1. **import** sys
2. sys.executable

'/root/anaconda3/bin/python'

如果出现以上所示的错误信息，则表示 Python 的替代版本尚未被 update-alternatives 命令识别。想解决这个问题，我们需要更新一下替代列表，将 和 python3.4 放入其中。

**[python]** [view plain](https://blog.csdn.net/u011534057/article/details/51615193) [copy](https://blog.csdn.net/u011534057/article/details/51615193)

1. update-alternatives --install /usr/bin/python python /root/anaconda3/bin/python（只改这个安装路径）**1**（这个1别漏了）
2. update-alternatives: using /usr/bin/python3.4 to provide /usr/bin/python (python) **in** auto mode

--install 选项使用了多个参数用于创建符号链接。最后一个参数指定了此选项的优先级，如果我们没有手动来设置替代选项，那么具有最高优先级的选项就会被选中。这个例子中，我们为 /usr/bin/python3.4 设置的优先级为2，所以update-alternatives 命令会自动将它设置为默认 Python 版本。

# 3 jupyter notebook

**1. 登陆远程服务器**

**2. 生成配置文件**

jupyter notebook --generate-config

**3. 生成密码**

打开ipython，创建一个密文的密码：

In [1]: from notebook.auth import passwd

In [2]: passwd()

Enter password:

Verify password:

Out[2]: 'sha1:ce23d945972f:34769685a7ccd3d08c84a18c63968a41f1140274'

In [3]: exit() 退出

**4. 修改默认配置文件**

vim ~/.jupyter/jupyter\_notebook\_config.py   
进行如下修改： (查找时，按字母顺序找)

c.NotebookApp.ip='\*'

c.NotebookApp.open\_browser = False

c.NotebookApp.password = u'sha:ce...刚才复制的那个密文'

c.NotebookApp.port =8882 #随便指定一个端口

c.NotebookApp.default\_url = '/root'  # 修改jupyter打开默认路径

**5. 启动jupyter notebook：**

$jupyter notebook

后台启动

nohup jupyter notebook --allow-root &

6.jupyter选择conda 环境

* 安装 conda install ipykernel
* 激活环境 source activate 环境名称
* python -m ipykernel install --user --name 环境名称 --display-name(紧密连接没有空格) "Python (环境名称)"

**7. 远程访问**

由于本人在开启jupyter时，给的端口是8889，所以需要给阿里云服务器添加安全规则，开放8889端口 选择防火墙—添加规则—自定义，输入自定义端口号

终端输入：服务器ip地址:8889

此时应该可以直接从本地浏览器直接访问http://address\_of\_remote:8888就可以看到jupyter的登陆界面。

**8. 建立ssh通道**

如果登陆失败，则有可能是服务器防火墙设置的问题，此时最简单的方法是在本地建立一个ssh通道：   
在本地终端中输入ssh username@address\_of\_remote -L127.0.0.1:1234:127.0.0.1:8888   
便可以在localhost:1234直接访问远程的jupyter了。

**windows客户端配置**

（1）需要安装一个windows远程ssh登录Linux的工具，可以是Xshell或SecureCRT，这两个工具的安装简单，如果遇到问题请自行百度；

（2）Xshell或SecureCRT第一次启动需要新建连接，一般输入服务器IP地址、用户名、密码即可，这里不细述；

（3）如果安装了Xshell，在Xshell的菜单栏，点击文件-属性-SSH-隧道-添加，侦听端口输入8888，目标主机设为服务器的IP地址，目标端口设为8888，“说明”中填入jupyter，确定。如果安装了SecureCRT，点击Options-Session Options-Port Forwarding-Add，name中填入jupyter，第一个port填入8888，勾选“Destination host is different from the SSH server”，填入服务器的IP地址，port填入8888，点两次OK；

（4）在第（2）步成功远程登录命令行窗口中输入 $ jupyter notebook；

（5）打开本地windows的浏览器，地址栏输入localhost:8888，回车，输入Linux服务器端配置时第（2）步设置的密码，回车，jupyter界面就出来了。

# Spark

### ****安装Java SE****

1、下载JAVA SE linux版本。下载地址为：   
<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

注意选择Linux版本，根据机器类型选择32位或者64位。

2、下载jdk-8u131-linux-x64.tar.gz后，解压到自己指定目录。

cd ~

cd /usr/local

sudo mkdir java

sudo mkdir scala

sudo mkdir spark

$ tar -zxvf jdk-8u131-linux-x64.tar.gz -C /usr/local/java

3、配置JAVA环境变量。

打开.bashrc文件。

$ nano ~/.bashrc

在.bashrc文件末尾增加如下内容：

export JAVA\_HOME=/dowmload/install/java7/jdk1.7.0\_80 (根据下载版本决定，下同)

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/local:${JRE\_HOME}/local

export PATH=${JAVA\_HOME}/bin:${JRE\_HOME}/bin:$PATH

保存后退出，运行如下命令，使修改环境变量即可生效：

$ source ~/.bashrc

4、检测java是否成功安装。

$ java -version

java version "1.8.0\_131"

如果显示了java安装的版本，则表示已正确安装，可以进行下一步了。

### ****安装Scala****

1、下载Scala的压缩文件。下载地址为：   
<http://www.scala-lang.org/download/>

2、下载scala-2.12.2.tgz后，解压到指定的目录：

$ tar -zxvf scala-2.12.2.tgz -C /usr/local/scala/ (上面已创建该文件夹，下同)

3、配置scala环境变量。

打开.bashrc文件。

$ nano ~/.bashrc

在.bashrc文件末尾增加如下内容：

export SCALA\_HOME=/dowmload/install/scala/scala-2.11.8

export PATH=${SCALA\_HOME}/bin:$PATH

保存后退出，运行如下命令，使修改环境变量即可生效：

$ source ~/.bashrc

4、检测scala是否正确安装。

$ scala -version

Scala code runner version 2.11.8 -- Copyright 2002-2017, LAMP/EPFL and Lightbend, Inc

若显示如上信息，则表示scala已安装正确。

### ****安装Spark****

1、下载Spark的压缩文件。下载地址为：   
<http://spark.apache.org/downloads.html>

2、下载spark-2.1.1-bin-hadoop2.7.tgz后，解压到指定的目录：

$ tar -zxvf spark-2.1.1-bin-hadoop2.7.tgz -C /opt/spark/

3、配置scala环境变量。

打开.bashrc文件。

$ nano ~/.bashrc

在.bashrc文件末尾增加如下内容：

export SPARK\_HOME=/dowmload/install/spark/spark-2.1.1-bin-hadoop2.7

export PATH=${SPARK\_HOME}/bin:$PATH

保存后退出，运行如下命令，使修改环境变量即可生效：

$ source ~/.bashrc

# 阿里云服务器搭建spark特别坑的地方

2017年07月24日 23:56:23

阅读数：1275

**特别坑的地方:**

1. 不能使用公网ip, 就算你把1001/65533给0.0.0.0/0 全部开放端口都没有用, 只能使用内网ip和主机名关联, 用内网ip启动spark

    在/etc/hosts文件: vi /etc/hosts

127.0.0.1 localhost

你的内网ip 你的主机名 (用ifconfig查看内网ip) 主机名的修改在下面

凡是遇到gedit 统统改为vi

vi /etc/hostname 修改主机名

重启一下

然后：

cp $SPARK\_HOME/conf/spark-env.sh.template $SPARK\_HOME/conf/spark-env.sh

打开对应的spark-env.sh文件

ifconfig

vi $SPARK\_HOME/conf/spark-env.sh

export SPARK\_LOCAL\_IP=内网ip

export SPARK\_MASTER\_HOST=你的主机名 (同上)

    对应的slaves文件

你的主机名

用公网ip就报错

修改系统默认的python和pip

进入python

**import** sys

sys.executable

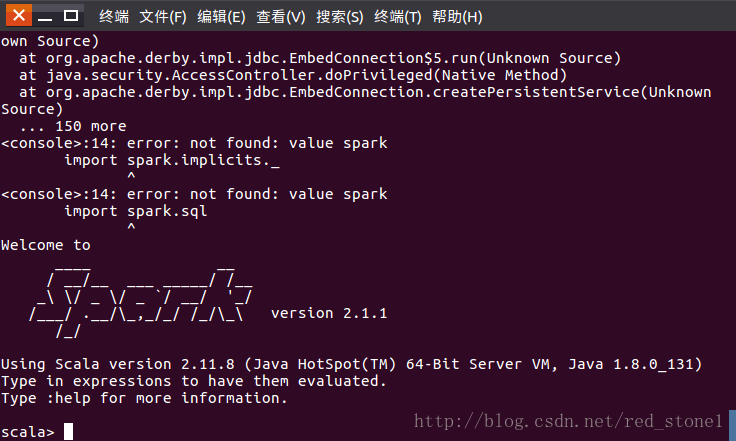
'/root/anaconda3/bin/python'

update-alternatives --install /usr/bin/python python /root/anaconda3/bin/python # 修改python

sudo ln -s /usr/bin/pip /root/anaconda3/bin/pip3.6

4、检测spark是否正确安装。

$ spark-shell

如果出现如下界面，则表示spark已正确安装：   


# 简略Spark输出

Spark（和PySpark）的执行可以特别详细，很多INFO日志消息都会打印到屏幕。开发过程中，这些非常恼人，因为可能丢失Python栈跟踪或者print的输出。为了减少Spark输出 – 你可以设置$SPARK\_HOME/conf下的log4j。首先，拷贝一份$SPARK\_HOME/conf/log4j.properties.template文件，去掉“.template”扩展名。

~$ cp $SPARK\_HOME/conf/log4j.properties.template $SPARK\_HOME/conf/log4j.properties

编辑新文件，用WARN替换代码中出现的INFO。你的log4j.properties文件类似：

vi $SPARK\_HOME/conf/log4j.properties

# Set everything to be logged to the console

log4j.rootCategory=**WARN**, console

log4j.appender.console=org.apache.log4j.ConsoleAppender

log4j.appender.console.target=System.err

log4j.appender.console.layout=org.apache.log4j.PatternLayout

log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Settings to quiet third party logs that are too verbose

log4j.logger.org.eclipse.jetty=**WARN**

log4j.logger.org.eclipse.jetty.util.component.AbstractLifeCycle=ERROR

log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=**WARN**

log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=WARN

[复制代码](javascript:void(0);)

现在运行PySpark，输出消息将会更简略！

# spark UI

cd $SPARK\_HOME/conf

cp spark-defaults.conf.template spark-defaults.conf

vi spark-defaults.conf

spark.eventLog.enabled=true # 去掉注释，设置为true

阅读https://www.cnblogs.com/sorco/p/7070922.html

更详细的日志配置

### spark的conf/spark-defaults.conf配置

spark.yarn.historyServer.address=node2:18080

spark.history.ui.port=18080

spark.eventLog.enabled=true

spark.eventLog.dir=hdfs:///tmp/spark/events

spark.history.fs.logDirectory=hdfs:///tmp/spark/events

如果你是运行在yarn之上的话，就要告诉yarn，你spark的地址，当我在yarn上点击一个任务，进去看history的时候，他会链接到18080里面，如果你配的是node1:18080，那么你就要在node1上启动spark的history server（见后面注①）

最下面两个配置是运行spark程序的时候配置的，一旦你运行spark，就会将日志等发送到那两个目录，有了这两个目录，spark的historyserver就可以读取spark的运行状态信息日志等读取并展示

18080是spark的history server，会显示出你最近spark跑过的一些程序，点击execution后，点击最右边的日志（如果有的话），会重定向到19888（见后面注②），这个是mr的jobserver的地址（启动命令：mr-jobhistory-daemon.sh start historyserver）

spark.yarn.historyServer.address和spark.history.ui.port如果缺少其中一个，日志就看不到

综上，1和2两个配置齐全，才可以查看spark的stdout和stderr日志

请注意,spark在交互式shell下运行时候,这里的sc即SparkContext 的一个实例已经自动生成了，所以在生成一个SparkContext会报错。具体配置看下节。

# 为Jupyter配置Spark开发环境

安装时，参照Quick Start What isApache Toree ，仅需两条命令即可完成安装。

　　pip install toree

　　jupyter toree install --spark\_home=your-spark-home12

pip install pyspark

pip uninstall py4j

pip install py4j==0.10.4 # 注意py4j版本要与spark内置的一致！

首先我们需要导入py4j:

    其实就是在python3里面导入了spark和sc模块

首先我们需要导入py4j:

    其实就是在python3里面导入了spark和sc模块



要注意了:下图红框里面的要对应你spark/python/lib里面的文件

import os

import sys

spark\_name = os.environ.get('SPARK\_HOME',None)

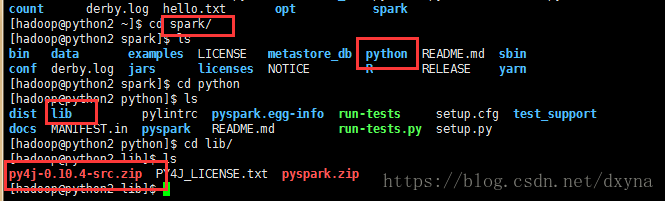
if not spark\_name:

raise ValueErrorError('spark环境没有配置好')

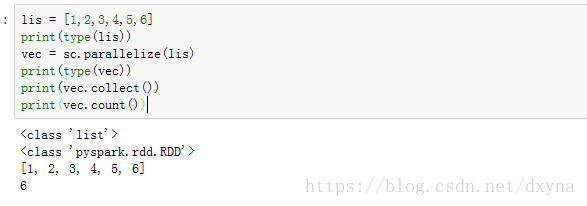
sys.path.insert(0,os.path.join(spark\_name,'python'))

sys.path.insert(0,os.path.join(spark\_name,'python/lib/py4j-0.10.4-src.zip'))

exec(open(os.path.join(spark\_name,'python/pyspark/shell.py')).read())



现在我们就可以使用pyspark了:



# 3.安装spark kernel

到目前为止jupyter只有一个默认的python3的kernel,而且并没有连接任何spark.使用一下命令查看

jupyter kernelspec list

pip install pyspark

1 **基于pyspark的jupyter notebook**

此处我们使用spark bin目录下的pyspark连接notebook,即启动./pyspark默认启动notebook.只需要在全局文件./bashrc中设置即可。

打开./bashrc文件

vim ~/.bashrc

添加如下两条全局命令

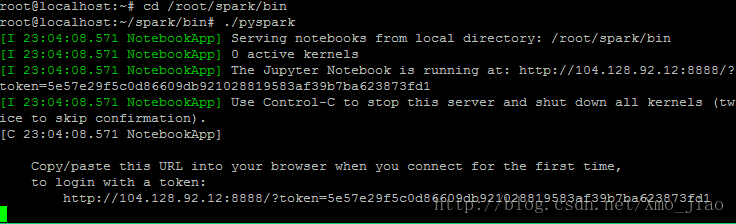
export PYSPARK\_DRIVER\_PYTHON=jupyter

export PYSPARK\_DRIVER\_PYTHON\_OPTS="notebook"

在spark bin目录下测试notebook是否安装了pyspark，成功即出现如下图：

cd $SPARK\_HOME/bin

./pyspark



jupyter容易报以下错误

Exception: Java gateway process exited before sending the driver its port number

原因之一是版本冲突（其他情况先百度，百度解决不了再解决版本问题）

直接卸了jupyter notebook，再重装。重装后会有红字警告出现，依次卸模块，并安装相应版本

pip uninstall jupyter notebook # 卸载

pip install jupyter notebook==4.4.0 # 指定安装版本，一定要指定版本！

主要的版本不兼容发生在 jupyter notebook、pyspark、toree上

* 2 **基于Scala spark的jupyter notebook**

此处使用Apache toree给notebook安装scala kernel

[toree官网下载页](https://github.com/apache/incubator-toree),不需要解压，直接使用pip install安装

wget https://dist.apache.org/repos/dist/dev/incubator/toree/0.2.0/snapshots/dev1/toree-pip/toree-0.2.0.dev1.tar.gz

pip install toree-0.2.0.dev1.tar.gz

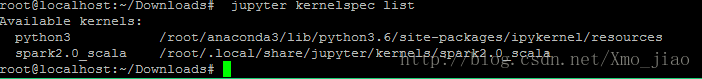
接着使用一下命令安装，其中spark://localhost.localdomain:7077为你的spark地址，$SPARK\_HOME为你的spark安装目录

jupyter toree install --spark\_opts='--master=spark://localhost.localdomain:7077' --user --kernel\_name=Spark**2.1** --spark\_home=$SPARK\_HOME

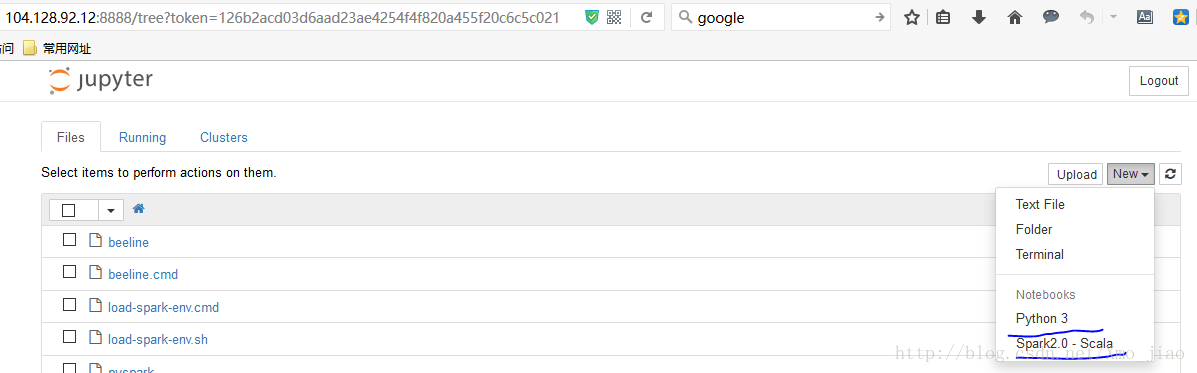
测试是否安装成功，列出kernel列表，发现有两个kernel：python3 和spark 2.1\_scala

jupyter kernelspec list

* 1



此时，python和scala版的jupyter安装成功



# Pychrme远程调试spqrk

## 先下载py4j模块

pip install py4j

## 设置环境变量

cd $SPARK\_HOME/python/lib

ls 查看py4j的版本，后面要用

vim /etc/profile

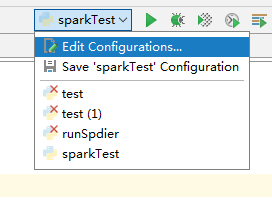
添加：

export PYTHONPATH=$SPARK\_HOME/python/:$SPARK\_HOME/python/lib/py4j-0.10.4-src.zip

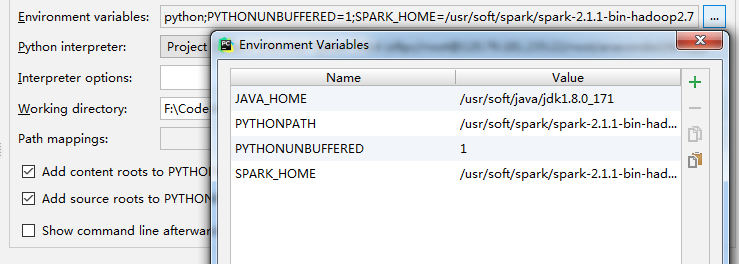
执行 source /etc/profile 生效

（注意py4j-0.10.4-src.zip要跟自己电脑上一致,如果不是root用户,设置相应用户环境变量（vim ~/.bashrc））

1. win下Pychram设置



配置4个环境变量（都是以服务器为准）



# 解决pychram中找不到scrapy

bash: scrapy: command not found

#### 一、场景

执行 pip3.6 install scrapy 后，安装成功且执行 import scrapy 成功

注意，一定要在/root/anacond3/bin下使用pip3.6 install scrapy 确保该scrapy版本依赖anaconda

#### 二、问题

在shell中执行 scrapy 返回 bash: scrapy: command not found

#### 三、解决办法

（1）进入 Python 的主目录，如cd /root/anacond3/bin ，查找 scrapy 项   
（2）检查 cd /usr/bin/ | ll | grep scrapy，查看是否存在   
（3）不存在则执行 ln -s /root/anacond3/bin scrapy（改为scrapy安装路径） /usr/bin/scrapy（这个路径不改）   
（4）回到shell，执行 scrapy version，成功