# **boost.python编译库说明**

## **一．Linux环境下编译说明**

### **1. centos7参考环境**

centos7.6

gcc、g++ (4.8.5版本)

cmake-3.4.1(tar.gz)

python3.9.13(tar.gz)

boost\_1\_80\_0(tar.gz)

### **2. 安装编译python3.9.13**

##### **步骤一：安装前的准备**

##### **Centos7默认安装了python2.7版本，我们需要编译安装python3.9.13。**

##### **#1. 先看一下已安装的python在哪里。**

[root@centos7 ~]*# whereis python*

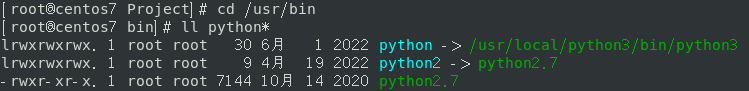
*1*

显示python在/usr/bin目录中。

##### **#2. 进入/usr/bin查看python详情：**

[root@centos7 ~]# cd /usr/bin

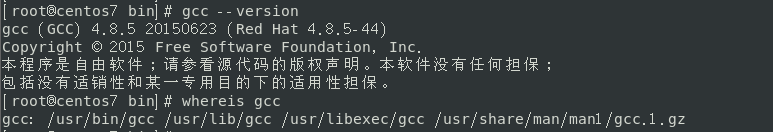
[root@centos7 bin]*# ll python\**

**

##### **如果没有安装python3.9.13，则需要按如下步骤安装。**

##### **3#. 编译安装gcc、g++：**

在安装cmake前，先要确认系统已经安装了gcc、g++，否则在安装cmake时会报错。centos7下默认的gcc、g++版本是4.8.5。



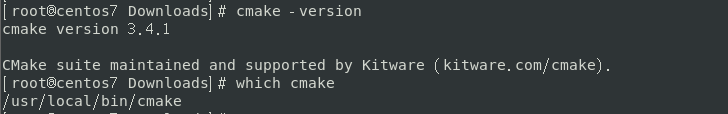
**但如果没有安装gcc**、g++**，则需要安装:**

[root@centos7 bin]# yum install gcc gcc-c++

##### **4#. 编译安装CMake：**

先检查是否已安装cmake：

[root@centos7 Downloads]# cmake -version



说明已安装cmake-3.4.1。

**但如果没有安装cmake，则需要安装，这里是cmake-3.4.1。**

1. 下载cmake包，解压到指定路径：

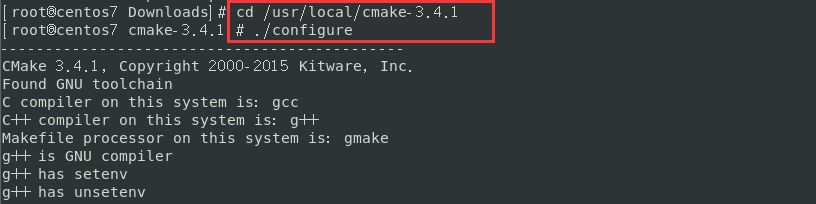
[root@centos7 Downloads]# wget <http://www.cmake.org/files/v3.4/cmake-3.4.1.tar.gz>

[root@centos7 Downloads]# tar -zxvf cmake-3.4.1.tar.gz -C /usr/local/

(2) 进入解压后的cmake目录配置：

[root@centos7 Downloads]# cd /usr/local/cmake-3.4.1

[root@centos7 cmake-3.4.1]# ./configure



(3) 编译、安装cmake：

[root@centos7 cmake-3.4.1]# *m*ake

[root@centos7 cmake-3.4.1]# make install

(4) 查看cmake版本：

[root@centos7 cmake-3.4.1]# *c*make -version

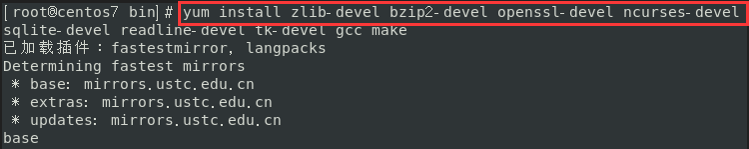
注意：

1. 在解压后的cmake-3.4.1目录下，编译源码及安装。

##### **步骤二：安装python3相关依赖包和编译环境**

**1#. 在安装python3时，需要先安装依赖环境：**

[root@centos7 bin]# *yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gcc make*

**

如果不安装相关依赖包，在安装python3时会报找不到zlib、SSL等一连串错误！

如果本机安装了gcc、make，那么可以去掉这2项。

##### **步骤三：下载、编译、安装**

##### **1#. 下载python3.9.13安装包**

可手动下载到Downloads文件夹：

下载链接：<https://www.python.org/downloads/source/>



或者wget命令下载到Downloads文件夹：

[root@centos7 Downloads]*# wget <https://www.python.org/>ftp/*

*python/3.9.13/Python-3.9.13.tgz*

##### **#2. 解压安装包**

解压到指定目录：/usr/local/

[root@centos7 Downloads]# tar -xvf Python-3.9.13.tgz -C /usr/local/

##### **#3. 编译安装包**

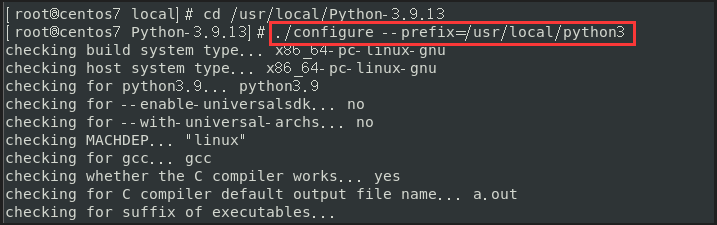
##### **因为下载的包未编译，所以我们需要编译一下。**

##### **先创建编译安装目录/usr/local/python3，再到/usr/local/Python-3.9.13 编译。**

[root@centos7 Downloads]# mkdir -p /usr/local/python3

[root@centos7 Downloads]# cd /usr/local/Python-3.9.13

[root@centos7 Python-3.9.13]# ./configure --prefix=/usr/local/python3



##### **注意：**

（1）--prefix=是指定安装的目录，可以把所有资源文件放在/usr/local/python3路径下，这样不会杂乱，也便于卸载或移植软件。

（2）如果编译报错，就先执行make clean，再重新执行 ./configure。

[root@centos7 Python-3.9.13]*# make clean*

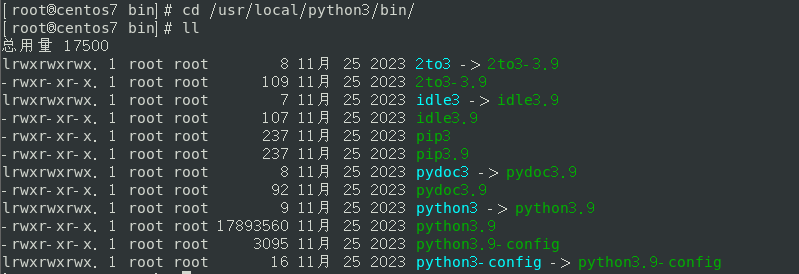
[root@centos7 Python-3.9.13]# ./configure --prefix=/usr/local/python3

（3）如果编译成功，则直接执行make ，再执行make install。

[root**@centos7** Python-3.9.13]*# make*

[root**@centos7** Python-3.9.13]*# make install*

执行完毕，就可以切换到/usr/local/python3/bin目录去查看了。



##### **步骤四：创建软链接**

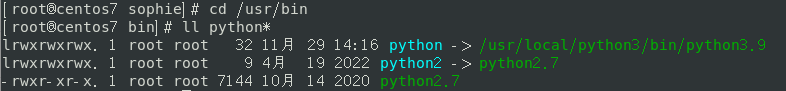
可通过软链接保留多个python版本，这个软链接决定了在命令行运行python的默认版本。

##### **#1. 先修改旧版本的软链接**

[root@centos7 bin]# unlink python

##### **#2. 再创建新版本的软连接**

[root@centos7 bin]# ln -s /usr/local/python3/bin/python3.9 /usr/bin/python



##### **#3. 检查python版本**

[root@centos7 bin]*# python -V*

*8*

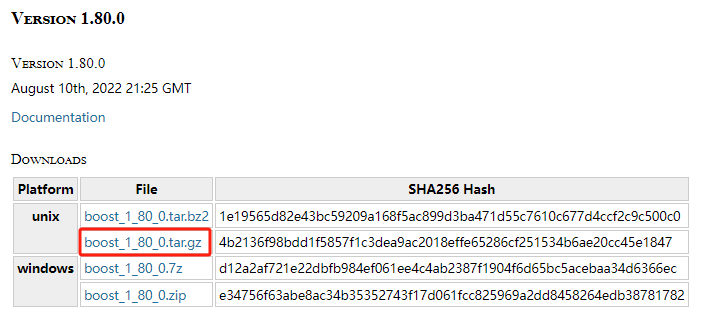
软链接创建成功。

### **3. 安装编译boost\_1\_80\_0**

### **进入官网**http://www.boost.org/**，选择较新的稳定版本，并且跟python版本相对应，这里是1.88.0版本。**

**步骤一：下载boost\_1\_80\_0安装包，解压缩**

可手动下载到Downloads文件夹：

下载链接：https://www.boost.org/users/history/version\_1\_80\_0.html  


或者wget命令下载到Downloads文件夹：

[root@centos7 Downloads]# **wget**

*https://boostorg.jfrog.io/artifactory/main/release/1.80.0/source/boost\_1\_80\_0.tar.gz*

##### **步骤二：解压安装包**

解压到指定目录：/usr/local/

[root@centos7 Downloads]# tar -xvf /home/sophie/Downloads/boost\_1\_80\_0.tar.gz -C /usr/local/

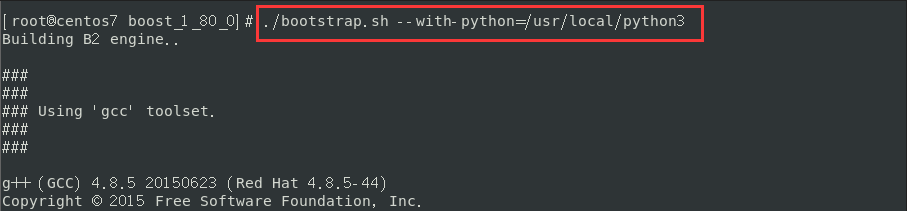
##### **步骤三：编译**

进入/usr/local/boost\_1\_80\_0目录编译

[root@centos7 Downloads]# cd /usr/local/boost\_1\_80\_0/

执行bootstrap.sh

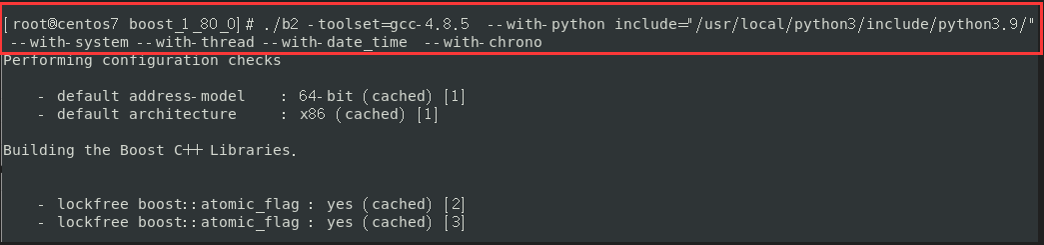
[root@centos7 boost\_1\_80\_0]# ./bootstrap.sh --with-python=/usr/local/python3



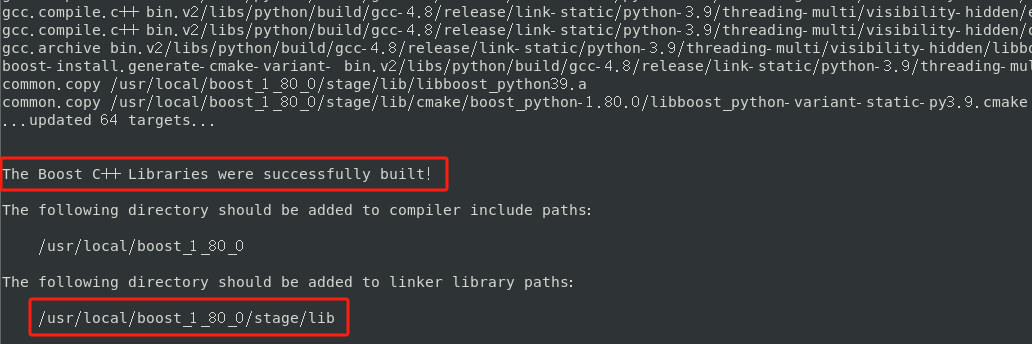
##### **步骤四：安装**

##### **说明：-with-python 需要python3版本，通过include参数指定自定义版本的python包含路径。**

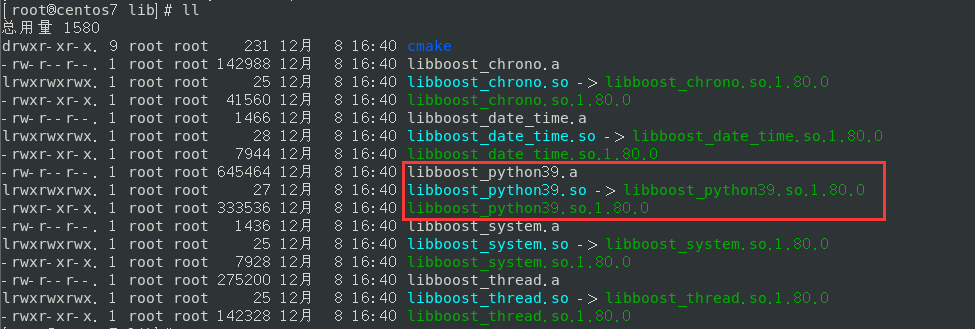
[root@centos7 boost\_1\_80\_0]# ./b2 -toolset=gcc-4.8.5  --**with**-python **include**="/usr/local/python3/include/python3.9/" --**with**-thread --**with**-date\_time  --**with**-chrono



**boost目录下会生成stage文件夹，/stage/lib就是C++所需的python3的lib文件。至此boost.python3库编译完成。如下图所示：**



##### **编译生成的 /usr/local/boost\_1\_80\_0/stage/lib文件，如下图所示：**



### **4. 编译source**

##### **如果没有特殊的需求，建议使用python3.9.13版本，直接使用我们提供的库。**

如果是其他python版本，则需要根据自己编译安装的python版本，及该python版本对应的编译后的boost库，自行编译所需的python封装库。

xtp\_api\_python-master/source下有封装api的python源码。

##### **步骤一. 查看CMakeList.txt中python和boost路径**

##### **路径：xtp\_api\_python-master/source/Linux/xtp\_python3\_2.2.39.3/CMakeList.txt**

注意：设置python目录及boost库目录要正确，必须为解压缩后的路径。

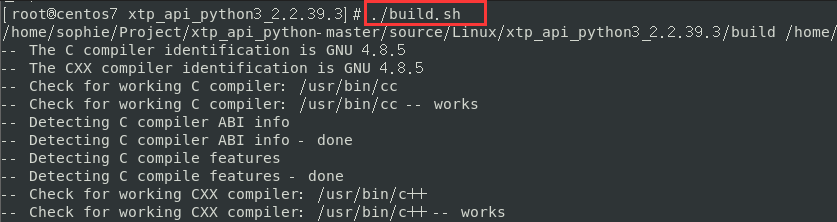


**步骤二. 执行build.sh**

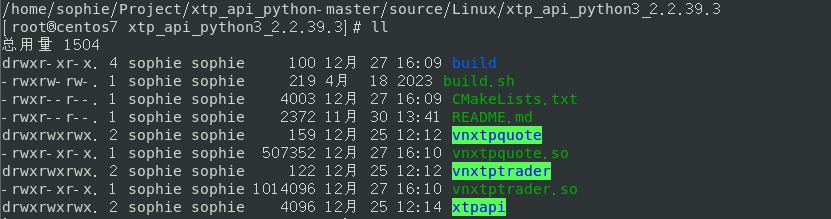
**路径：xtp\_api\_python-master/source/Linux/xtp\_python3\_2.2.39.3/build.sh**

进入：xtp\_api\_python-master/source/Linux/xtp\_python3\_2.2.39.3/

[root@centos7 xtp\_python3\_2.2.39.3]*# ./build.sh*

**

**执行完毕后，该目录会生成vnxtpquote.so和vnxtptrader.so库文件。**



### **5. 运行Demo**

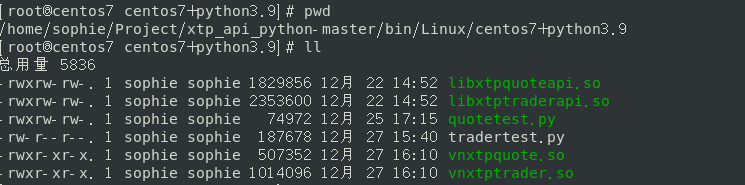
##### **运行方式：把xtpapi中两个.so文件、编译source生成的两个.so文件，以及/bin/test中的两个.py文件 拷贝到对应的平台下，并修改.py文件的账户信息，启动脚本运行。**

##### **运行平台路径如：**xtp\_api\_python-master/bin/Linux/Linux+python3.9/。

（1）先将xtpapi中原有的两个库文件 [libxtpquoteapi.so](http://libxtpquoteapi.so/)、[libxtptraderapi.so](http://libxtptraderapi.so/) 拷贝至 运行平台路径下。

（2）再将编译source生成的两个库文件 [vnxtpquote.so](http://vnxtpquote.so/)、[vnxtptrader.so](http://vnxtptrader.so/) 拷贝至 运行平台路径下。

（3）最后将/bin/test中的两个脚本文件 [quotetest.py](http://quotetest.py/) 和 [tradertest.py](http://tradertest.py/) 拷贝至 运行平台路径下。



（4）如果测试行情，则修改quotetest.py文件，改为实际使用的 ip、port、user、password、local\_ip，然后运行quotetest.py。**命令提示符方式如下：**

[root@centos7 Linux+python3.9]*# python quotetest.py*

（5）如果测试交易，则修改tradertest.py文件，改为实际使用的 ip、port、user、password、local\_ip，然后运行tradertest.py。

###### **命令提示符方式如下：**

[root@centos7 Linux+python3.9]*# python tradertest.py*

## **二．windows环境下编译说明**

### **1. win10参考环境**

Windows10 (64位)

cmake-3.4.1(64位.zip)

VS2010 (V10.0版本) 或VS2015(V14.0版本)

python3.9.13 (64位)

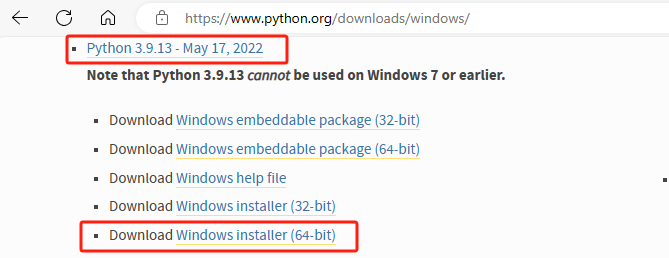
boost\_1\_80\_0(64位.7z)

### **2. 安装64位python3.9.13**

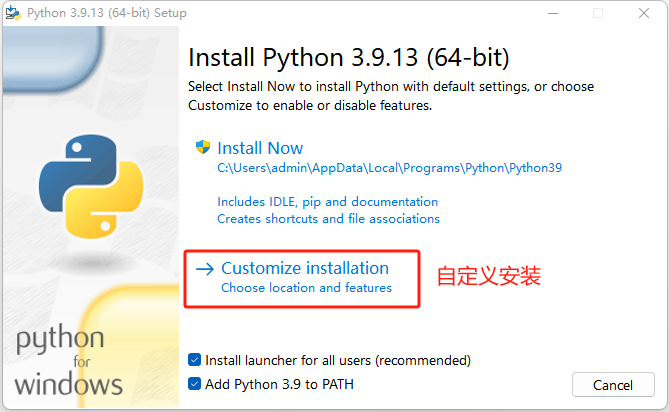
##### **步骤一：下载python**

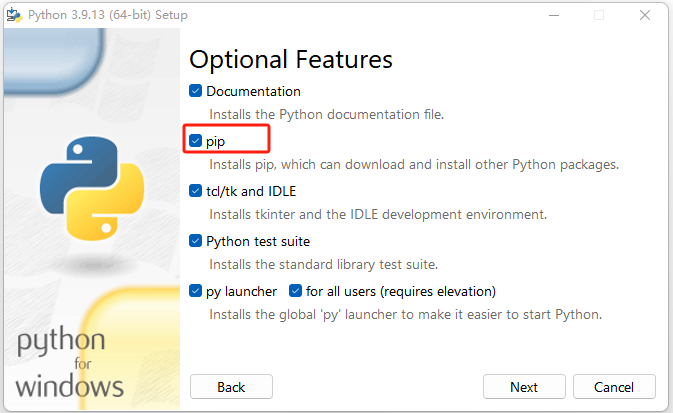
进入官网：<https://www.python.org/downloads/windows/>，选择64位python3.9.13。

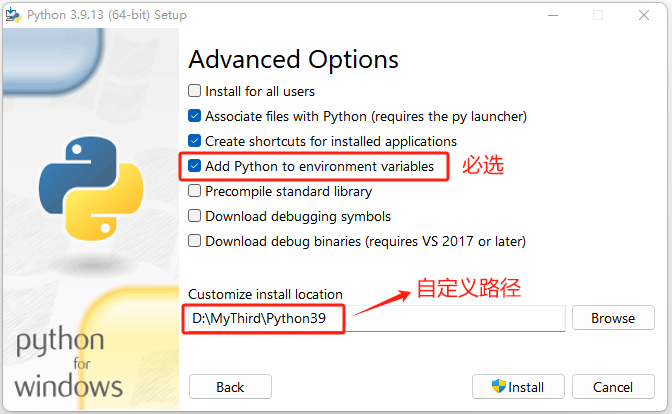
说明：python3开始使用的是Anaconda3，但是在编译boost时提示错误，无法完成编译，所以最终选择的是64位python3.9.13  。



##### **步骤二：安装python**

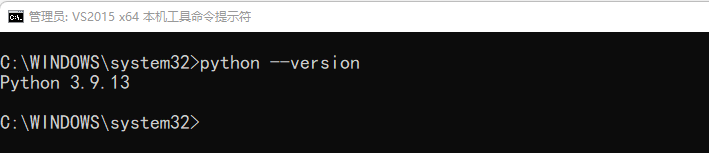






##### **步骤三：查看python版本**

安装成功后，打开cmd进入命令提示符，输入：python --version



### **3. 安装64位boost\_1\_80\_0**

##### **步骤一：下载boost**

#### 进入官网：<http://www.boost.org/>，选择较新的稳定版本，并且跟python版本相对应，这里是1.80.0版本

##### **#1. 官网选择历史版本**w_6

##### **或直接从官网链接下载**

##### 下载链接：[https://boostorg.jfrog.io/artifactory/main/release/1.80.0/source/](https://boostorg.jfrog.io/artifactory/main/release/1.66.0/source/)，选择boost\_1\_80\_0.7z **步骤二：解压缩、安装**

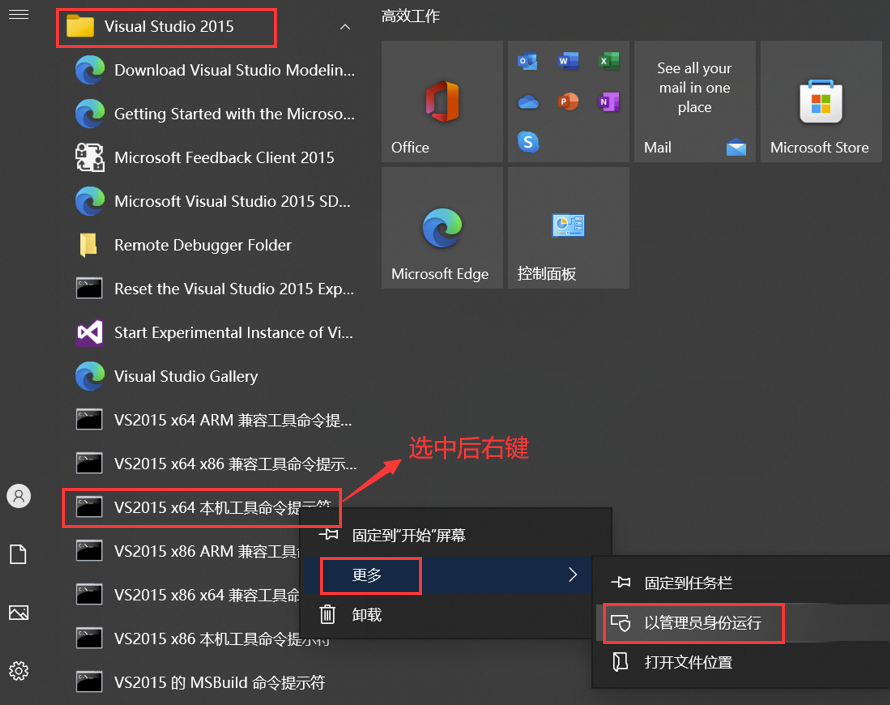
下载完成之后，解压至 D:\MyThird\boost\_1\_80\_0

### **4. 安装VS2015**

下载并安装 VS2010 或 VS2015，或者 VSCode。

### **5. 编译boost.python库**

以VS2015版本64位为例，打开64位的编译终端，注意：以“管理员身份运行”。



##### **步骤一：执行bootstrap.bat**

（1）切换到D盘，执行命令：

C:\WINDOWS\system32>d:

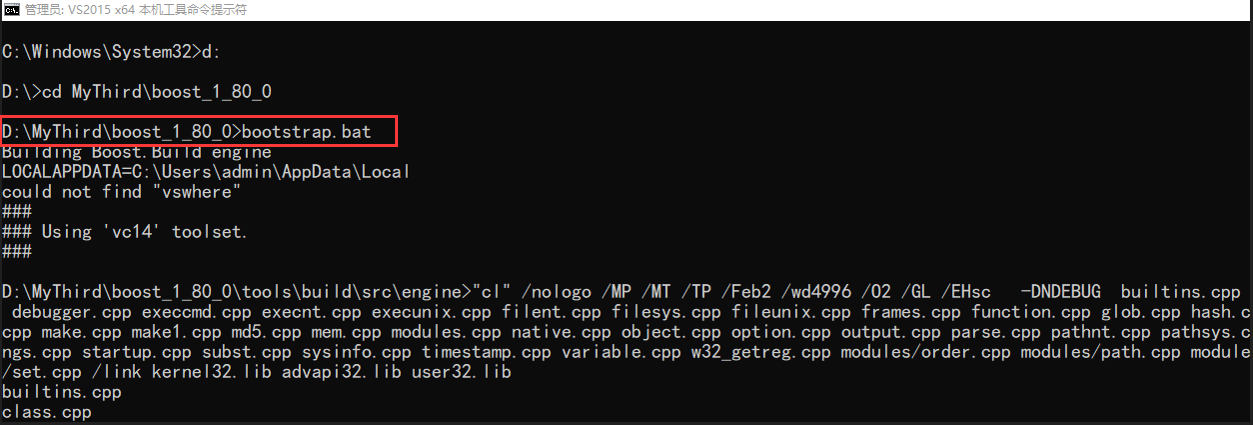
（2）进入boost源码目录，执行命令：

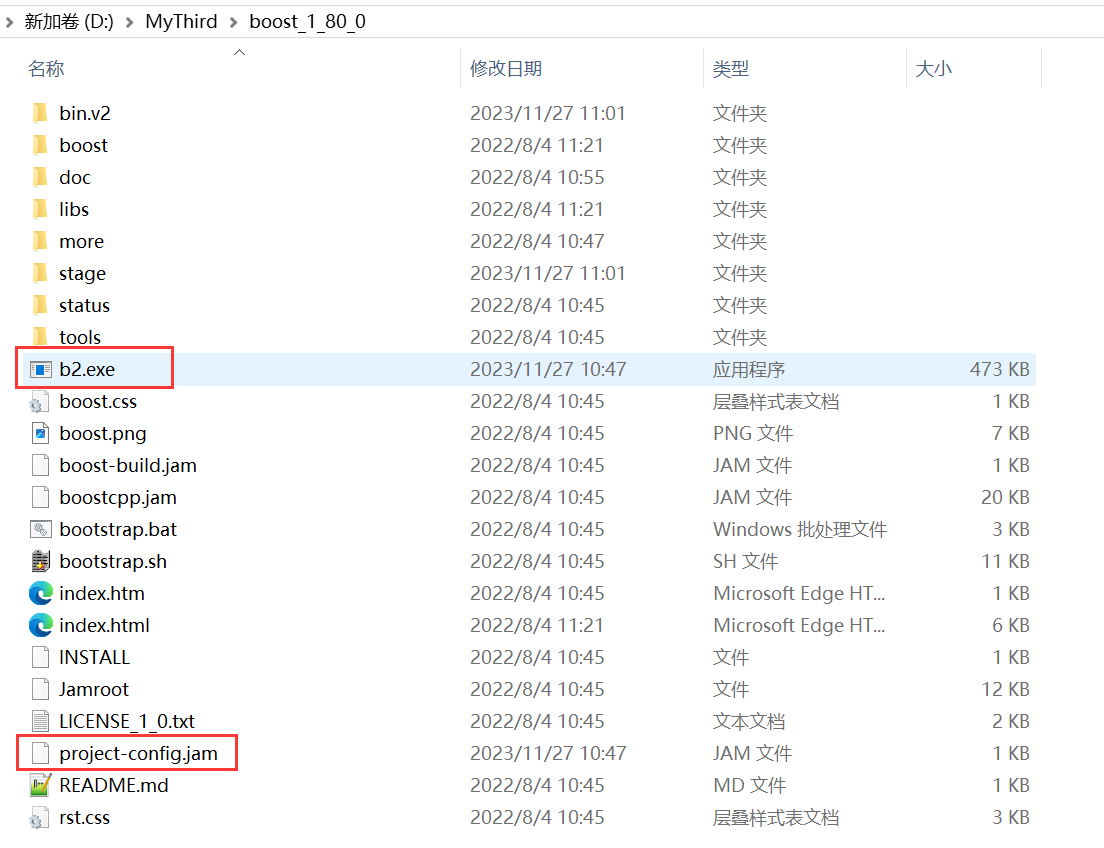
D:\>cd MyThird\boost\_1\_80\_0

（3）运行bootstrap.bat，执行命令：

D:\MyThird\boost\_1\_80\_0\bootstrap.bat

运行成功后，生成 b2.exe文件。

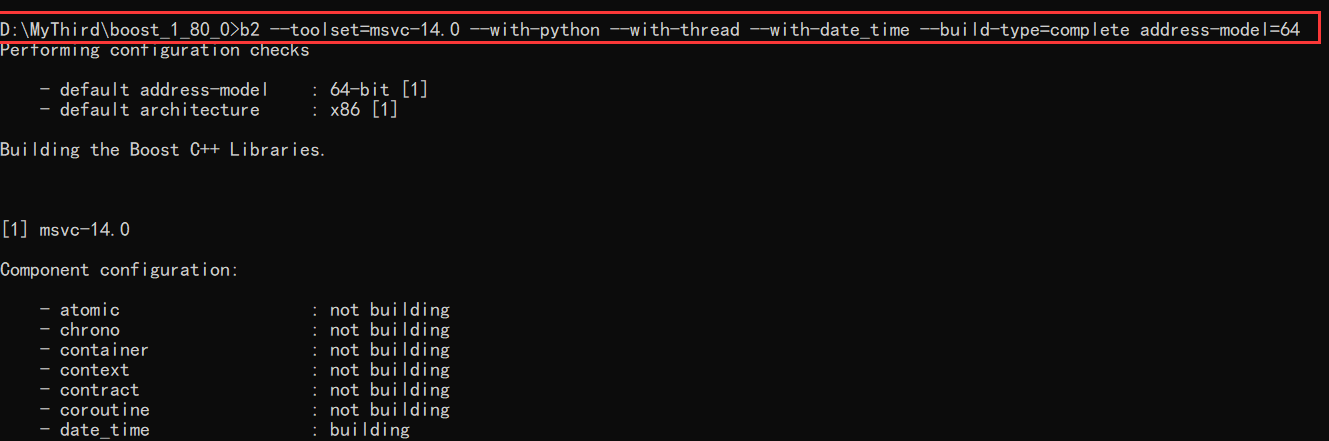
注意：较高版本的boost，运行成功只会生成b2.exe，不会生成bjam.exe。



**步骤二： 编译，生成64位的静态库**

在命令行继续执行命令：

D:\MyThird\boost\_1\_80\_0>b2 --toolset=msvc-14.0 --with-python --with-thread --with-date\_time --build-**type**=complete **address**-model=64



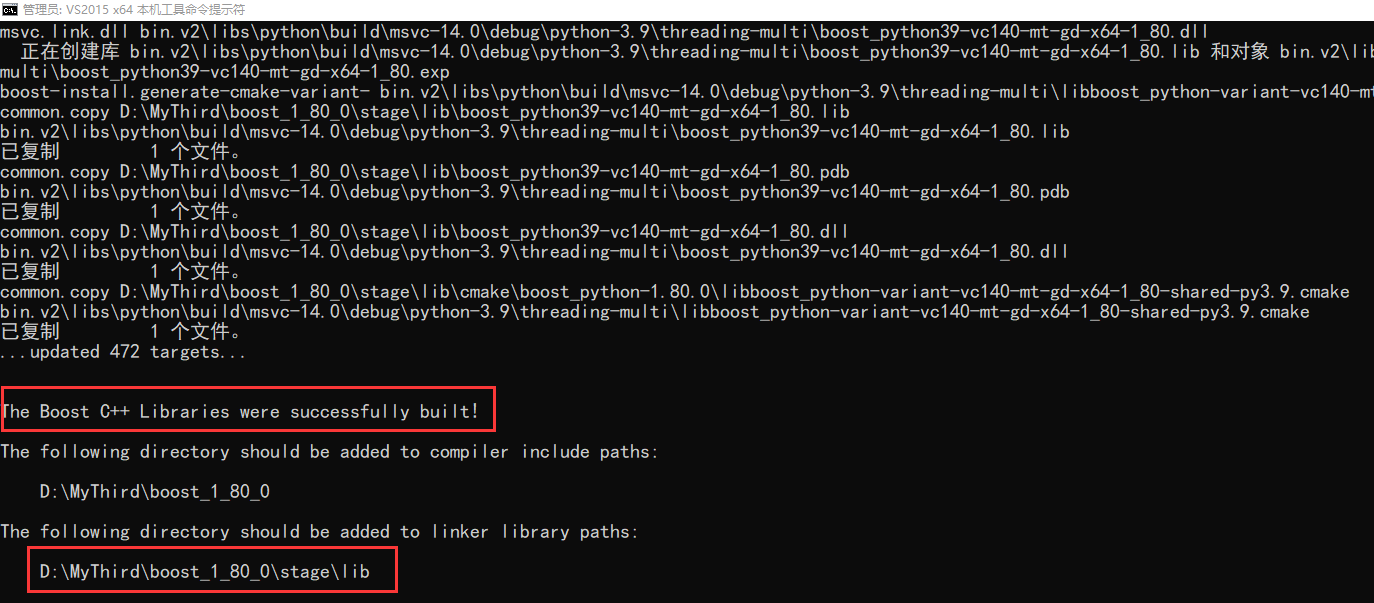
说明：

（1）如果使用较高版本的boost没有生成bjam.exe，那么只能使用b2编译，而不能使用 bjam --toolset。

（2）--toolset=msvc-14.0 表示VS2015的版本号为14.0，如果使用的VS2010则=msvc-10.0，请更改为实际使用的VS版本号。

（3）address-model=64 表示生成的库为64位，如果=32就表示32位

（4） --with-python，系统要能找安装的python3版本，直接在cmd里面输入python能弹出python3的具体版本信息。



**执行成功后，在boost\_1\_80\_0/stage/lib/下就是C++所需python的lib文件。至此boost.python库编译完成。**

### **6. 使用w_12CMake生成工程**

##### **步骤一：创建工程目录**

##### **python版本SDK的 source下包含 C++版本的api 和 封装api的python源码。**

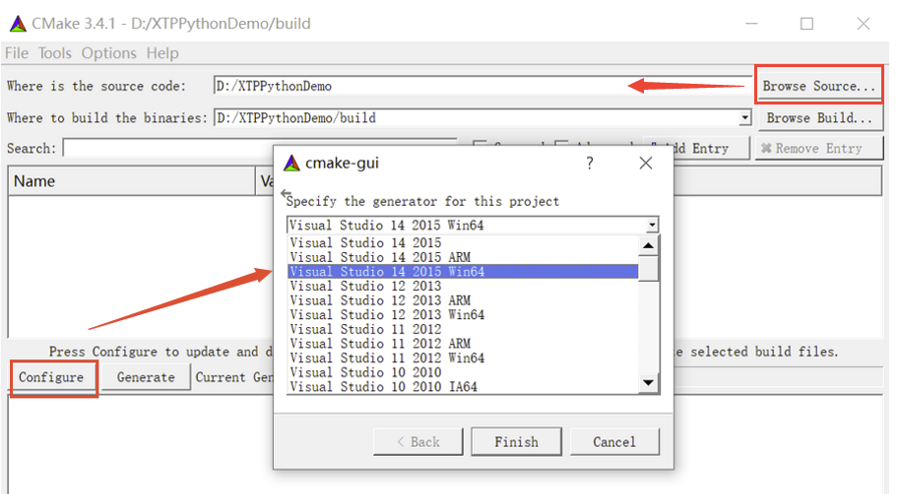
1. python版本SDK下载地址：<https://github.com/ztsec/xtp_api_python/>， 下载后解压缩 xtp\_api\_python-master.zip。  
     
   （2）将source/Windows/xtp\_api\_python3\_2.2.39.3/ 下所有文件拷贝到 D:/XTPPythonDemo文件夹。  
     
   （3）在XTPPythonDemo下创建build文件夹。

##### **步骤二：修改CMakeLists.txt**

****

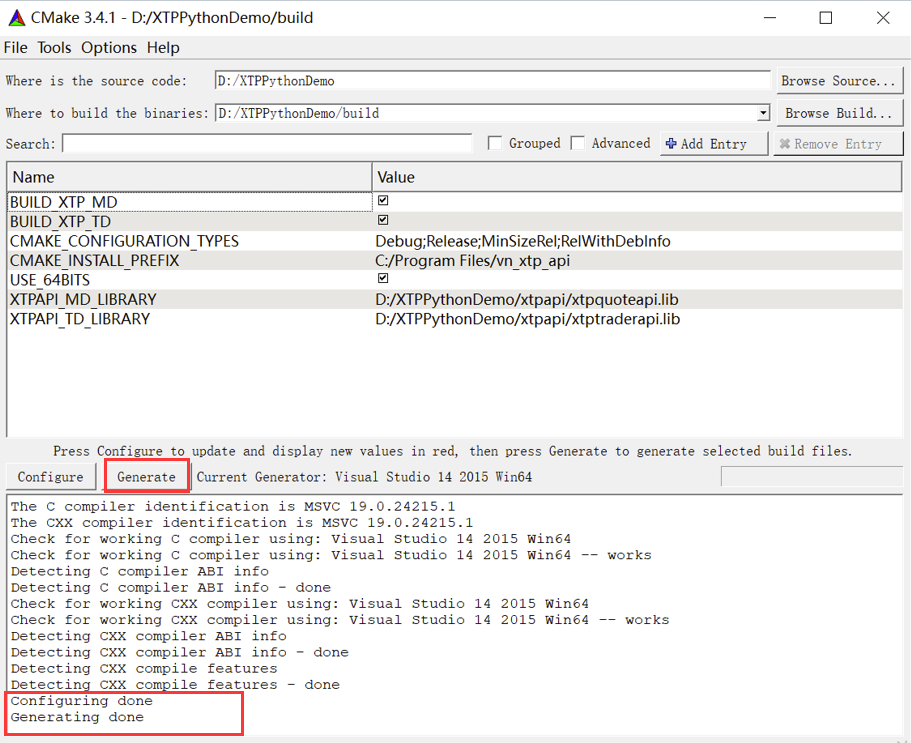
##### **步骤三：Cmake配置**

1. 点击Browse Source按钮，指定XTPPythonDemo文件夹。
2. 点击Browse Build按钮，指定XTPPythonDemo/build文件夹。  
     
   （3）点击Configure按钮，选择VS版本，同编译boost.python库的vs版本一致。



##### **步骤四：Cmake生成工程**

点击Genertate按钮，执行成功后，可进入build文件夹查看生成的C++工程。



### **7. 编译source**

##### **将source下封装api的python源码，编译生成64位的release版本库文件：vnxtpquote.dll、vnxtptrader.dll。**

1. 进入D:/XTPPythonDemo/build/，使用VS2015打开.sln工程，配置X64位、release版本。  
     
   （2）点击菜单 生成->生成解决方案，编译成功后，可至 XTPPythonDemo/build/lib/Release/ 查看生成的 vnxtpquote.dll、vnxtptrader.dll文件。

（3）将生成的.dll文件后缀改名为.pyd，也可点击Cmake的Genertate按钮生成.pyd文件，修改成功后显示为 vnxtpquote.pyd、vnxtptrader.pyd文件。

### **8. 运行Demo**

##### **运行方式：把xtpapi中两个.dll文件、编译source生成的两个.pyd文件，以及/bin/test中的两个.py文件 拷贝到对应的平台下，并修改.py文件的账户信息，启动脚本运行。**

##### 运行平台路径如：xtp\_api\_python-master/bin/Windows/win64+python3.9/。

##### （1）先将xtpapi中原有的库文件 xtpquoteapi.dll、xtptraderapi.dll 拷贝至 运行平台路径下。

##### （2）再将编译source生成的库文件 vnxtpquote.pyd、vnxtptrader.pyd 拷贝至 运行平台路径下。

##### （3）最后将/bin/test中的脚本文件 quotetest.py 和 tradertest.py 拷贝至 运行平台路径下。

##### （4）如果测试行情，则修改quotetest.py文件，改为实际使用的 ip、port、user、password、local\_ip，然后运行quotetest.py。

###### **命令提示符方式如下：**

C:WINDOWS\system32> python D:\xtp\_api\_python-master\bin\Windows\win64+python3.9\quotetest.py

（5）如果测试交易，则修改tradertest.py文件，改为实际使用的ip、port、user、password、local\_ip，然后运行tradertest.py。

###### **命令提示符方式如下：**

C:WINDOWS\system32> python D:\xtp\_api\_python-master\bin\Windows\win64+python3.9\tradertest.py