Python依赖包迁移

当目标服务器有外网时，只需导出包列表，再到目标服务器安装包：

导出包列表：pip freeze > requirements.txt

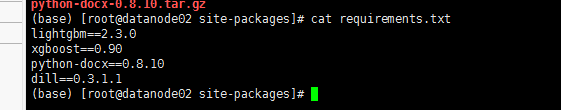
按照包列表安装包：pip install -r requirement.txt

当目标服务器没外网时，不只要导出包列表，还要导出包文件，再到目标服务器安装：

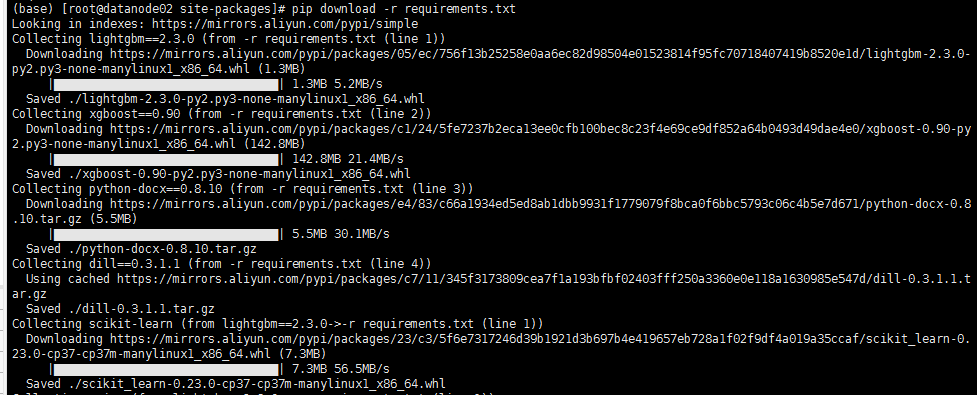
1.新建site-packages目录，进入到site-packages目录下；

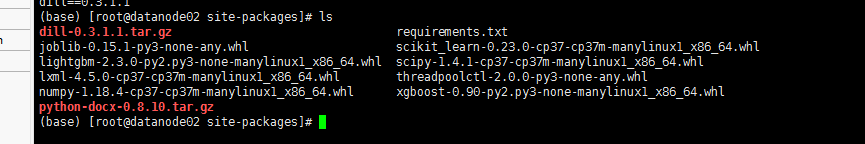
2.在site-packages目录下执行pip freeze >requirements.txt；

3.查看requirements.txt，可以看到当前机器的python所有依赖包已生成列表



4.在当前目录下执行pip download -r requirements.txt,可以看到pip开始下载安装包及所有依赖包



5.等待下载完成后，可以看到当前目录下已经有了依赖包文件  


6.将site-packages文件夹打包，移动至我们需要这些依赖包的机器上；

7.在当前机器的site-packages目录下执行：

pip install --no-index --find-links=/xxx/xxx/site-packages -r /xxx/xxx/site-packages/requirements.txt

(其中：/xxx/xxx/site-packages是指定依赖包路径，/xxx/xxx/site-packages/requirements.txt是指定依赖包列表路径)；

8.执行pip list，查看当前pip安装的所有python依赖库，可以看到我们迁移过来的库已经安装成功了

Python虚拟环境迁移

## **第三步：在linux服务器上为项目创建虚拟环境，并安装项目所需的依赖**

1.切换到pip3所在的目录 /usr/local/python/bin，执行以下命令

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | # 安装虚拟环境 pip3 install virtualenv  # 创建虚拟环境 ENV virtualenv ENV  # 切换到虚拟环境所在的目录 cd ENV  # 启用虚拟环境 source ./bin/activate  # 安装依赖清单里的库 pip3 install -r requirements.txt  # 列出当前虚拟环境所安装的依赖库 pip3 list |

# ****Virtualenv的安装****

## ****pip安装virtualenv****

pip install virtualenv   #py2安装

pip3 install virtualenv   #py3安装，这样用virtualenv创建的virtualenv默认python版本是py3

2.创建venv

virtualenv --no-site-packages --python=python3 venv

venv就是虚拟环境的文件夹，通常取名venv(用户自定义)。

--no-site-packages　　表示不添加系统里面python已安装的第三方库

3.启动虚拟环境并安装第三方库

venv\Script\activate #(windows)

source venv/bin/activate #(linux/macos)  
  
pip install xxx

4.部署到服务器，环境迁移

# 开发的电脑上导出 pip list 到 requirements.txt 文件

pip freeze > requirements.txt

# 另一台新服务器上，安装项目虚拟环境里面的依赖包  
  
pip uninstall -ry requirements.txt

pip install -r requirements.txt

5.退出虚拟环境

deactivate

如果是有网环境：

 source env/bin/activate   # 先激活本地虚拟环境

pip freeze > requirements.txt

pip install -r requirements.txt

作者：SoloSmart  
链接：https://www.jianshu.com/p/ef54a30eb936  
来源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

## 方法二

 source venv/bivn/activate   # 先激活本地虚拟环境

pip freeze > requirements.txt

在开发环境先下载好离线包：pip download -d your\_offline\_packages -r requirements.txt

打包your\_offline\_packages，并copy到目标环境

在目标环境创建新的virtualenv环境并激活:

virtualenv -p python\_path venv

source venv/bivn/activate

批量安装离线安装包

pip install --no-index --find-links=your\_offline\_packages -r requirements.txt

作者：SoloSmart  
链接：https://www.jianshu.com/p/ef54a30eb936  
来源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

## 方法三

打包已经创建好的venv，并拷贝到目标环境解压

venv/bin/下修改activate文件中参数

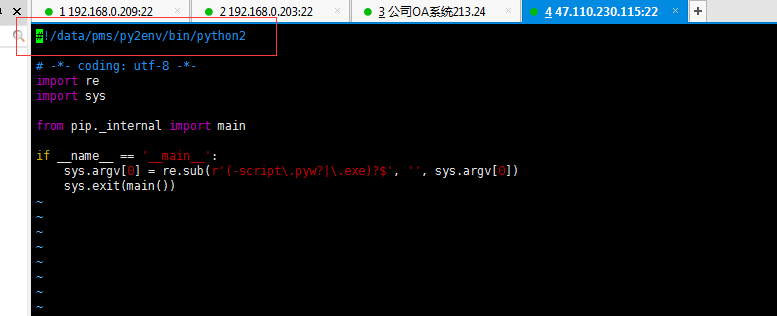
VIRTUAL\_ENV="your new venv path"

export VIRTUAL\_ENV

source venv/bin/activate



Vim pip #把第一行的路径修改为正确路径



pyvenv /path/to/new/virtual/environment

Python虚拟环境可以为项目创建相互独立的开发环境，也就是你可以为每个项目安装各自使用依赖模块。

使用虚拟环境可以很好的解决上面A、B项目遇到的问题：为A、B项目分别创建虚拟环境，然后在各自的虚拟环境中安装不同的tensorflow版本。

使用虚拟环境需要借助virtualenv或pyvenv，它们的使用非常简单。

安装virtualenv、pyvenv

如果你使用Python 2，你可以使用pip安装virtualenv：

$ sudo pip install virtualenv

如果你使用Python 3，它默认安装了pyvenv。

virtualenv和pyvenv的使用方法类似。由于pyvenv是较新的工具，本帖以它为例。

创建一个存放虚拟环境的目录：

$ mkdir python-env

$ cd python-env

创建一个虚拟环境：

$ pyvenv env\_A

上面命令在当前目录创建了env\_A目录，目录结构：

├── bin

│ ├── activate

│ ├── activate.csh

│ ├── activate.fish

│ ├── easy\_install

│ ├── easy\_install-3.5

│ ├── pip

│ ├── pip3

│ ├── pip3.5

│ ├── python -> python3.5

│ ├── python3 -> python3.5

│ └── python3.5 -> /opt/local/Library/Frameworks/Python.framework/Versions/3.5/bin/python3.5

├── include

├── lib

│ └── python3.5

│ └── site-packages

└── pyvenv.cfg

使用创建的虚拟环境env\_A：

$ source env/bin/activate

提示符中包含：

(env\_A) $

现在你通过pip安装的包都会安装这个虚拟环境中；导入（import）模块时也只能使用这个虚拟环境中的模块。

如果要退出env\_A虚拟环境，执行：

(env\_A) $ deactivate

虚拟环境的工作原理

主要就是操作环境变量。

Python在非虚拟环境下的路径：

$ which python

/usr/bin/python

激活虚拟环境再次查看Python路径：

$ source env/bin/activate

(env) $ which python

/Users/snail/python-env/env\_A/bin/python

$PATH 环境变量在激活前后不一样了：

$ echo $PATH

/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:

$ source env/bin/activate

(env) $ echo $PATH

/Users/snail/python-env/env\_A/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:

你可以看看 sys.prefix 和其他Python系统环境变量。