EI-PaaS上部署Python Web-App

Date： 09 / 17/ 2017

|  |  |  |
| --- | --- | --- |
| Approved by | Checked by | Prepared by |
| Baihui.cai |  | YuFeng.Wu |

目录

[1 教程环境 3](#_Toc493530103)

[2 Python开发环境 3](#_Toc493530104)

[2.1 前提 3](#_Toc493530105)

[2.2 Python安装环境 3](#_Toc493530106)

[2.2.1 环境检查 3](#_Toc493530107)

[2.2.2 下载 3](#_Toc493530108)

[2.2.3 Python环境变量添加 4](#_Toc493530109)

[2.3 Django安装环境 5](#_Toc493530110)

[2.3.1 Django安装环境检查 6](#_Toc493530111)

[2.3.2 Django下载安装 6](#_Toc493530112)

[2.3.3 pip工具安装 6](#_Toc493530113)

[2.4 总结 8](#_Toc493530114)

[3 Cloud Foundry工具CLI安装 8](#_Toc493530115)

[3.1 CLI安装 8](#_Toc493530116)

[3.2 CF命令检查 9](#_Toc493530117)

[3.2.1 登录 9](#_Toc493530118)

[3.2.2 CF常用命令 10](#_Toc493530119)

[4 Python web开发 11](#_Toc493530120)

[4.1 环境 11](#_Toc493530121)

[4.2 示例开发 11](#_Toc493530122)

[4.3 工程配置 17](#_Toc493530123)

[4.3.1 Buildpack 17](#_Toc493530124)

[4.3.2 配置 17](#_Toc493530125)

[5 利用CLI上传项目 20](#_Toc493530126)

[5.1 登录 20](#_Toc493530127)

[5.2 上传 20](#_Toc493530128)

# 教程环境

本教程环境：win10 + python2.7

、

# Python开发环境

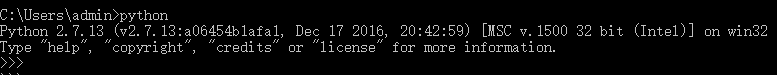
# 前提

本教程供环境搭建参考，可单独拿出来检查本地环境是否安装完整。

# Python安装环境

# 环境检查

首先检查本地是否已经安装python: 进入windows下cmd界面，若出现类似于以下界面：

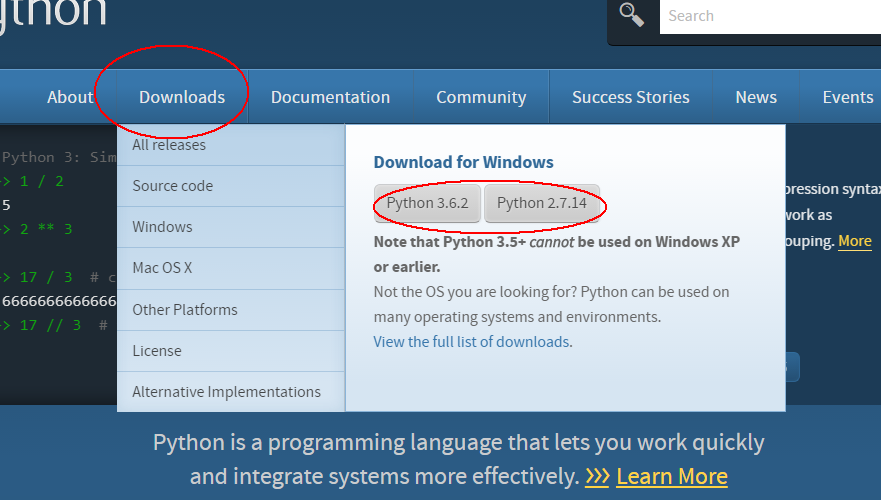


则本地已安装Python2.7.13（或其它版本），可跳过以下步骤。

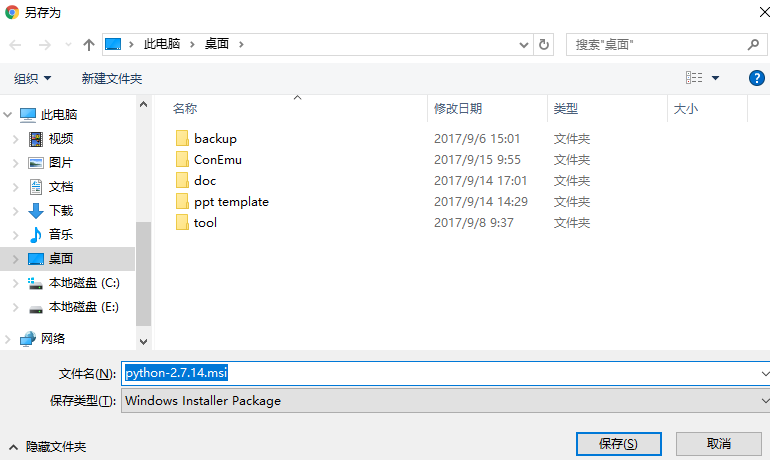
# 下载

若系统环境没有安装python，可进入Python官网：<http://www.python.org/> ，进行python下载。

点击图中红圈处，选择python版本



会弹出弹窗进行路径选择

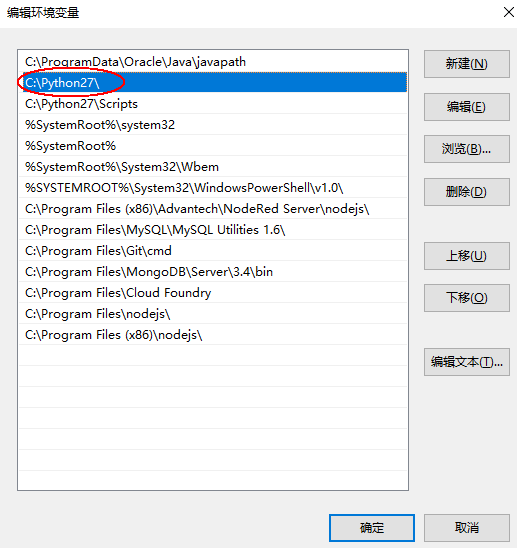


点击之后等待下载，下载完成之后，按照提示在电脑中安装python。注意要将python加入环境变量才能在命令行中执行python命令。

# Python环境变量添加

环境变量添加可通过 右键我的电脑—属性—高级系统设置—环境变量 进行设置

点击右侧新建，将python安装目录添加进去

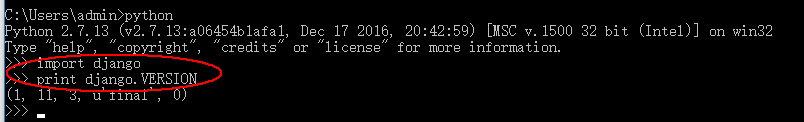


安装完python及配置好环境变量后，可通过2.2.1中步骤来检查python是否成功安装。

# Django安装环境

Django是python的一个开源框架，便于web开发者进行web开发，详情可见django官方文档：https://www.djangoproject.com/

# Django安装环境检查

进入windows下cmd界面，输入python，打开python解释器后输入以下命令行：

若能读到图中信息，说明django环境需求已完成。

# Django下载安装

如果提示Django未安装，在cmd窗口中键入

pip install Django

通过pip工具来安装Django，如图



安装完之后再通过前文步骤来检查django是否成功安装。

若提示系统环境中未安装pip工具，请参考2.3.3

# pip工具安装

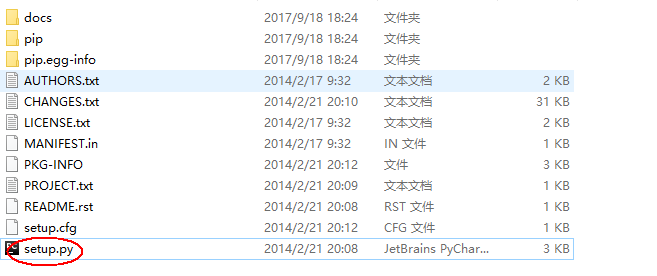
pip是python的包管理工具，可以帮助我们很轻易地安装或卸载包。

进入链接：

<https://pypi.python.org/packages/source/p/pip/pip-1.5.4.tar.gz#md5=834b2904f92d46aaa333267fb1c922bb>

下载得到pip-1.5.4.tar.gz

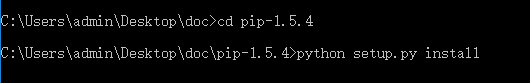
解压到自己指定的文件夹，可以看到，其中有一个setup.py文件。



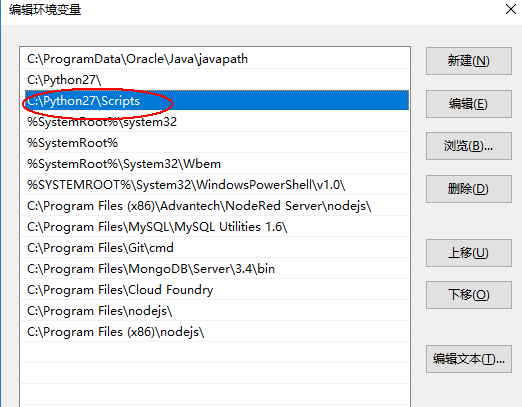
进入cmd，切换到该文件夹的目录下

执行以下语句：

python setup.py install



随后，添加pip的环境变量，过程同2.2.3，将pip的安装目录添加进去，默认为如图所示



到这里完成了pip的安装，可以继续2.3.2的步骤。

# 总结

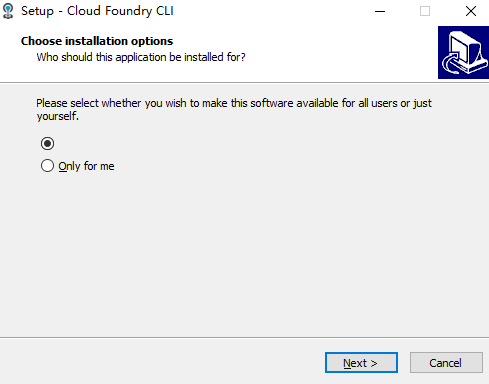
本节内容检查了python环境与本教程所用框架django环境，开发者亦可通过其它框架如flask、tornado等进行web app开发，环境检查的步骤也基本一致。值得注意的是，在本地开发的时候，这些环境（或者称之为库）都是需要依赖的，在服务器部署的时候我们需要配置好这些。在后续工作中，我们需要指定好这些库的版本来让服务器帮我们自动部署好这些环境，自动部署正是Cloud Foundry的核心功能之一。在CF中，python环境依赖包称为buildpack，后续开发过程中会继续提及。

# Cloud Foundry工具CLI安装

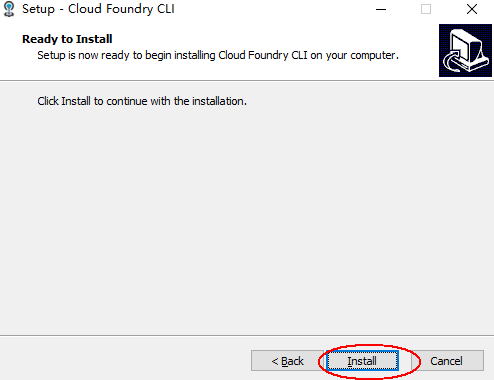
# CLI安装

进入以下链接：https://docs.cloudfoundry.org/cf-cli/install-go-cli.html ，下载之后，解压到文件夹，运行其中的cf\_install.exe。





点击next



点击install后，安装程序自动执行。

到这里，CF安装完成

# CF命令检查

# 登录

打开cmd窗口

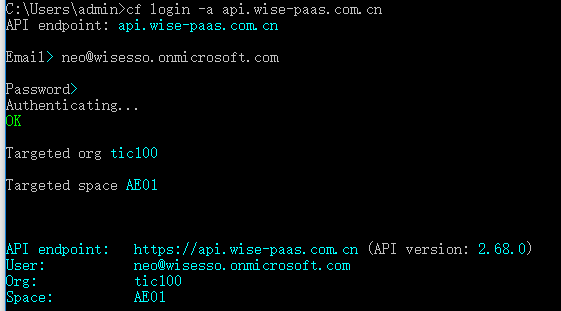
输入cf login –a api.wise-paas.com.cn



在email后输入帐号



接着输入密码（注意，密码输入无界面显示）



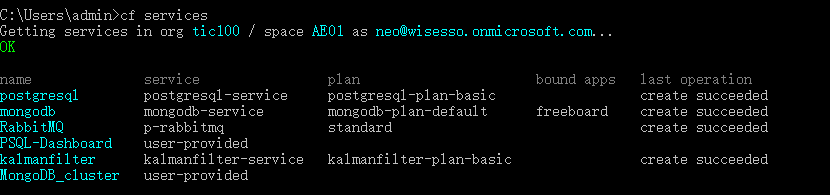
输入后按回车键，若有以上界面，则证明登录成功。

# CF常用命令

在CF环境下，键入cf apps检查该space下所有应用。



键入cf services 检查该space下所有服务。



更多CF命令可通过查看文档或输入cf help 或cf help –a 查看。

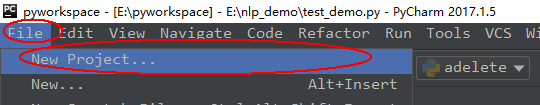
# Python web开发

# 环境

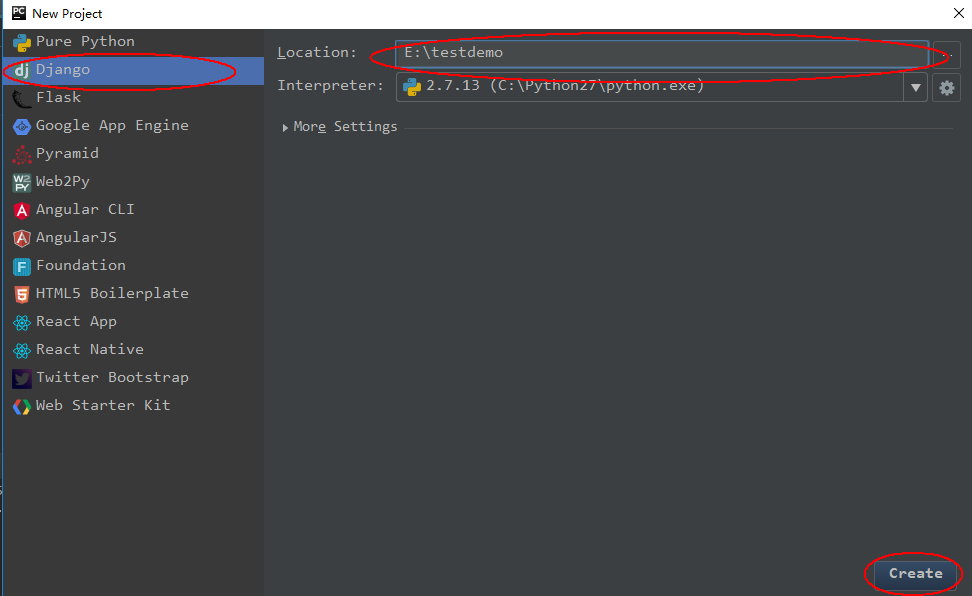
使用Python2.7 以及 Django1.11版本，利用JetBrains的Pycharm进行开发

# 示例开发

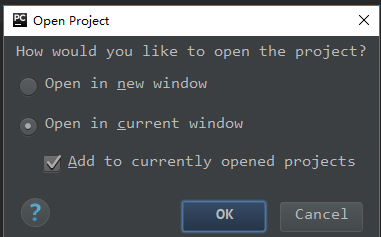
创建一个新的project



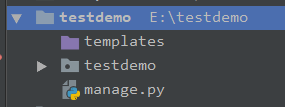
点击new project后可以得到下图，



选择右侧django项目，（diango已在前文有介绍），填入project名称。（本示例命名为testdemo）

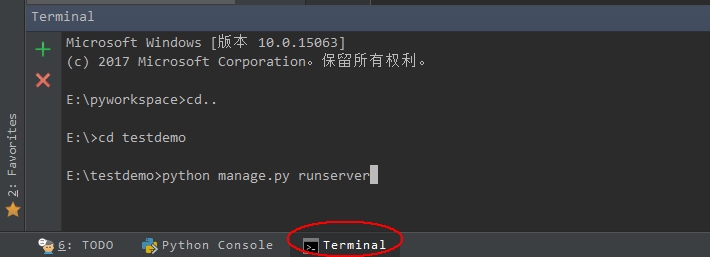


点击OK，可以看到testdemo工程下的目录架构



注意，这里面的manage.py是django提供的配置工具，后续我们会利用它来做很多工作。

打开pycharm底端terminal界面：

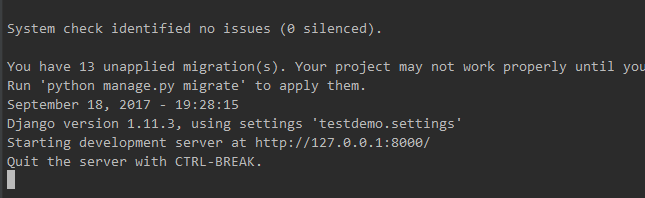


切换到当前目录

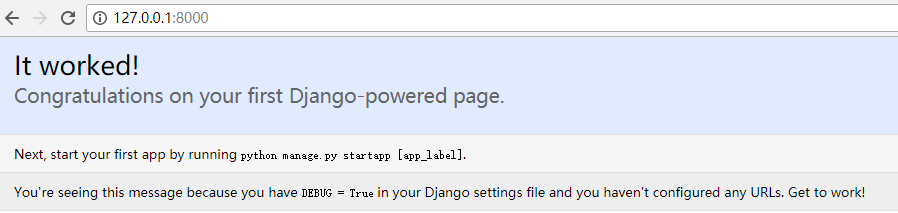
执行以下命令：

python manage.py runserver

这个时候，完成了第一步工作



可在浏览器输入127.0.0.1：8000来查看工程是否执行



项目成功运行。

接下来，我们将访问界面改为hello world

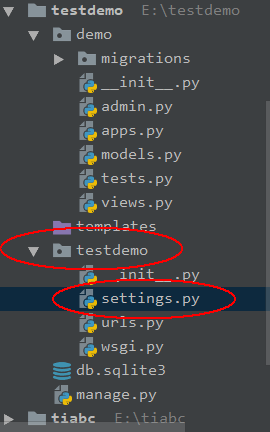
同样，进入terminal界面

执行命令：

Python manage.py startapp demo

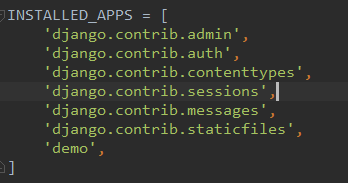
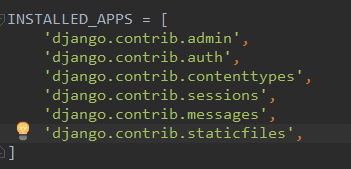


回头查看项目结构，已经发生改变

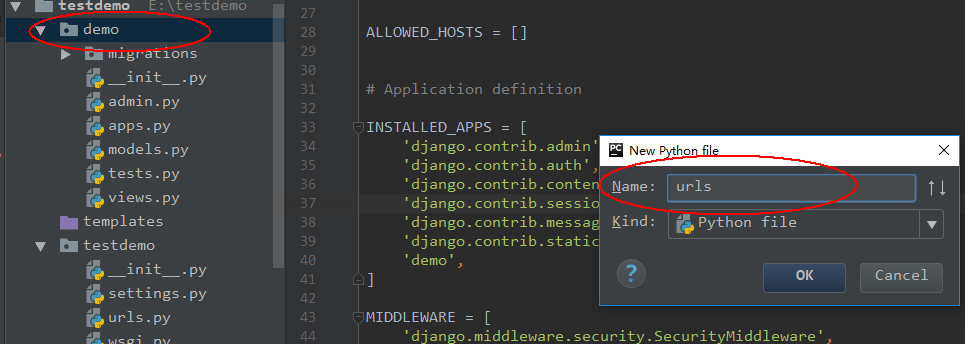


进入testdemo下的setting.py（这是工程的配置文件）

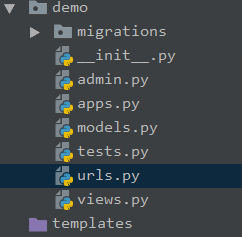
找到 INSTALLED\_APPS 设置项，将 demo应用添加进去。



**在 demo应用的目录下创建一个 urls.py 文件**



这时你的目录看起来是这样：



在demo\urls.py里写入如下代码：

blog/urls.py

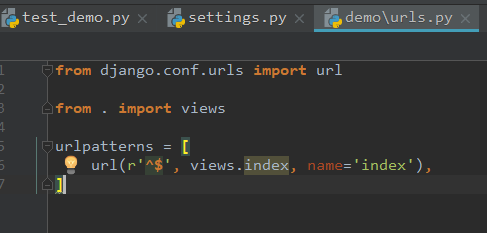
**from** **django.conf.urls** **import** url

**from** **.** **import** views

urlpatterns = [

url(r'^$', views.index, name='index'),

]

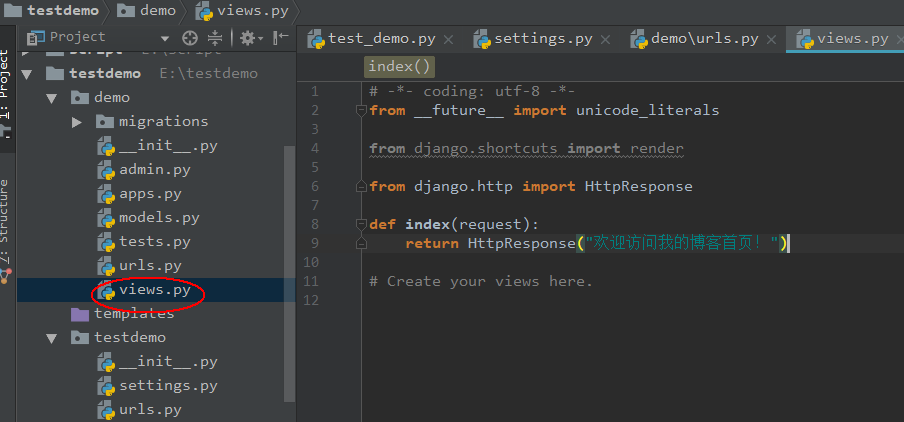


在demo\views.py下写入如下代码：

**from** **django.http** **import** HttpResponse

**def** index(request):

**return** HttpResponse("helloworld")



我们前面建立了一个 urls.py 文件，并且绑定了 URL 和视图函数 index，但是 Django 并不知道。Django 匹配 URL 模式是在 testdemo\ 目录（即 settings.py 文件所在的目录）的 urls.py 下的，所以我们要把 demo 应用下的 urls.py 文件包含到 testdemo\urls.py 里去，打开这个文件看到如下内容：

**from** django.conf.urls **import** url  
**from** django.contrib **import** admin  
  
urlpatterns = [  
 url(r'^admin/', admin.site.urls),  
]

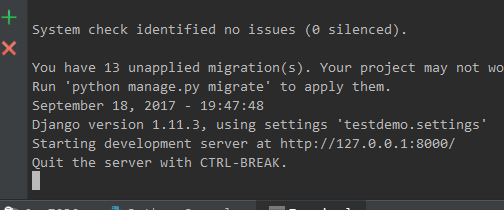
将第一行代码加入include库

将urlpatterns加入一个url

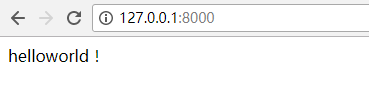
修改为：

**from** django.conf.urls **import** url,include  
**from** django.contrib **import** admin  
  
urlpatterns = [  
 url(r'^admin/', admin.site.urls),  
 url(r'',include('demo.urls'))  
]

这个时候，再次在terminal下运行python manage.py runserver



在浏览器输入127.0.0.1：8000



到此，本地helloworld实例工程建立及测试已完成

# 工程配置

# Buildpack

在前面我们提到，一个python web 工程运行需要依赖于相对应的运行环境。上传项目的时候，Cloud Foundry在通过Router层接收到项目后，Cloud Foundry中Blob Store内的Buildpack将会对应用程序进行检查，它为工程项目提供运行所需的框架和环境支持，以及确定依赖关系，下载相关的第三方库，从而运行我们的项目。

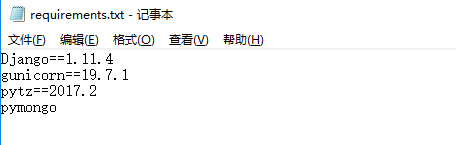
Reference：<http://docs.cloudfoundry.org/buildpacks/>

# 配置

**requirement.txt**

在本地的该web项目目录下，建立一个requirement.txt，在里面输入该项目所依赖的所有库和服务器。

Django==1.11.4  
gunicorn==19.7.1  
pytz==2017.2



所实例代码使用的django版本为1.11.4，指定gunicorn为代理服务器。若需要nginx，也需要在此txt中进行指定。其他的库如pymongo等则是特定项目所依赖的不同的库。

requirement.txt起的作用：

作用：指定了web项目所依赖的库，上传之后服务器将会自动下载这些库。

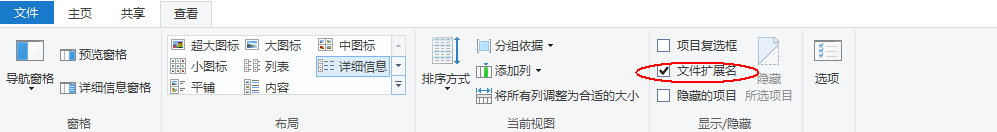
**Procfile**

本地测试的过程，能注意到需要在终端环境下输入python manage.py runserver来进行服务运行。在服务器上，如何让服务器在运行程序时自动启动这个命令呢？这里就需要建立一个procfile来指定gunicorn运行这个项目。

建立一个txt文件，输入以下内容：

web: gunicorn testdemo.wsgi --log-file -

然后保存，将名字命名为Procfile，并不加后缀名。若无法去掉后缀名，可根据下图红圈范围来将后缀展开并消除。





**manifest.yml**

建立一个txt文件，输入以下内容：

**applications:**- **name:** testhello  
 **memory:** 128M  
 **instances:** 1  
 **host:** fortest  
 **buildpack:** python\_buildpack

保存时，重命名为manifest.yml，记住后缀是yml，从而得到一个yml文件。

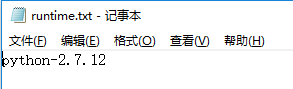
作用：这个文件中，testhello为开发者指定的app的名字(这里命名为fortest)，memory为项目所占内存。Instances为产生的实例数量，这里指定为1。Host为域名，buildpack则指定了CF系统选择哪个buildpack来进行程序的环境配置。



runtime.txt

建立一个txt文件，输入：

python-2.7.12



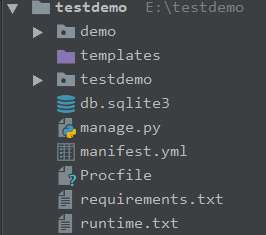
作用：

指定python版本。

这里可能有疑问产生：为何指定python为2.7.12，但本地是2.7.14或2.7.13？

解释：测试时cloud foundry上所支持的python版本为2.7.12

**以上所有文件都存在于工程的根目录下，目录结构如图：**



Setting.py中修改：

打开setting.py

找到

ALLOWED\_HOSTS = []

修改为

ALLOWED\_HOSTS = ['\*']

找到

STATIC\_URL = '/static/'

添加代码

STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')

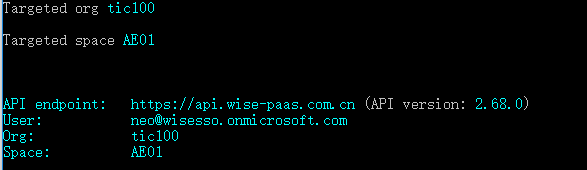
到此，完成上传前对工程的配置。

# 利用CLI上传项目

# 登录

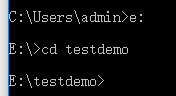
在前文已对CLI的简单使用做了介绍，在这里利用获取到的帐号密码登录CF

进入cmd，按3中步骤输入网址、帐号、密码



# 上传

切换到上传的项目的目录



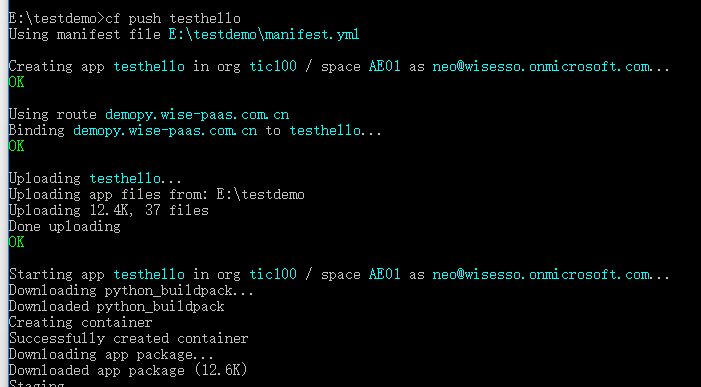
执行命令：

cf push testhello

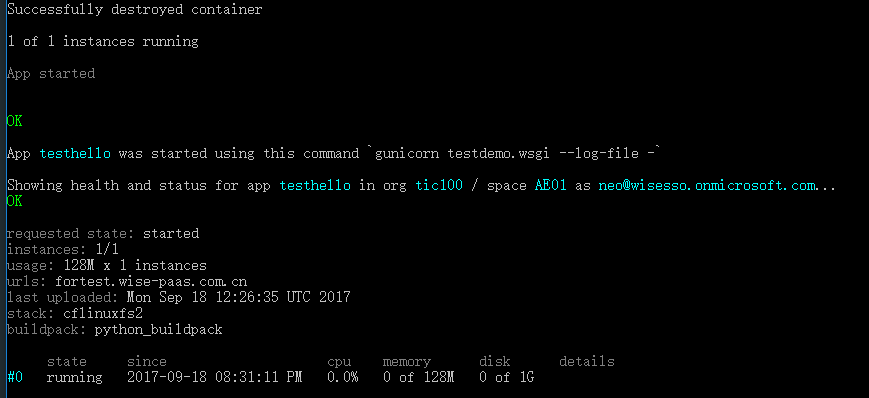
注：这里的testhello是自己自定义的名字



等待执行



看到如下界面，意味着部署成功：



浏览器键入绑定的URL：fortest.wise-paas.com.cn，可以看到我们上传的工程

