Python 运行环境（Centos7.4 64 位）

文档更改记录

日期	修改人	版本
2018 年 9 月 10 日	康展云计算	V1.0
2018 年 11 月 27 日	康展云计算	V1.1

1 镜像基本介绍

1.1 镜像软件环境

（1）操作系统：centos 7.4 64 位。

（2）python 运行环境 :python 多版本自由切换 ,mysql 5.6.36 ,nginx 1.12.2 ,pyenv 1.2.3 ,ipython 5.6.0，pip 10.0.0，git 1.8.3.1，django 1.11.12。

1.2 镜像安装说明

mysql，nginx 使用源码编译安装，其他 python 集成的环境使用 centos 最新 yum 源进行安装。

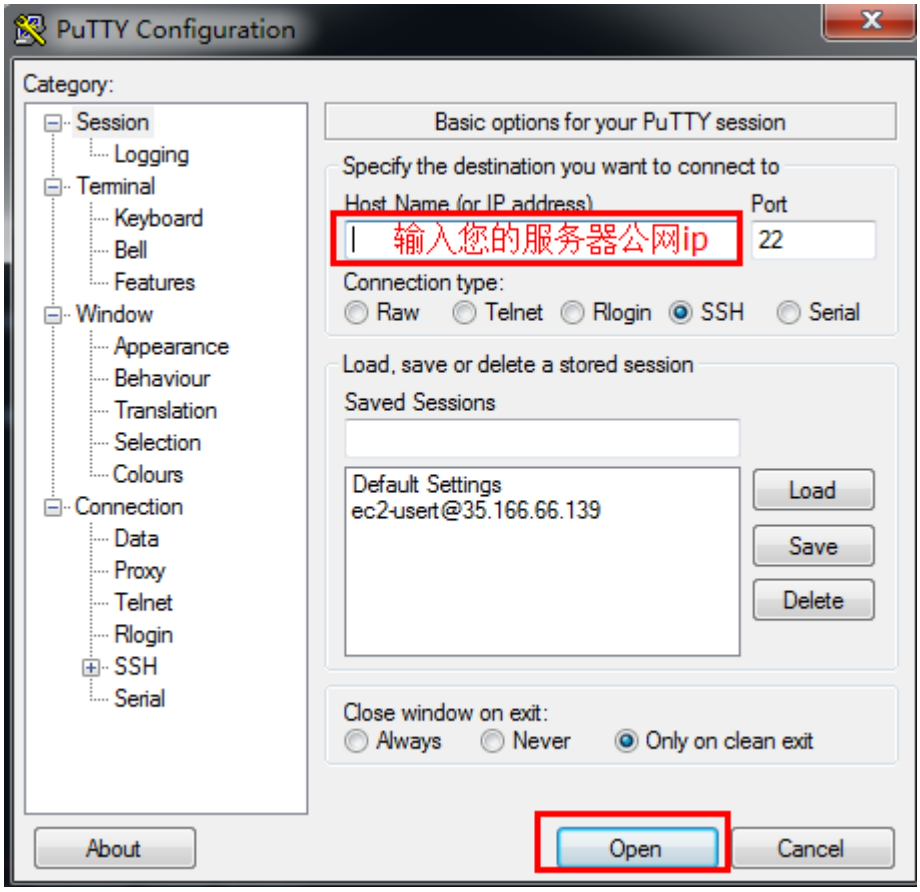
2 软件使用介绍

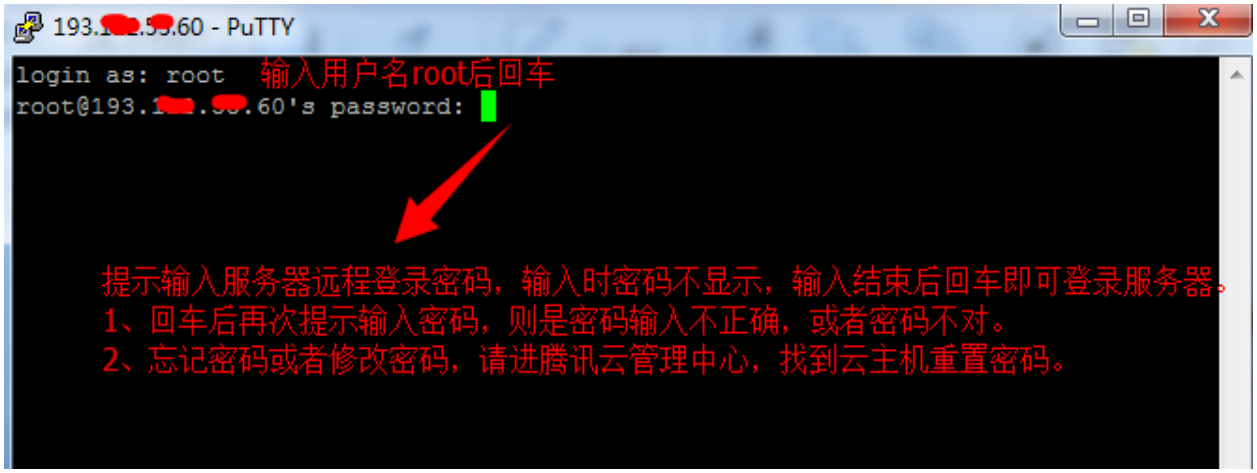
2.1 关于 mysql 的使用说明

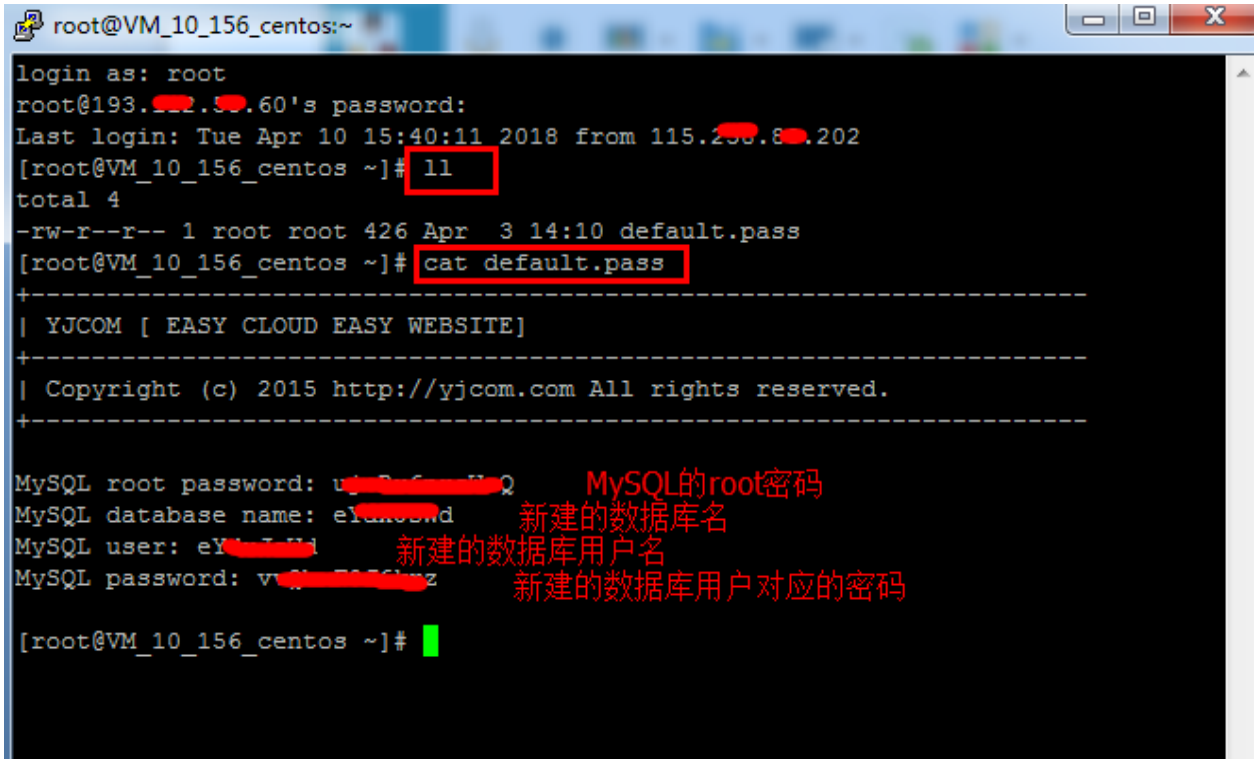
2.1.1 获取 mysql 的权限

在您的服务器里保存着一份随机生成的 mysql 权限，可通过 putty 远程登录服务器查看。

(putty 是登录 linux 操作系统的远程管理工具，可以百度搜 putty，下载到本地电脑后直接双击运行 putty.exe 即可)，如下：


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
输入您的服务器公网ip
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
ec2-usert@35.166.66.139
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel


PuTTY
login as: root
输入用户名root后回车
password:
提示输入服务器远程登录密码，输入时密码不显示，输入结束后回车即可登录服务器。
1、回车后再次提示输入密码，则是密码输入不正确，或者密码不对。
2、忘记密码或者修改密码，请进腾讯云管理中心，找到云主机重置密码。



如上，用 root 账号登陆服务器后：

输入命令英文 ll 回车后可以看到 default.pass 这个 mysql 权限文件。

输入命令 cat　default.pass　即可查看到您的 mysql 数据库的权限信息。

2.1.2 mysql 安装路径

名称	路径地址
mysql5.6	/usr/local/mysql/
mysql5.6 的 data 路径	/usr/local/mysql/data/
mysql5.6 配置文件 my.cnf	/etc/my.cnf

mysql5.6 日志文件 mysqld.log	/var/log/mysqld.log

2.1.3 mysql 允许远程连接

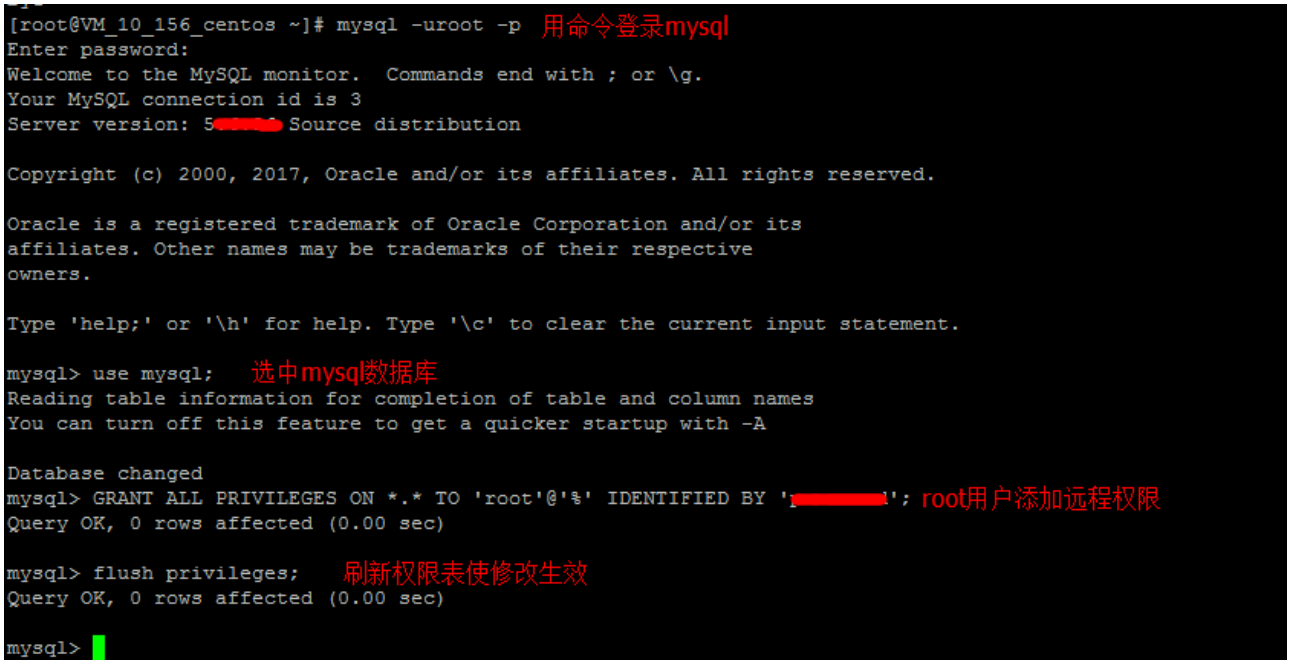
考虑 mysql 的相关安全性，默认情况下只允许服务器本地连接。如需服务器外部进行连接 mysql，可通过以下方式进行操作：

（1）使 root 用户可以远程连接：

将 host 字段的值改为%就表示在任何客户端机器上能以 root 用户登录到 mysql 服务器，建议在开发时设为%。

```
mysql> use mysql;
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'password';
mysql> flush privileges;
```

注：请将 password 替换为 mysql 的 root 用户的密码。



（2）使特定用户从特定主机远程连接：

例如：mysql 特定用户名：username

mysql> use mysql;

mysql> GRANT ALL PRIVILEGES ON *.* TO 'username'@'ip' IDENTIFIED BY 'password';

mysql> flush privileges;

注： 请将 username 替换为 mysql 的数据库用户名， ip 替换为特定主机的 ip，password 替换为 mysql 的 username 用户的密码。

推荐使用第二种方法，特定数据库对特定用户和特定主机授权远程连接权限，此方式更加安全。

注 :请找到对应 mysql 云主机 ,进安全组 ,添加入站规则 ,确保安全组开放 mysql 默认远程端口 3306。

2.1.4 mysql 常用命令

服务启动，停止，重启操作

mysql: systemctl (start|stop|restart) mysqld.service

2.2 账号密码安全提醒

（1）mysql 修改 root 密码方法：

服务器命令行输入 mysql -uroot -p 回车，输入 root 用户对应的密码（密码不显示）

```
[root@V[redacted]_centos ~]# mysql -uroot -p
Enter password:
```

进入 mysql 管理命令行

```
[root@VM_16_6_centos ~]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.26 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

命令行输入 use mysql; 回车选择 mysql 数据库

```
mysql> use mysql;
Database changed
```

命令行输入 update user set password=password('123woheGE') where user='root'; 回车

（其中 123woheGE 修改为你自己需要重新设置的 root 密码）

```
mysql> update user set password=password('123woheGE') where user='root';
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql>
```

命令行输入 flush privileges; 回车刷新权限表，至此新密码设置生效。

```
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

（2）关闭 mysql 方法

服务器命令行直接输入命令 systemctl stop mysqld.service 将关闭 mysql 数据库

```
[root@V[redacted]s ~]# systemctl stop mysqld.service
[root@V[redacted]s ~]#
```

2.3 关于 pyenv 的使用说明

pyenv 是 python 多版本自由切换管理工具。服务器内默认系统自带的 python 版本为 2.7.5，当需要

使用到其他 python 版本时，比如 python3.x，又不影响系统自带 python，即多版本 python 共存。可以使用 pyenv 来安装管理 python。

2.3.1 pyenv 常用命令

pyenv install --list　　//列出当前可用的 python 版本

pyenv install 3.5.5　　//安装 3.5.5 版本的 python，此镜像已经安装 python 3.5.5

pyenv rehash　　//更新数据库，在安装 Python 或者其他带有可执行文件的模块之后，需要对数据库进行更新

pyenv versions　　//列出目前本机上已经安装有哪些版本的 python

pyenv version　　//查看当前目录 python 版本

pyenv local 3.5.5　　//切换 python 版本，在当前目录及子目录下改变 python 版本（这个设置在我们切换到其它目录就失效）

pyenv local --unset　　//取消改变，恢复为 system 版本的 python

pyenv local system　　//回到系统默认的 python 版本

pyenv global 3.5.5　　//全局改变 python 版本【强烈不建议使用】

pyenv uninstall 3.5.5　　//卸载 3.5.5 版本的 python（注意卸载后，还要去 /root/.pyenv/version 删掉对应的行 3.5.5，不然使用 pyenv versions 时候还会有警告）

pyenv update　　//更新 pyenv 版本

python -V　　//查看当前 python 版本

2.3.2 使用 pyenv 管理 virtualenv

pyenv-virtualenv: 管理虚拟环境的插件 ,pyenv 解决的是同一个系统中不同版本的 python 并存的问题 ,而 pyenv-virtualenv 解决的是不同项目所依赖的软件包之间可能产生冲突的问题。在实际使用 python 的过程中，很容易出现这样的问题：

通过 pip 安装软件包 A 时安装了 A 所依赖的软件包 B；之后又通过 pip 安装软件包 C 时再次安装了 B 并将之前的覆盖，但是因为 C 和 A 所依赖的 B 版本不同，安装完 C 后导致 A 无法运行。

pyenv-virtualenv 通过为每个项目设置独立的虚拟环境(目录)来解决上述问题。

（1）使用 pyenv-virtualenv 创建虚拟环境：

pyenv virtualenv 3.5.5 proj1 //将创建一个名为 proj1 的虚拟环境(目录)，并且将 python 3.5.5 对应的 bin 和 lib 复制到该环境中。当该虚拟环境被激活后，所有的 python 操作都只在该环境中进行，从而和其它 python 内容隔离。（3.5.5 的 python 需要事先安装好）

pyenv local 3.5.5/envs/proj1 #切换到新建的 proj1 环境下（必须事先存在这个 proj1），当前目录及子目录就是 proj1 环境，（这个设置在我们切换到其它目录就失效）

（2）用命令手动激活和退出：

pyenv activate proj1 //进入 proj1 环境，(则服务器当前所有目录都是 proj1 的环境)

pyenv deactivate //退出 proj1 环境

pyenv uninstall proj1 或 pyenv virtualenv-delete proj1 //删除 proj1 这个虚拟环境

2.4 关于 ipython 的使用说明

ipython 支持 Python2.7 版本或者 3.3 以上的版本。ipython 是一个 python 的交互式 shell，比默认的 python shell 好用得多，支持变量自动补全，自动缩进，支持 bash shell 命令，内置了许多很有用的功能和函数。

2.4.1 ipython 的主要功能

1.运行 ipython 控制台

2.使用 ipython 作为系统 shell

3.使用历史输入(history)

4.Tab 补全

5.使用%run 命令运行脚本

6.使用%timeit 命令快速测量时间

7.使用%pdb 命令快速 debug

8.使用 pylab 进行交互计算

9.使用 IPython Notebook

2.5 关于 pip 的使用说明

pip 是一个 python 包管理工具，主要用于安装和管理 pypi 上的软件包。

2.5.1 pip 常用命令

```
# pip -V                                    //查看 pip 版本
```

```
# pip install package_name                  //安装名为 package_name 的包
# pip show –files package_name              //查看已安装软件包 package_name 的信息
# pip list –outdated                        //检查需要更新的软件包
# pip install --upgrade package_name        //升级软件包 package_name
# pip uninstall package_name                //卸载软件包 package_name
```

2.6 关于 nginx 的使用说明

Nginx 是高性能 http 和反向代理服务器，具有轻量级高并发特点。

2.6.1 nginx 安装路径

名称	安装路径
nginx1.12.2	/usr/local/nginx/
nginx 的配置文件 nginx.conf	/usr/local/nginx/logs/nginx.conf
nginx 的访问日志文件 nginx.log	/usr/local/nginx/logs/access.log
nginx 的错误日志文件 error.log	/usr/local/nginx/logs/error.log

2.6.2 nginx 反向代理 django 配置

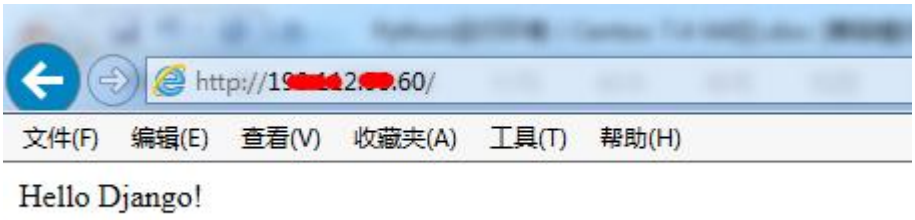
```
# vim /usr/local/nginx/logs/nginx.conf      //打开 nginx 配置文件
```

```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        index  index.html index.htm ;
        proxy_pass   http://127.0.0.1:8000;
    }


    #error_page  404              /404.html;
```

可以绑定域名

nginx转发django，django启动的地址和端口

结合 **3 案例部署**后，浏览器访问公网 ip 如下：



注：proxy_pass http://127.0.0.1:8000; 这行请结合 3.1.2 节启动 django 的形式修改。

可以停用 nginx，不使用反向代理，直接用 3.1.2 节（2）方式启动 django。

2.6.3 nginx 常用命令

服务启动，停止，重启操作

nginx: systemctl (start|stop|restart) nginx.service

3 案例部署

3.1 django 使用说明

Django 是基于 Python 开发的免费的开源网站框架，也是 python web 开发中重量级的 web 框架，可以用于快速搭建高性能并且优雅的网站！

3.1.1 django 项目部署

（1）创建名为 project 的项目，比如已经创建/yjdata/www/目录，进此目录下执行如下命令：

django-admin startproject project　　//创建项目 project

tree　　// 进/yjdata/www/project 下执行此命令查看目录结构

（2）创建名为 app1 的应用：

可以在 project 下直接建立视图文件，但是为了应用结构的清晰不建议这样做，正确方法是在容器根目录/yjdata/www/project/使用如下命令建立一个应用，一个项目可以包括多个应用：

django-admin startapp app1 或者 python manage.py startapp app1

这样在容器下就会出现一个 app1 的目录，结构如下：

```
[root@VM_10_156_centos protject]# cd app1/
[root@VM_10_156_centos app1]# tree
.
|-- admin.py
|-- apps.py
|-- __init__.py
|-- migrations
|   `-- __init__.py
|-- models.py
|-- tests.py
`-- views.py

1 directory, 7 files
```

（3）配置 views.py

vim /yjdata/www/project/project/app1/views.py　　//编辑应用 app1 的 views.py 视图文件

补充如下代码，打印一行简单的输出：

from django.http import HttpResponse

def index(request):

　　return HttpResponse("Hello Django!")

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello Django!")
```

（4）添加应用：

　　由于项目管理着所有的应用，每个应用都要添加到项目的应用列表中，打开

/yjdata/www/project/project 下的 settings.py 文件，在 INSTALLED_APPS 下添加：

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app1',
]
```

（5）配置 url

vim /yjdata/www/project/project/urls.py

头部追加一行 from app1 import views as app1_views，这个 app1_views 是我们自己定义的导入的视图名，然后给 urlpatterns 这个列表添加一项：url(r'^$', app1_views.index),

```
from django.conf.urls import url
from django.contrib import admin
from app1 import views as app1_views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', app1_views.index),
]
```

3.1.2 启动 django

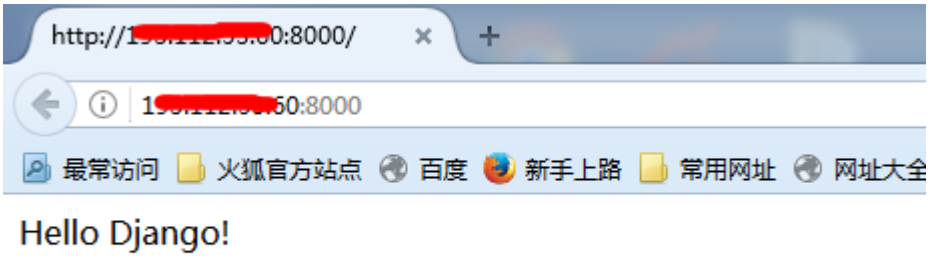
进入/yjdata/www/project/下，有多种方式启动，如下命令：

（1）默认以 127.0.0.1 服务器本地监听 8000 端口启动，可以和 nginx 反向代理结合使用（请看 2.5.2 节）。

python manage.py runserver

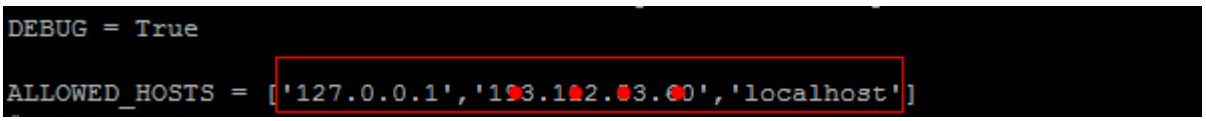
（2）允许外部访问，并监听 8000 端口。

python manage.py runserver 0.0.0.0:8000　//浏览器输入 http://服务器公网 ip:8000 能正常显示。



vim /yjdata/www/project/project/settings.py

注：为了使 django 允许外部访问，编辑项目配置文件 settings.py，把 ALLOWED_HOSTS = [] 修改为 ALLOWED_HOSTS = ['127.0.0.1','服务器公网 ip','localhost']



注：监听的端口 8000 可以改成任何服务器没有被监听的其他端口，比如 80。并且请确保腾讯云主机对应的安全组放通 8000 等需要使用的端口。

3.1.3 关闭 django

通过 Ctrl+C 快捷键可以终止 django 服务。当项目文件发生修改时，django 服务器会自动重启，不需要手动关闭后启动。

3.1.4 查看 django 版本

python -c "import django; print(django.get_version())" //直接在服务器命令行输入此行命令

3.2 配置 django 模板

（1）在之前的 views.py 视图中再定义一个方法。

vim /yjdata/www/project/project/app1/views.py //编辑视图文件 views.py

代码如下：

```
def home(request):
    return render(request, 'home.html')
```

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello Django!")
def home(request):
    return render(request, 'home.html')
```

（2）render 方法的作用就是调用对应的视图模板也就是 home.html，在 app1 目录下执行下面命令：

mkdir templates

cd templates

vim home.html

在 django 加载 app1 应用时，会自动读取 templates 模板目录从而渲染里面的数据，home.html 就是一个网页模板，为了简单，没有添加动态的数据：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>home 视图</title>
</head>
<body>
    <h3>Hello Django Templates!</h3>
</body>
```

</html>

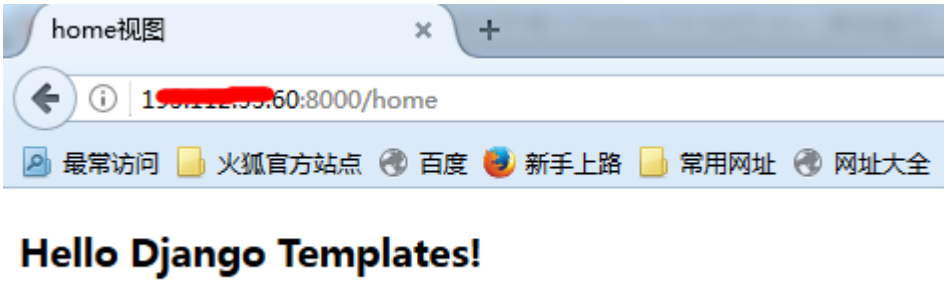
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>home视图</title>
</head>
<body>
    <h3>Hello Django Templates!</h3>
</body>
</html>
```

（3）设置访问 url

vim /yjdata/www/project/project/urls.py　　//编辑文件 urls.py，加如下代码：

```
from django.conf.urls import url
from django.contrib import admin
from app1 import views as app1_views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', app1_views.index),
    url(r'^home', app1_views.home, name='home'),
]
```

浏览器访问 http://服务器公网 ip:8000/home 就可以看到对应的响应



3.3 django 连接 mysql 配置

3.3.1 配置连接信息

（1）将默认数据库引擎 sqlite3 修改为 mysql

vim /yjdata/www/project/project/settings.py　　//编辑项目文件 settings.py

mysql 连接代码如下：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mytest',
        'USER': 'root',
        'PASSWORD': 'mysql',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

注：数据库名，数据库用户，数据库密码请对应修改为自己的。

```
#DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
#         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
#     }     //默认的sqlite3数据库配置注释掉
#}

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mytest',          数据库名
        'USER': 'root',            数据库用户
        'PASSWORD': 'mysql',       数据库密码
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

（2）需要将 pymysql 导入到 django 中

vim /yjdata/www/project/project/app1/__init__.py　　//编辑应用 app1 的文件__init__.py

加入如下代码：

import pymysql

pymysql.install_as_MySQLdb()

```
import pymysql
pymysql.install_as_MySQLdb()
```

注：一定要导入 pymysql，否则无法连接数据库。

（3）生成 django 自带 app 的数据库表

在容器/yjdata/www/project/下运行如下代码：

python manage.py migrate　　//应用到数据库

```
[root@VM_10_156_centos project]# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying sessions.0001_initial... OK
```

（4）登录 mysql 数据库查看对应数据库所有表

```
mysql> use eY[redacted]d; 选中数据库
Database changed
mysql> show tables;   显示对应数据库的所有表
+----------------------------+
| Tables_in_eYdxJsWd         |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
+----------------------------+
10 rows in set (0.00 sec)

mysql>
```

注：此时说明 django 链接 mysql 数据库没有问题。

3.3.2 如何创建数据库表

（1）定义实体，对应数据表，如下定义：

vim /yjdata/www/project/project/app1/model.py　//编辑 model.py

加入如下代码：

class StoreInfo (models.Model):

name = models.CharField(max_length=20, default='') #name 属性，字段

address = models.CharField(max_length=50, default="China") #address 属性，字段

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.db import models

# Create your models here.
class StoreInfo (models.Model):
    name = models.CharField(max_length=20, default='') #name属性，字段
    address = models.CharField(max_length=50, default="China") #address属性，字段
```

（2）生成数据库表

python manage.py makemigrations　　//当 model 改变后，会创建一个新的 model

```
[root@VM_10_156_centos protject]# python manage.py makemigrations
Migrations for 'app1':
  app1/migrations/0001_initial.py
    - Create model StoreInfo
[root@VM_10_156_centos protject]#
```

python manage.py migrate　　//应用到数据库

```
[root@VM_10_156_centos protject]# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, app1, auth, contenttypes, sessions
Running migrations:
  Applying app1.0001_initial... OK
[root@VM_10_156_centos protject]#
```

（3）登录 mysql 数据库查看对应数据库新生成的表

```
mysql> use eY[redacted]Wd;
Database changed
mysql> show tables;
+----------------------------+
| Tables_in_eYdxJsWd         |
+----------------------------+
| app1_storeinfo             |
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
+----------------------------+
11 rows in set (0.00 sec)

mysql>
```

新建的表已经生产，表里的内容可以用 mysql 命令去查看。

（4）查看 app1_storeinfo 表结构 ： name 字段和 address 字段

```
mysql> show columns from app1_storeinfo;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int(11)     | NO   | PRI | NULL    | auto_increment |
| name    | varchar(20) | NO   |     | NULL    |                |
| address | varchar(50) | NO   |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql>
```

此时，数据表就建立好了。

4 售后服务

镜像为我司技术研发制作，相关软件使用都经过反复测试验证。镜像使用过程中可能涉及到的问题，均在产品文档中提出，请仔细阅读。除镜像本身环境问题外，不包括任何人工服务。