安装python3的pip

**sudo apt-get install python3-pip**

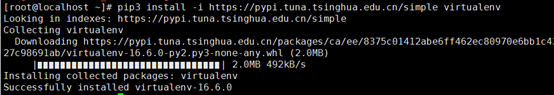
安装虚拟环境：

[**linux虚拟环境搭建**](https://www.cnblogs.com/shenjianping/p/10973197.html)

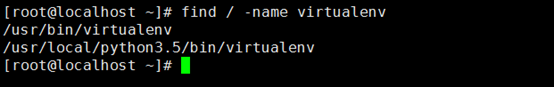
一、virtualenv的安装与使用

安装virtualenv sudo pip3 install virtualenv

pip3 install -i https://pypi.tuna.tsinghua.edu.cn/simple virtualenv

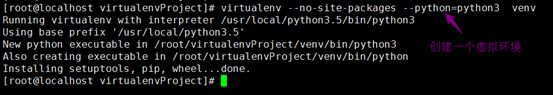


如果想查看安装位置



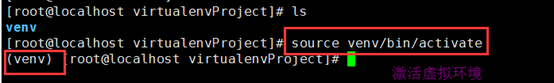
、创建虚拟环境

virtualenv --no-site-packages --python=python3 venv#创造无第三方包的干净环境，并且指定解释器是python3



、激活虚拟环境

source venv/bin/activate#进入虚拟环境



、退出虚拟环境

deactivate

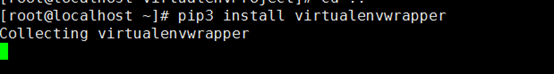
https://img2018.cnblogs.com/blog/1137363/201906/1137363-20190604131943515-1699543870.png

二、virtualenvwrapper的安装与使用

virtualenv能够创造独立的环境，但是它有一个很大的缺点，就是必须知道的摸个虚拟环境的目录，因为打开虚拟环境是用source+虚拟环境下bin目录的activate命令进行激活，这是相当费事的，由此引入virtualenvwrapper，统一管理虚拟环境目录，无需source命令。

安装

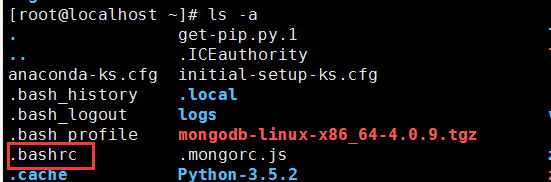
pip3 install virtualenvwrapper



、设置环境变量

在每次启动时就自动加载virtualenvwrapper

）打开家目录下的.bashrc 文件

1. 
2. （2）写入以下内容

export WORKON\_HOME=~/Envs #设置virtualenv的统一管理目录

export VIRTUALENVWRAPPER\_VIRTUALENV\_ARGS='--no-site-packages' #创建无第三方包的环境

export VIRTUALENVWRAPPER\_PYTHON=/usr/local/python3.5/bin/python3 #指定python解释器

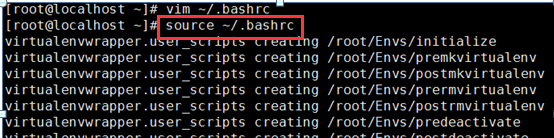
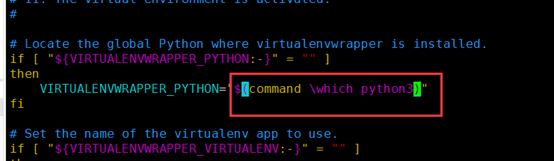
source /usr/local/python3.5/bin/virtualenvwrapper.sh #执行virtualenvwrapper安装脚本

export WORKON\_HOME=~/Envs

source /usr/local/bin/virtualenvwrapper.sh

VIRTUALENVWRAPPER\_PYTHON=/usr/bin/python3

写入以后执行：

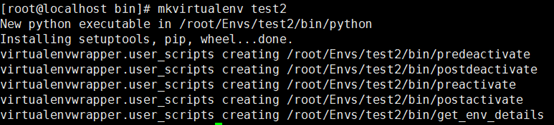
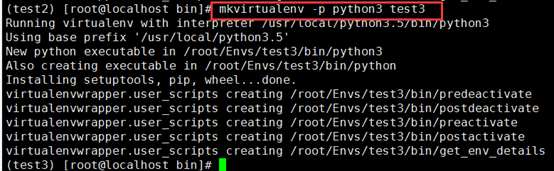
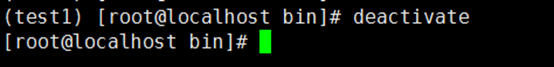
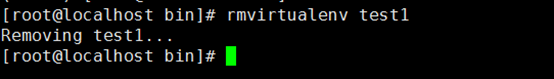
1. sudo Vim~/.bashrc
2. source ~/.bashr
3. 
4. 注意：
5. 在写入的的内容virtualenvwrapper.sh中规定了默认的python解释器，可以进行修改：
6. 

当然在创建环境时也是可以指定的。

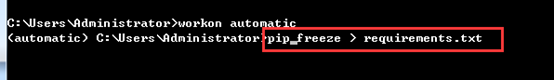
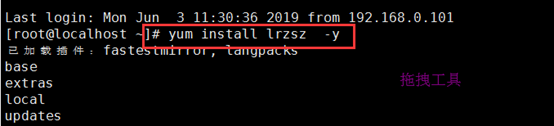
使用virtualenvwrapper

（1）创建虚拟环境

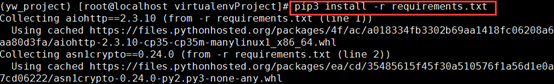
mkvirtualenv test2

1. 
2. 这是用默认的python解释器，如果需要指定，可以使用以下命令
3. mkvirtualenv -p python3 test3
4. 
5. （2）虚拟环境工作
6. workon test1
7. https://img2018.cnblogs.com/blog/1137363/201906/1137363-20190604134240711-849693906.png
8. （3）退出虚拟环境
9. deactivate
10. 
11. 4、其他命令
12. （1）删除虚拟环境
13. rmvirtualenv test1
14. 
15. （2）列出所有虚拟环境
16. lsvirtualenv
17. 详情查看官网：<https://virtualenvwrapper.readthedocs.io/en/latest/command_ref.html>
18. 三、环境一致性
19. 如何保证在linux下的环境与windows下的环境一致，有时我们是在windows下开发的项目，后面需要部署到linux上，如何将windows环境中的包安装到linux上。
20. 1、将windows下的包导出
21. 进入windows下开发的虚拟环境，执行以下命令

**pip3 freeze > requirements.txt**

1. 
2. 2、上传文件
3. 执行以上步骤后会在当前目录下生成requirements.txt文件，然后将文件拖拽上传，可以在系统上安装拖拽上传的插件
4. yum install lrzsz -y
5. 
6. 3、执行requirements.txt文件

**pip3 install -r requirements.txt**

1. 
2. 此时等待完成安装即可。

安装mysql 数据库

sudo apt-get install mysql-server

sudo apt-get install mysql-client

sudo apt-get install libmysqlclient-dev

第一步完成后会要求你输入数据库root密码

或者 sudo mysql –uroot –p 直接登陆后修改密码

3. 开启、关闭MySQL服务：

/etc/init.d/mysql start

/etc/init.d/mysql stop

4.现在就可以免密码登录数据库了

sudo mysql -u root –p

ERROR 1698 (28000): Access denied for user 'root'@'localhost'

mysql  Ver 14.14 Distrib 5.7.25, for Linux (x86\_64)

问题描述：

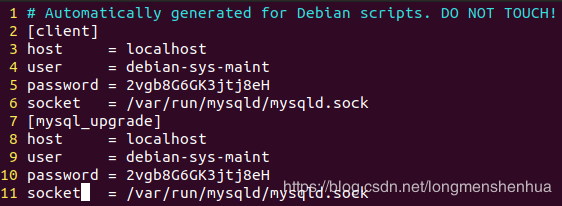
Ubuntu在安装完成mysql后，无法直接通过mysql -u root -p命令登录mysql，出现如下错误：ERROR 1698 (28000): Access denied for user 'root'@'localhost'

解决方法：

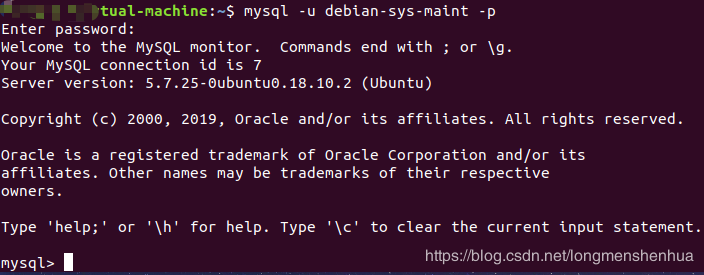
Ubuntu linux下安装新版本mysql，安装时root密码是随机的，因此会出现上述错误。

第一步，登录mysql

既然我们无法通过root用户进入MySQL，那我们就只能通过其他用户登录MySQL了。在我们下载安装MySQL的时候系统给我们创建了一个用户。打开 /etc/mysql/debian.cnf如下。

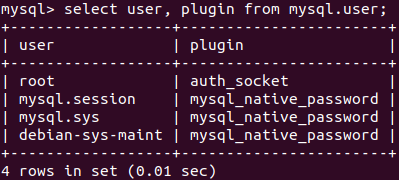


我们找到client的user和password，然后在终端输入 mysql -u debian-sys-maint -p 然后回车输入 password的内容。这样就进入了MySQL了，如图：



第二步，查看user表

在mysql中输入：select user, plugin from mysql.user; 结果如下：



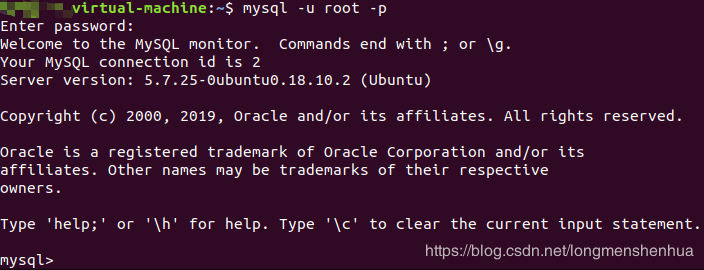
第三步，修改root密码

输入以下命令：update mysql.user set authentication\_string=PASSWORD('kongweixv'), plugin='mysql\_native\_password' where user='root';

第四步，刷新

输入以下命令：flush privileges;

然后重新启动服务器，就可以正常登录啦



python3 manage.py makemigrations

python3 manage.py migrate

pip3 install uwsgi 之后，运行uwsgi 报错：[uwsgi: command not found]

解决方案：建立软链接

sudo ln -s /usr/local/python3/bin/uwsgi /usr/bin/uwsgi

sudo apt install uwsgi-core

sudo apt install uwsgi-plugin-python

sudo apt install uwsgi-plugin-python3

Uwsgi.ini

测试uwsgi 拉起项目

sudo uwsgi --http :8000 --wsgi-file wsgi.py

[uwsgi]

#使用nginx连接时使用

socket=127.0.0.1:8080

#直接做web服务器使用 python manage.py runserver ip:port

#http=127.0.0.1:8080

#项目目录

chdir=/home/ubuntu/damei/

#项目中wsgi.py文件的目录，相对于项目目录

wsgi-file=damei/wsgi.py

#指定启动的工作进程数

processes=4

#指定工作进程中的线程数

threads=2

#有主进程

master=True

#保存启动之后主进程的pid

pidfile=uwsgi.pid

#设置uwsgi后台运行，uwsgi.log保存日志信息

daemonize=uwsgi.log

Uwsgi.ini

[uwsgi]

#使用nginx连接时使用

socket=127.0.0.1:8000

#直接做web服务器使用 python manage.py runserver ip:port

#http=127.0.0.1:8000

#项目目录

chdir=/home/ubuntu/damei/

#项目中wsgi.py文件的目录，相对于项目目录

module =damei.wsgi

#指定启动的工作进程数

processes=4

#指定工作进程中的线程数

threads=2

#有主进程

master=True

#保存启动之后主进程的pid

pidfile=uwsgi.pid

#设置uwsgi后台运行，uwsgi.log保存日志信息

daemonize=uwsgi.log

#指定python版本

pythonpath = /usr/local/lib64/python3.6/site-packages

pythonpath = /usr/local/lib/python3.6/site-packages

Nginx

安装nginx

sudo apt-get install nginx

Nginx 配置文件所在位置 /usr/loacl/nginx/niginx.conf

把下面文件拷贝进去/etc/nginx/conf.d

添加倒nginx 的启动文件夹里

sudo cp /home/ubuntu/damei/config/damei\_config.conf /etc/nginx/conf.d/

拉取所有的静态文件到同一个目录下

在django 的settings.py 文件中 ，添加下面一行的内容 ，把原来的STATICFILES\_DIRS注释掉，在生产环境中这个配置失效

STATIC\_ROOT = os.path.join(BASE\_DIR,”static/”)

运行命令

python3 manage.py collectstatic

重启运行nginx

sudo service nginx restart

#启动nginx服务

systemctl start nginx.service

#停止nginx服务

systemctl stop nginx.service

#重启nginx服务

systemctl restart nginx.service

#重新读取nginx配置(这个最常用, 不用停止nginx服务就能使修改的配置生效)

systemctl reload nginx.service

damei\_config.conf

# the upstream component nginx needs to connect to

upstream django{

# server unix://path/to/your/mysite/mysite.sock; # for a file socket

server 127.0.0.1:8000;

# for a web port socket (we’ll use this first)

}

# configuration of the server

server{

# the port your site will be served on

listen 80;

# the domain name it will server for

server\_name 49.235.183.58;

# substitute your mechine’s IP address or FQDN

charset utf-8;

# max upload size

client\_max\_body\_size 75M;

# django mdia

location /media {

alias /home/ubuntu/damei/static/media ; #指向django 的media目录

}

location /static {

alias /home/ubuntu/damei/static; #指向django的static目录

}

location ^~ /upload\_files/ {

root /home/ubuntu/damei/; #指向django上传文件的目录

}

location ~ .\*\.(gif|jpg|jpeg|png)$ {

expires 24h;

root /home/ubuntu/damei/upload\_files/; #指定图片存放路径

access\_log /home/ubuntu/damei/images.log; #日志存放路径

proxy\_store on;

proxy\_store\_access user:rw group:rw all:rw;

proxy\_temp\_path /home/ubuntu/damei/upload\_files/; #图片访问路径

proxy\_redirect off;

proxy\_set\_header Host 127.0.0.1;

client\_max\_body\_size 10m;

client\_body\_buffer\_size 1280k;

proxy\_connect\_timeout 900;

proxy\_send\_timeout 900;

proxy\_read\_timeout 900;

proxy\_buffer\_size 40k;

proxy\_buffers 40 320k;

proxy\_busy\_buffers\_size 640k;

proxy\_temp\_file\_write\_size 640k;

if ( !-e $request\_filename)

{

proxy\_pass <http://127.0.0.1>; #默认80端口

}

}

# finally, send all non-media requests to the Django server

location / {

root /home/ubuntu/damei;

uwsgi\_pass 127.0.0.1:8000;

include /etc/nginx/uwsgi\_params; # the uwsgi\_params file you installed

}

}

运行nginx

sudo service nginx restart

启动项目要在uwsgi.ini 所在的文件夹下使用

sudo uwsgi -i uwsgi.ini

# 解决思路：no python application found, check your startup logs for errors

**sudo uwsgi --reload uwsgi.pid**

uwsgi3 --http :9090 -w rest.py这种形式启动即可

使用uwsgi 配置 django的时候，遇到问题：

no python application found, check your startup logs for errors

表示uwsgi无法找到django项目，原因是根目录下的uwsgi.ini配置文件错误

原配置：

module=/var/www/project/project.wsgi #原配置为绝对路径

改为：

module=project.wsgi:application #新配置

停止项目：

uwsgi --stop uwsgi.pid

& 符号是后台启动

uwsgi -i /home/ubuntu/damei/config/uwsgi.ini &

uwsgi –i uwsgi.ini &

修改静态文件夹的所有权限

sudo chmod 777 /home

sudo chmod 777 /home/ubuntu

sudo chmod 777 /home/ubuntu/damei

sudo chmod 777 /home/ubuntu/damei/static

卸载：  
1、卸载python3.4  
sudo apt-get remove python3.4

2、卸载python3.4及其依赖  
sudo apt-get remove --auto-remove python3.4

3、清除python3.4  
sudo apt-get purge python3.4  
or  
sudo apt-get purge --auto-remove python3.4

安装：  
下载python  
wget <https://www.python.org/ftp/python/2.7.9/Python-2.7.9.tgz>  
wget -P /usr/lib <https://www.python.org/ftp/python/2.7.9/Python-2.7.9.tgz(>下载文件到指定目录)

解压、编译安装  
tar -zxvf Python-2.7.9.tgz  
cd Python-2.7.9  
./configure --prefix=/usr/local/python-2.7.9  
make  
make install

系统自带了python版本，我们需要为新安装的版本添加一个软链  
ln -s /usr/local/python-2.7.9/bin/python /usr/bin/python2.7.9