CMDB系统开发三

一、资产主机模块

1、资产主机模型设计

主机展示信息表

常规性的收集：

`主机模型`：IP地址、连接端口、操作系统（Linux，Windows，Mac）、机器类型（虚拟机，物理机）、主机名、CPU型号、CPU物理个数、内存信息、系统平台、系统平台位数、UUID、SN、探测状态、创建主机时间、更新主机时间

还要收集`硬盘信息，网卡信息`，注意一台主机,硬盘可能有多个，网卡也会有多个。所以要单独把硬盘和网卡各单独设计成一张表，然后使用多对一跟主机模型连接起来。

`硬盘模型`：硬盘名、硬盘大小、外键关联（server）

`网卡模型`：网卡名、ip地址、外键关联（server）

还需要一张探测表，把探测时候的一些信息记录下来，比如在探测的时候，要输入主机IP，端口，要把这些信息收集下来，并且跟主机表进行关联。

①创建模型

```
class Server(models.Model):
    hostname = models.CharField(max_length=32, verbose_name='主机名')
    cpu_info = models.CharField(max_length=32, verbose_name='CPU信息')
    cpu_count = models.IntegerField(verbose_name='CPU个数')
    mem_info = models.CharField(max_length=32, verbose_name='内存信息')
    os_system = models.CharField(max_length=32, verbose_name='系统平台')
    os_system_num = models.IntegerField(verbose_name='系统平台位数')
    uuid = models.CharField(max_length=64, verbose_name='uuid')
    sn = models.CharField(max_length=32, verbose_name='sn')
    scan_status_list = [
        (0, '连接异常'),
        (1, '连接正常')
    ]
    scan_status = models.IntegerField(choices=scan_status_list, verbose_name='探测状态')
    create_date = models.DateTimeField(auto_now_add=True, verbose_name='创建主机时间')
    update_date = models.DateTimeField(auto_now=True, verbose_name='更新主机时间')
    server_auto = models.OneToOneField('ServerAuto')

class Disk(models.Model):
    name = models.CharField(max_length=32, verbose_name='硬盘名字')
    size = models.CharField(max_length=32, verbose_name='硬盘大小')
    server = models.ForeignKey('Server')


class NetWork(models.Model):
    name = models.CharField(max_length=32, verbose_name='网卡名字')
    ip_address = models.CharField(max_length=32, verbose_name='网卡地址')
    server = models.ForeignKey('Server')
```

②迁移数据库

```
manage.py@syscmdb > makemigrations resources
"D:\Program Files\JetBrains\PyCharm 2018.3.4\bin\runnerw64.exe" "D:\Pr
Tracking file by folder pattern:  migrations
Migrations for 'resources':
  resources\migrations\0003_auto_20190219_0116.py
    - Create model Disk
    - Create model NetWork
    - Create model Server
    - Create model ServerAuto
    - Add field server_auto to server
    - Add field server to network
    - Add field server to disk

Following files were affected
 E:\syscmdb\resources\migrations\0003_auto_20190219_0116.py
Process finished with exit code 0
manage.py@syscmdb > migrate resources
```

2、资产主机展示列表

路由

```
13        url(r'^server/list/', ServerListView.as_view(), name='server_list'),
```

视图

```
141    class ServerListView(ListView):
142        template_name = 'server_list.html'
143        model = Server
```

模板

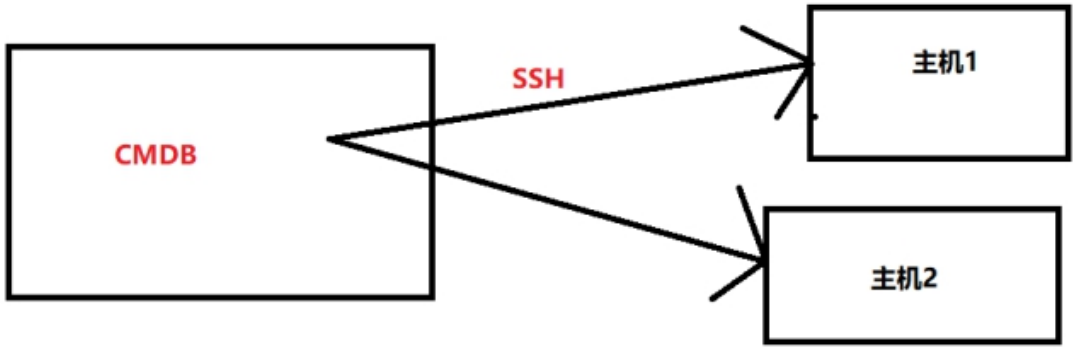
```
118                    {% for object in object_list %}
119                        <tr>
120                            <td class="text-center">{{ object.hostname }}</td>
121                            <td class="text-center">{{ object.cpu_info }}</td>
122                            <td class="text-center">{{ object.cpu_count }}</td>
123                            <td class="text-center">{{ object.mem_info }}</td>
124                            <td class="text-center">{{ object.os_system }}</td>
125                            <td class="text-center">{{ object.os_system_num }}</td>
126                            <td class="text-center">{{ object.scan_status }}</td>
127                            <td class="text-center">
128                                <a class="btn btn-xs btn-info" onclick="server_flush({{ object.id
    }})">刷新探测</a>
129                                <a class="btn btn-danger btn-sm" onclick="">删除</a>
```

没有数据，后期通过检查脚本推送服务器的信息到cmdb系统中，然后就添加到资产主机数据表中。

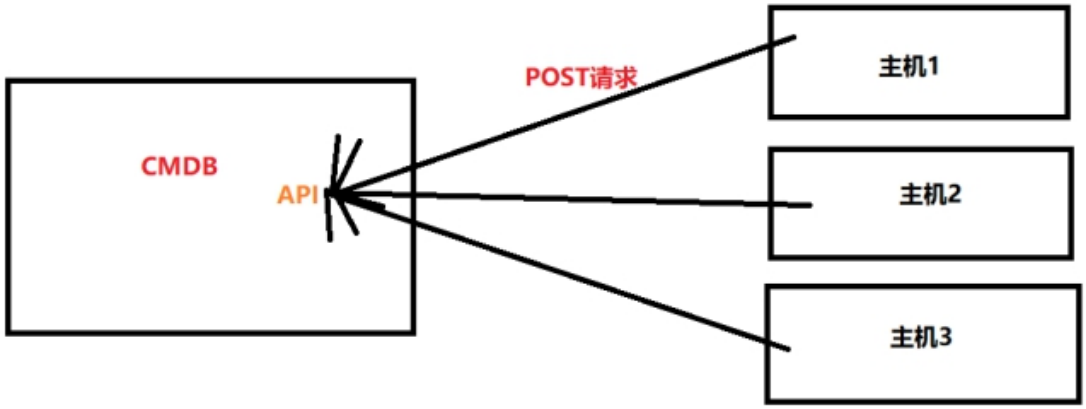
3、资产主机探测功能设计

常规的信息收集，有两种方式，一种是主动采集，一种是客户端收集。

主动探测：一般是通过，SNMP/SSH/TELNET等手段进行远程收集。



客户端采集 ：需要在每一个要部署的主机中部署一个客户端进行数据采集并且发送到远程服务端进行接收。



> Tip:升级安装Linux平台Python版本3以上
>
> 注意：本安装需要使用到CentOS base源 epel源

```
#!/bin/bash
yum install -y openssl-devel bzip2-devel expat-devel gdbm-devel readline-devel sqlite-devel libffi-devel
if [ ! -f './Python-3.7.3.tgz' ];then
    echo "本地没有Python源码包,需要远程下载,请耐心等待\n"
    wget https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tgz
fi
tar xvf Python-3.7.3.tgz
cd Python-3.7.3
./configure --prefix=/usr/local/python3 --with-ssl && make && make install
echo 'PATH=/usr/local/python3/bin:$PATH' >> /etc/profile
source /etc/profile
================================================================================
==========
#执行完成之后查看命令行
[root@localhost ~]# python3 --version
Python 3.7.3
[root@localhost ~]# pip3 --version
pip 19.0.3 from /usr/local/python3/lib/python3.7/site-packages/pip (python 3.7)
```

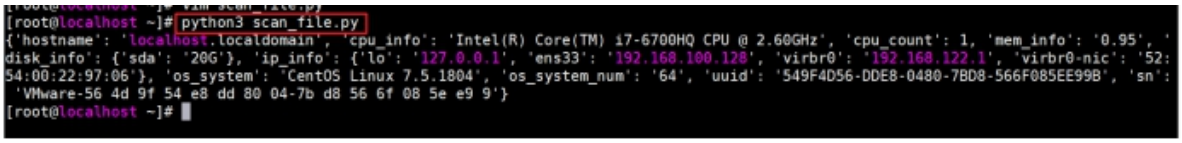
```
18  ================================================================================
    ==========
19  #配置pip国内源
20  shell > cd ~
21  shell > mkdir .pip
22  shell > cd .pip/
23  shell > vim pip.conf
24  #文件内容
25  [global]
26  index-url = http://pypi.douban.com/simple/
27  [install]
28  trusted-host = pypi.douban.com
```

4、数据收集API和客户端编写

①使用客户端收集到主机信息

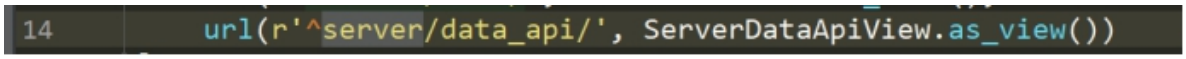
收集信息监测脚本需要安装的依赖包

```
1   shell > pip3 install psutil requests distro
```

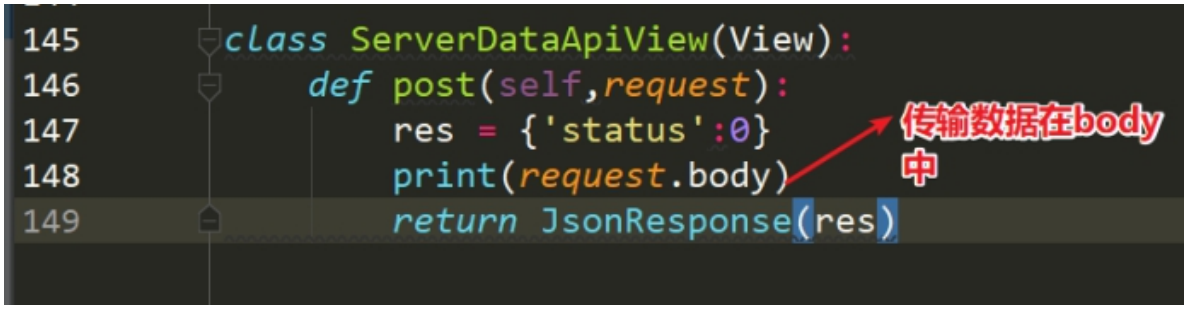


②cmdb编写一个接收数据的API

路由

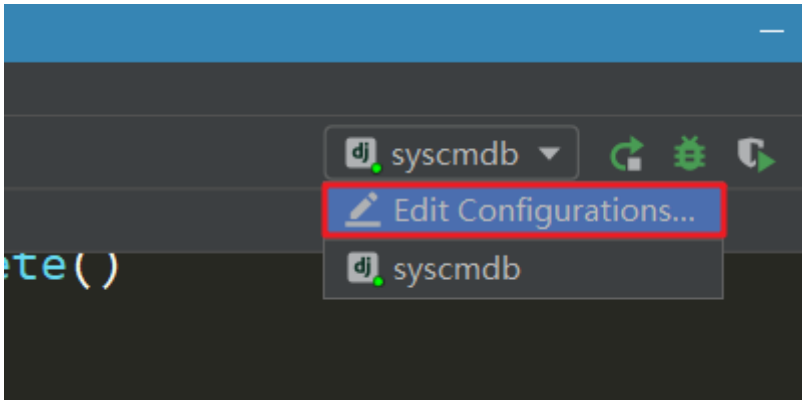


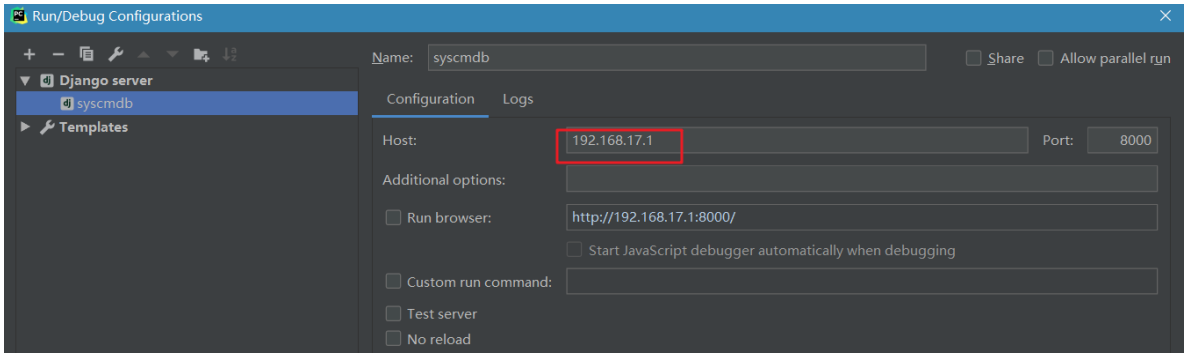
视图

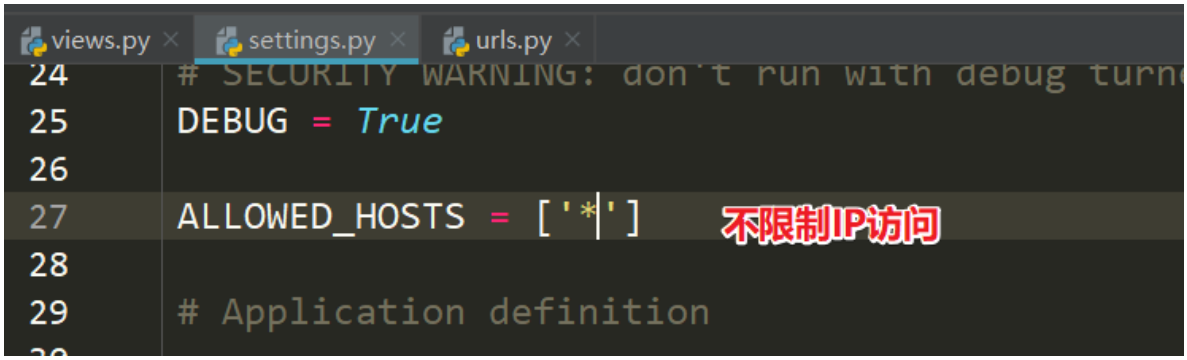


③修改请求API的地址 scan_file.py

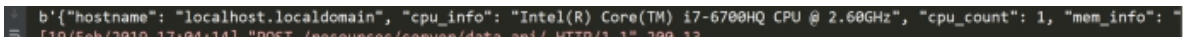
可以把cmdb系统运行在和Linux客户端在同一网段







④查看可以获取到数据



csrf_token验证解决

```
from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator
```

方法一:在视图类中定义dispatch方法，为dispatch方法加csrf_exempt装饰器

```
@method_decorator(csrf_exempt)
def dispatch(self, request, *args, **kwargs):
 return super(UserAuthView,self).dispatch(request,*args,**kwargs)
```

方法二：为视图类上方添加装饰器

```
@method_decorator(csrf_exempt,name='dispatch')
class UserAuthView(View):
```

方式三：在url.py中为类添加装饰器

```
url(r'^auth/', csrf_exempt(views.UserAuthView.as_view())),
```

5、添加资产主机

视图

```
132
133     class ServerDataApiView(View):
134         def post(self, request):
135             # print(request.body)
136             data = json.loads(request.body)
137             res = {'status': 0, 'msg': 'success'}
138             # 确认  主机是否已经被添加  uuid查询是否已经存在
139             try:
140                 server = Server.objects.get(uuid=data['uuid'])
141             except Exception:
142                 server = Server()
143             try:
144                 server.hostname = data['hostname']
145                 server.cpu_info = data['cpu_info']
146                 server.cpu_count = data['cpu_count']
147                 server.mem_info = data['mem_info']
148                 server.os_system = data['os_system']
149                 server.os_system_num = data['os_system_num']
150                 server.uuid = data['uuid']
151                 server.sn = data['sn']
152                 server.scan_status = 1
153                 server.save()
152                 server.scan_status = 1
153                 server.save()
154                 # 磁盘
155                 disk = Disk()
156                 for key, value in data['disk_info'].items():
157                     disk.name = key
158                     disk.size = value
159                     disk.server = server
160                     disk.save()
161                 # 网卡
162                 network = NetWork()
163                 for key, value in data['ip_info'].items():
164                     network.name = key
165                     network.ip_address = value
166                     network.server = server
167                     network.save()
168             except Exception as e:
169                 print(e)
170                 res = {'status': 1, 'msg': 'error'}
171             return JsonResponse(res)
172
```

在收集主机上执行

```
[root@localhost ~]# python3 scan_file.py
{"status": 0}
[root@localhost ~]#
```

显示效果

项目这里暂时开发到目前模块。

> **扩展其他功能：参考一些开源的cmdb产品 jumpserver**
>
> **①主动发送客户端检测脚本到被监控机** python sftp传输
>
> **②管理主机使用webssh** 可以实现一些开源的工具 webssh（后端建立长连接） xtemjs(前端显示命令行)
>
> https://github.com/huashengdun/webssh
>
> **③web端的ftp**
>
> **④接入监控的API** 获取数据 通过highchart echart 出监控图表
>
> **⑤CI CD** 实现从代码仓库获取代码发布到服务器
>
> http://www.walle-web.io/
>
> **开源cmdb参考：**
>
> Bigops java、jumpserver、蓝鲸 tencent等等

二、django项目上线服务器

1、基本环境要求

> python3
>
> django == 1.11.18
>
> mysql5.6

```
#shell > pip3 install django==1.11.18 pymysql faker paramiko
#根据文件内容安装模块及其对应版本
shell > pip3 install -r  requirement.txt
```

①打包项目并上传到服务器

②解压部署

```
shell > cd /usr/local/
shell > unzip syscmdb.zip
```

③数据导入

```
mysql > create database syscmdb;
mysql > use syscmdb;
mysql > source /usr/local/syscmdb/syscmdb.sql;
```

配置数据库连接

```
shell > cd  /usr/local/syscmdb/syscmdb
shell > vim settings.py
```

```
# }
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'syscmdb',
        'USER': 'root',
        'PASSWORD': '123456',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
```

④启动

```
shell > cd /usr/local/syscmdb
shell > python3 manage.py runserver 192.168.17.110:8000
```

如果遇到以下问题，模块的依赖没有安装。根据pip进行安装

```
dashboard  db.sqlite3  manage.py  __pycache__  resources  static  syscmdb  syscmdb.sql  templates  users
[root@server001 syscmdb]# python3 manage.py runserver
Traceback (most recent call last):
  File "manage.py", line 8, in <module>
    from django.core.management import execute_from_command_line
ModuleNotFoundError: No module named 'django'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "manage.py", line 14, in <module>
    import django
ModuleNotFoundError: No module named 'django'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "manage.py", line 17, in <module>
    "Couldn't import Django. Are you sure it's installed and "
ImportError: Couldn't import Django. Are you sure it's installed and available on your PYTHONPATH environment variable? Did you fo
get to activate a virtual environment?
[root@server001 syscmdb]#
```

没有安装django

```
dashboard  db.sqlite3  manage.py  __pycache__  resources  static  syscmdb  syscmdb.sql  templates  users
[root@server001 syscmdb]# python3 manage.py runserver
Traceback (most recent call last):
  File "manage.py", line 22, in <module>
    execute_from_command_line(sys.argv)
  File "/usr/local/python3/lib/python3.7/site-packages/django/core/management/__init__.py", line 364, in execute_
    utility.execute()
  File "/usr/local/python3/lib/python3.7/site-packages/django/core/management/__init__.py", line 308, in execute
    settings.INSTALLED_APPS
  File "/usr/local/python3/lib/python3.7/site-packages/django/conf/__init__.py", line 56, in __getattr__
    self._setup(name)
  File "/usr/local/python3/lib/python3.7/site-packages/django/conf/__init__.py", line 41, in _setup
    self._wrapped = Settings(settings_module)
  File "/usr/local/python3/lib/python3.7/site-packages/django/conf/__init__.py", line 110, in __init__
    mod = importlib.import_module(self.SETTINGS_MODULE)
  File "/usr/local/python3/lib/python3.7/importlib/__init__.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "<frozen importlib._bootstrap>", line 1006, in _gcd_import
  File "<frozen importlib._bootstrap>", line 983, in _find_and_load
  File "<frozen importlib._bootstrap>", line 953, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "<frozen importlib._bootstrap>", line 1006, in _gcd_import
  File "<frozen importlib._bootstrap>", line 983, in _find_and_load
  File "<frozen importlib._bootstrap>", line 967, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 677, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 728, in exec_module
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "/usr/local/syscmdb/syscmdb/__init__.py", line 1, in <module>
    import pymysql
ModuleNotFoundError: No module named 'pymysql'
[root@server001 syscmdb]#
```

pymysql安装解决

以上操作只能开发环境调试使用，不建议使用在生产模式下。

2、uwsgi启动django项目

python项目django框架上线推荐使用uwsgi启动方式。前端采用nginx做反向代理，实现动静分离

https://docs.djangoproject.com/en/2.2/howto/deployment/wsgi/uwsgi/

```
shell > pip3 install uwsgi
shell > cd /usr/local/syscmdb/syscmdb
shell > vim uwsgi.ini
```

uwsgi.ini配置文件的内容

```
[uwsgi]
# 配置服务器的监听ip和端口，让uWSGI作为nginx的支持服务器的话，设置socke就行；如果要让uWSGI作为单独的web-server，用http
socket = 127.0.0.1:8000
# 配置项目目录（此处设置为项目的根目录）
chdir = /usr/local/syscmdb
# 配置入口模块 (django的入口函数的模块，即setting同级目录下的wsgi.py)
wsgi-file = syscmdb/wsgi.py
# 开启master，将会多开一个管理进程，管理其他服务进程
master = True
# 服务器开启的进程数量
processes = 2
# 以守护进程方式提供服，输出信息将会打印到log中
daemonize = wsgi.log
# 服务器进程开启的线程数量
threads = 4
# 退出的时候清空环境变量
vacuum = true
# 进程pid
pidfile = uwsgi.pid
# 配uWSGI搜索静态文件目录（及django项目下我们存放static文件的目录，用uWSGI作为单独服务器时才需要设置，此时我们是用nginx处理静态文件）
# check-static = /usr/local/syscmdb/static
```

启动uwsgi

```
shell > uwsgi --ini uwsgi.ini
```

3、Nginx反向代理实现动静分离

```
server {
        listen 80;
        server_name cmdb.heimadevops.com;
        location / {
           #引入uwsgi的必要参数
           include uwsgi_params;
           #转发到uwsgi的ip和端口
           uwsgi_pass 127.0.0.1:8000;
        }
        #静态资源 Nginx直接返回
        location /static {
           # 资源路径指定
           alias /usr/local/nginx/html/syscmdb1/static/;
```

```
            expires 1d;
        }
}
```