SimpleMonitor1

一个基于django框架的轻量级后台监控系统

功能简介

本系统基于django框架开发，使用`pyvmomi`库实现了对vcentre中物理服务器和服务器上虚拟机基本状态的监控，使用`easysnmp`库实现了对网络打印机的基本状态监控。具体实现的功能如下：

- 对物理服务器的电源/CPU/内存/GPU状态的监控（GPU监控尚未解决数据接口问题）
- 对虚拟机的电源/IP地址/CPU/磁盘空间/所属主机的监控
- 对网络打印机的打印页数/墨粉状态的监控
- 前端和后端均可按需实时更新监控数据
- 邮件告警功能：当设备的某些指标超过设定的阈值时，系统将判定设备状态异常，并向指定邮箱地址发送邮件告知（不会重复发送），当设备恢复正常后也会再次发邮件告知（阈值可修改或增删）：
  - 物理服务器：电源关闭，CPU使用率超过80%，内存使用率超过90%
  - 虚拟机：CPU使用率超过80%
  - 网络打印机：墨粉量低于20%，打印机状态代码异常
- 后台管理系统：可在后台管理系统手动设置设备状态，管理用户等

环境配置

基础环境为`Python 3.10`，按如下方式安装所需第三方库：

```
pip install -r requirements.txt
```

也可根据`requirements.txt`文件内容手动安装

在`settings.py`文件中需要填写以下配置信息：

- `VSPHERE_CONFIG`：填入vcentre服务器地址，用户名，密码
- `PRINTER_CONFIG_1`：填入网络打印机IP，通讯接口类型（一般为`public`或`private`），后面所有打印机配置同理
- `EMAIL_BACKEND`：邮件smtp服务后端，此处默认使用django后端
- `EMAIL_HOST`：邮件供应商的smtp主机名
- `EMAIL_PORT`：邮件供应商指定的smtp端口
- `EMAIL_USE_TLS`：是否使用TLS协议
- `EMAIL_USE_SSL`：是否使用SSL协议
- `EMAIL_HOST_USER`：邮件账户
- `EMAIL_HOST_PASSWORD`：邮件账户的密码（可能是smtp专用密码）
- `DEFAULT_FROM_EMAIL`：发件人地址，一般与`EMAIL_HOST_USER`相同
- `ALERT_RECIPIENTS`：收件人地址，使用英文逗号隔开

然后迁移数据库结构：

```
python manage.py makemigrations
python manage.py migrate
```

创建管理员账号：

```
python manage.py createsuperuser
Username: admin
Email address: admin@example.com
Password: **********
Password (again): *********
Superuser created successfully.
```

启动开发服务器：

```
python manage.py runserver
```

此时访问`127.0.0.1:8000`，即可看到监控系统，访问`127.0.0.1:8000/admin`并以创建的管理员账号登录，即可进行后台数据管理。注意，开发服务器模式仅适用于开发和功能测试，在正式使用时应当按照下文所属方式进行部署。

部署说明

> **Important**
>
> 注意：在正式部署使用时，必须在`settings.py`文件中设置`DEBUG = False`

本系统已经集成了Daphne来托管django应用，推荐使用以下两种方式之一来启动服务：

1 直接使用Daphne运行服务

在项目根目录下运行

```
python manage.py runserver
```

此时服务监听本地端口，一般为`127.0.0.1:8000`或以终端输出结果为准，为了使得服务可以从外部访问，需要使服务监听所有外来请求：

```
python manage.py runserver 0.0.0.0: 8000
```

同时需要在`settings.py`中设置对IP地址的限制以保证安全：

```
ALLOWED_HOSTS = ["your_client_ip_1", "your_client_ip_2", ...]  # 或临时允许所有: ["*"]
```

之后从允许的外部设备访问`http://<服务器IP>:8000`即可

2 使用nginx配置反向代理

2.1 安装nginx

首先确保系统已经安装了nginx，并且防火墙开放了80和443端口

```
sudo apt update
sudo apt install nginx
sudo systemctl start nginx
```

然后在浏览器直接访问本机地址`localhost`或`127.0.0.1`，如果看到nginx欢迎界面则说明安装成功

2.2 编辑配置文件

编写nginx配置文件，其完整路径为/etc/nginx/sites-available/your_project（根据项目需要命名文件，编辑需要sudo权限）

```
sudo vim /etc/nginx/sites-available/your_project
```

以下是一个配置文件的示例：

```
# 基础配置
upstream django_asgi {
    server 127.0.0.1:8000;  # 指向Daphne运行端口
    keepalive 32;           # 保持长连接
}

server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;  # 替换为服务器的实际域名/IP

    # 静态文件处理（Django的collectstatic目录）
    location /static/ {
        alias /path/to/your/staticfiles/;  # 替换为实际路径
        expires 30d;
        access_log off;
    }

    # 媒体文件处理（可选）
    location /media/ {
        alias /path/to/your/media/;  # 替换为实际路径
        expires 7d;
    }

    # ASGI应用代理
    location / {
        # 基础代理设置
        proxy_pass http://django_asgi;
        proxy_http_version 1.1;

        # 连接升级头（WebSocket支持）
        proxy_set_header Upgrade $http_upgrade;
```

```
        proxy_set_header Connection "upgrade";

        # 标准代理头
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # 超时设置
        proxy_read_timeout 300;
        proxy_connect_timeout 300;
        proxy_send_timeout 300;

        # 禁用缓冲
        proxy_buffering off;
    }

    # 错误页面
    error_page 500 502 503 504 /custom_50x.html;
    location = /custom_50x.html {
        root /usr/share/nginx/html;
        internal;
    }
}
```

重点关注两处修改：

一是`server_name`改为服务器自身的IP地址，以供其他用户访问

二是静态文件的路径改为实际保存的路径，保存方法见2.4小节

2.3 启用配置

```
sudo ln -s /etc/nginx/sites-available/your_project /etc/nginx/sites-enabled/
sudo rm /etc/nginx/sites-enabled/default  # 移除默认配置
```

2.4 收集静态文件

静态文件指的是django项目中使用的css，img，js等格式的文件，如果没有这些文件，网站会很难看，所以需要收集这些文件供nginx加载使用。

首先在django项目的`settings.py`文件中设置保存的文件类型和地址：

```
STATIC_URL = 'static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

然后在django项目中运行命令：

```
python manage.py collectstatic
```

2.5 测试并重载nginx配置

```
sudo nginx -t        # 检查配置语法
sudo systemctl reload nginx  # 重载配置
```

2.6 重新启动服务

```
python manage.py runserver
```

此时在外部直接访问`http://<服务器IP>`（不加8000端口），如果能够看到正常的监控系统页面，则说明配置成功。如果出现静态文件加载失败的情况，可从以下几点进行排查：

（1）`settings.py`文件中必须配置`DEBUG = False`，并且`ALLOWED_HOSTS`中已经包含了本机IP地址（`127.0.0.1`和`localhost`都要有）和外部允许的IP地址；

（2）`settings.py`文件中`STATIC_ROOT`必须配置为绝对路径；

（3）nginx配置文件中静态文件路径alias的末尾必须加上`/`，且与`STATIC_ROOT`完全一致；

（4）确保nginx用户对于保存的静态文件具有足够的权限，假设静态文件存储在`/var/www/yourproject/static/`路径下，那么可作如下调整：

```
# 确保静态文件目录所有权为 Nginx 用户（通常是 www-data）
sudo chown -R www-data:www-data /var/www/yourproject/static/

# 设置正确的权限（目录755，文件644）
sudo find /var/www/yourproject/static -type d -exec chmod 755 {} \;
sudo find /var/www/yourproject/static -type f -exec chmod 644 {} \;

# 验证权限
ls -ld /var/www/yourproject/static/
```

同时，还要确保nginx对每个上级目录都有执行权限（至少为755）

```
# 从根目录开始检查（示例路径）
sudo namei -l /var/www/yourproject/static/

# 修复上层目录权限（至少需要755）
sudo chmod 755 /var /var/www /var/www/yourproject
```

> **Important**
>
> 以上每个方法修复后都应当重载nginx配置、重启django服务以验证效果。

未来开发计划

本系统功能仍较为简陋，未来计划开发如下功能：

- 增加对服务器网络流量的监测，包括上下行数据量，丢包/延迟情况等
- 实现对常用服务/网站的在线状态监控
- 实现对物理服务器磁盘状态监控

- 增加打印机的队列监控，尝试追踪打印者IP