loonflow 工单系统

该项目是基于django的工作流引擎，工单。项目托管在 **Github**

一、安装基础环境

1.1 安装python 和 pip

```
yum install -y epel-release
yum install -y https://centos7.iuscommunity.org/ius-release.rpm
yum install -y python36u python36u-devel python36u-pip

# 更新pip
pip3.6 install --upgrade pip
pip -V
```

1.2 安装和新建python3虚拟环境

```
# 安装
pip install -U virtualenv
rm -f /usr/bin/python3 && ln -s /usr/bin/python3.6 /usr/bin/python3

# 建立virtualenv目录
mkdir -p /data/virtualenv

# 新建python3虚拟环境（纯净安装，名称为loonflow）
cd /data/virtualenv
virtualenv --no-site-packages -p /usr/bin/python3.6 loonflow

# 激活进入虚拟环境
source /data/virtualenv/loonflow/bin/activate
(loonflow) [root@localhost virtualenv]#
```

1.3 安装mysql

```
# 下载yum源
wget -i -c http://dev.mysql.com/get/mysql57-community-release-el7-10.noarch.rpm

# 安装yum源
yum -y install mysql57-community-release-el7-10.noarch.rpm
```

```
# yum安装MySQL服务
yum -y install mysql-community-server MySQL-python mysql-de
vel

# 首先启动MySQL
systemctl start  mysqld.service

# 查看密码
此时MySQL已经开始正常运行，不过要想进入MySQL还得先找出此时root用户的
密码，通过如下命令可以在日志文件中找出密码：
grep "password" /var/log/mysqld.log

# 如下命令进入数据库：
mysql -uroot -p
# 输入初始密码，此时不能做任何事情，因为MySQL默认必须修改密码之后才能
操作数据库：
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'Admin@1
23';
```

1.4 创建数据库

```
# 创建库指定utf8编码
# 创建后端数据库
mysql> CREATE DATABASE loonflow DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci;

# 创建前端数据库
mysql> CREATE DATABASE shutongflow DEFAULT CHARACTER SET ut
f8 COLLATE utf8_general_ci;
```

1.5 安装redis

启动redis(用于生成唯一的工单流水号+celery异步任务[执行脚本、通知脚本])

```
# 安装
yum install -y redis

# 启动
systemctl start redis
systemctl enable redis
```

二、安装后端

2.1 下载loonflow代码

```
cd /opt
git clone https://github.com/blackholll/loonflow.git

# 切换到 r0.3.19 tag
cd loonflow/
git checkout r0.3.19
```

2.2 安装loonflow依赖

```
# 进入虚拟环境
source /data/virtualenv/loonflow/bin/activate

# 安装requirements依赖
```


阅读排行榜

```
cd /opt/loonflow/requirements

# 添加requests模块
echo "requests" >> common.txt

# 安装
(loonflow) # pip install -r pro.txt
```

2.3 配置loonflow

1）配置DB

```
# 文件：loonflow/settings/pro.py
……
DATABASES = {
    'default': {
            'ENGINE': 'django.db.backends.mysql',
            'NAME': 'loonflow', # 刚刚新建的数据库名称
            'USER': 'root',     # 数据库用户
            'PASSWORD': 'Admin@123',  # 数据库密码
            'HOST': '127.0.0.1',       # 数据库机器IP
            'PORT': '3306',            # 端口
        }
}
……
```

2）允许访问地址

最后设置以域名访问的时候不会报错。

```
# 文件：loonflow/settings/common.py
# ALLOWED_HOSTS 改成如下
……
ALLOWED_HOSTS = ['*']
……
```

3）指定配置文件

```
# 文件：loonflow/tasks.py
# DJANGO_SETTINGS_MODULE 对应配置文件改成为 pro
……
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'settings.pro')
……
```

4）初始化db

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

5）创建超级用户

```
python manage.py createsuperuser

账号：root,密码：Admin123
```

2.4 启动后端

```
python manage.py runserver 0.0.0.0:6060
```

启动之后，就可以访问admin后台，然后随便测试一个接口

```
# 后端url
```

```
http://172.16.34.250:6060

# 后端管理url
http://172.16.34.250:6060/admin

# api
http://172.16.34.250:6060/api/v1.0/workflows
```

三、安装前端

使用第三方代码：https://github.com/youshutong2080/shutongFlow
本系统使用Vue.js + Django开发，前端展示由loonflow配置决定，可以说前端是全动态，只需要配置好loonflow即可。

3.1 下载shutongFlow

```
cd /opt
git clone https://github.com/youshutong2080/shutongFlow.git
```

3.2 安装依赖

```
# 激活进入虚拟环境
source /data/virtualenv/loonflow/bin/activate

cd /opt/shutongFlow/apps
pip install -r requirements.txt
```

3.3 配置shutongFlow

1）初始化db

```
# 初始化db
python manage.py makemigrations
python manage.py migrate
```

2）导入数据

```
# 导入第三方数据（这里主要是用户数据）
cd /opt/shutongFlow/
mysql -uroot shutongflow < shutongflow.sql
```

```
# 导入loonflow数据（配置数据及用户数据）
mysql -uroot loonflow < loonflownew.sql
```

3）监听主机地址

```
# 文件：shutongFlow/fronted/config/index.js
# host: 'localhost' 改成如下
……
host: '0.0.0.0'
……
```

3.4 启动前端

1）启动shutongFlow后端

```
# 启动shutongFlow
cd /opt/shutongFlow/apps
```

```
python manage.py runserver 0.0.0.0:6062
```

2）启动shutongFlow前端vue

```
# 启动vue，会监听6061端口，前端的访问端口。
npm install .
npm run dev
```

3.5 启动通知服务

```
# 启动celery任务：
celery -A tasks worker -l info -Q loonflow
```

生产环境命令如下：celery multi start -A tasks worker -l info -c 8 -Q loonflow --logfile=xxx.log --pidfile=xxx.pid
参数说明：-c为启动的celery进程数， logfile为日志文件路径，pidfile为pid文件路径，可自行视情况调整。

四、前后端用户同步

前端数据库：shutongflow.user(username,alias,email)
后端数据库：loonflow.account_loonuser(username,alias,email)
当前端shutongflow库中的user表有插入动作，则把(username,alias,email)数据同步到后端loonflow库的account_loonuser表的(username,alias,email)数据。

4.1 插入的触发器

```
use shutongflow
DELIMITER //
CREATE TRIGGER user_trigger AFTER INSERT ON user FOR EACH R
OW
BEGIN
INSERT INTO loonflow.account_loonuser(username,alias,email)
 VALUES(new.username,new.alias,new.email);
END ;
//
DELIMITER ;
```

注：在命令提示符下输入delimiter // 这样是用//替换换行符，这样可避免点击换行键时执行程序。

4.2 查看数据

当前端集成好了LDAP后，前端采用LDAP账号登入，会同步账号信息到后端数据库，这时候可以查看数据是否已经同步。

```
select * from shutongflow.user;
select * from loonflow.account_loonuser;
```

4.3 触发器操作

```
# 查询触发器
select * from information_schema.triggers\G

# 删除触发器
use shutongflow
DROP TRIGGER user_trigger;
```

五、配置LDAP认证

当前端ldap登入后，会在user表中插入用户信息，然后触发器会同步数据到后端loonflow.account_loonuser用户表。

5.1 安装模块

```
# 激活进入虚拟环境
source /data/virtualenv/loonflow/bin/activate

pip install python-ldap django_auth_ldap
```

5.2 配置文件

修改前端同步ldap文件：shutongFlow/apps/apps/settings.py

```
import ldap
from django_auth_ldap.config import LDAPSearch, LDAPSearchU
nion, GroupOfNamesType

AUTHENTICATION_BACKENDS = (
    'django_auth_ldap.backend.LDAPBackend',  #配置为先使用LDA
P认证，如通过认证则不再使用后面的认证方式
    'django.contrib.auth.backends.ModelBackend',
)
AUTH_LDAP_SERVER_URI = "ldap://ldap.wmq.com:389"  # ldap服务
器地址
AUTH_LDAP_BIND_DN = 'cn=manager,dc=wmq,dc=com'  # 管理员账号
AUTH_LDAP_BIND_PASSWORD = 'xxxxx'
OUg = 'DC=wmq,DC=com'
AUTH_LDAP_USER_SEARCH = LDAPSearch(OUg, ldap.SCOPE_SUBTREE,
 "(&(objectClass=inetOrgPerson)(uid=%(user)s))")
AUTH_LDAP_USER_ATTR_MAP = {"first_name": "givenName", "last
_name": "sn", "alias": "sn", "email": "mail", "username":"n
ame"}
AUTH_LDAP_ALWAYS_UPDATE_USER = True  # 是否同步LDAP修改
AUTH_USER_MODEL = 'account.Users'
```

六、supervisor守护进程

supervisor 是基于 python 的任务管理工具，用来自动运行各种后台任务，当然你也能直接利用 nohup 命令使任务自动后台运行，但如果要重启任务，每次都自己手动 kill 掉任务进程，这样很繁琐，而且一旦程序错误导致进程退出的话，系统也无法自动重载任务。

6.1 安装supervisor

```
yum install -y epel-release
yum install -y supervisor

# 开机自启动
systemctl enable supervisord
```

6.2 配置supervisor

```
# 创建配置文件
mkdir /etc/supervisor
echo_supervisord_conf > /etc/supervisor/supervisord.conf
```

```
# 修改日志、pid、sock 存放目录
vim /etc/supervisor/supervisord.conf
……
[unix_http_server]
file=/var/run/supervisor.sock
```

```
……
[supervisord]
logfile=/var/log/supervisord.log
pidfile=/var/run/supervisord.pid
……
[supervisorctl]
serverurl=unix:///var/run/supervisor.sock
……
[include]
files = /etc/supervisor.d/*.ini
```

6.3 配置项目启动文件

```
# 创建项目配置文件
vim /etc/supervisord.d/loonflow.ini

[program:loonflow]
directory=/opt/loonflow
environment=HOME="/opt/loonflow"
command=/data/virtualenv/loonflow/bin/python manage.py runs
erver 0.0.0.0:6060      ;在python3的虚拟环境运行，需要指定virtua
lenv的绝对路径
autostart=true
autorestart=true
startsecs=3
stdout_logfile = /var/log/loonflow/loonflow_access.log
stderr_logfile = /var/log/loonflow/loonflow_error.log
stopasgroup=true
killasgroup=true

[program:shutongflow]
directory=/opt/shutongFlow/apps
command=/data/virtualenv/loonflow/bin/python manage.py runs
erver 0.0.0.0:6062
autostart=true
autorestart=true
startsecs=3
stdout_logfile = /var/log/loonflow/shutongflow_access.log
stderr_logfile = /var/log/loonflow/shutongflow_error.log
stopasgroup=true
killasgroup=true

[program:vue]
directory=/opt/shutongFlow/fronted
command=npm run dev
autostart=true
autorestart=true
startsecs=3
stdout_logfile = /var/log/loonflow/vue_access.log
stderr_logfile = /var/log/loonflow/vue_error.log
stopasgroup=true
killasgroup=true

[program:celery]
directory=/opt/loonflow
environment=HOME="/opt/loonflow"
command=/data/virtualenv/loonflow/bin/celery -A tasks worke
r -l info -c 8 -Q loonflow --pidfile=/var/run/loonflow_cele
ry.pid
autostart=true
autorestart=true
startsecs=3
stdout_logfile = /var/log/loonflow/celery.log
stderr_logfile = /var/log/loonflow/celery_error.log
stopasgroup=true
killasgroup=true
```

```
# 创建日志存储目录
mkdir /var/log/loonflow
```

参考：https://github.com/jimmy201602/workflowdemo/blob/master/supervisord.conf

参数说明：https://www.restran.net/2015/10/04/supervisord-tutorial/

6.4 启动

```
# 启动
systemctl start supervisord

systemctl status supervisord    //运行状态
systemctl stop supervisord       //停止
systemctl restart supervisord    //重启
```

启动完成后，可以查看项目对应进程的启动状态，如果有错误查看日志文件。

```
supervisorctl status

celery                           RUNNING   pid 31359, uptim
e 0:00:10
loonflow                         RUNNING   pid 31361, uptim
e 0:00:10
shutongflow                      RUNNING   pid 31362, uptim
e 0:00:10
vue                              RUNNING   pid 31360, uptim
e 0:00:10
```

6.5 访问

```
访问前端：http://172.31.57.1:6061
默认账号：admin,密码：yxuqtr
前端数据库：http://172.31.57.1:6062/admin/

访问后端（支持ldap）：http://172.31.57.1:6060
默认账号：admin，密码：123456
后端数据库：http://172.31.57.1:6060/admin/

去除"所有工单"功能按钮，找到//注释即可。
shutongFlow/fronted/src/router/routers.js
```

七、配置域名

vue前端需要采用nginx进行代理才能正常以域名地址访问；其他端口直接做了域名解析即可访问，不用设置代理。

7.1 安装nginx

```
yum install -y nginx
```

7.2 配置文件

```
vim /etc/nginx/conf.d/loonflow.conf

server {
  listen *:80;
```

```
  server_name     loonflow.wmq.com;

  location / {
    proxy_pass      http://127.0.0.1:6061;
  }
}
```

7.3 域名解析

```
172.31.57.1 loonflow.wmq.com
```

这样 6.5 节的访问url地址全部可以通过域名进行访问了。

八、附：企业微信通知脚本

send_wechat.py
内容如下：

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
"""
通知脚本loonflow会将标题、内容、参与人等信息作为全局变量传给通知脚本，
脚本中直接使用这些参数，通过各自的发送消息逻辑将信息发送出去。
详情参数参考：loonflow/media/notice_script/demo_notice_script.
py 里面有变量名称。
本脚本引用三个变量：participant：接收人，content_result：内容，tit
le_result：标题。
另外通知脚本内容是exec来执行的，不是直接执行脚本。所以 __name__ !==
"__main__"，不能用if __name__ == "__main__"。
"""

import json
import requests

class WeChat:
    def __init__(self):
        self.CORPID = 'wxc08*****7d5b2cb'
        self.CORPSECRET = 'pmv******************QBRv5w'
        self.AGENTID = '1000020'

    def _get_access_token(self):
        url = 'https://qyapi.weixin.qq.com/cgi-bin/gettoken
'
        values = {'corpid': self.CORPID,
                  'corpsecret': self.CORPSECRET,
                  }
        req = requests.post(url, params=values)
        return req

    def get_access_token(self):
        get_req = self._get_access_token()
        if get_req.status_code != 200:
            print('连接服务器失败')
        else:
            get_req_json = json.loads(get_req.text)
            if get_req_json['errcode'] != 0:
                print('响应结果不正确')
            else:
                access_token = get_req_json['access_token']
                return access_token

    def send_data(self):
        send_url = 'https://qyapi.weixin.qq.com/cgi-bin/mes
sage/send?access_token=' + self.get_access_token()
        send_values = {
            "touser": participant,
```

```
            "msgtype": "text",
            "agentid": self.AGENTID,
            "text": {
                "content": '%s <a href=\"http://172.31.57.1
:6061/ticket/todo\">%s</a>' % (content_result,title_result)
            },
            "safe": "0"
        }
        send_msges = (bytes(json.dumps(send_values), 'utf-8
'))
        respone = requests.post(send_url, send_msges)
        respone = respone.json()
        return respone["errmsg"]


wx = WeChat()
wx.send_data()
```

相关链接

后端（django）：https://github.com/blackholll/loonflow

本文采用的前端demo（vue+django）：https://github.com/youshutong2080/shutongFlow

其他前端demo（bootstrap+django）：https://hub.docker.com/r/webterminal/workflowdemo

通知脚本：https://github.com/blackholll/loonflow-helper/blob/master/notice_script_demo/notice_script_demo.py

django接入ldap：https://igolang.cn/Python/django%20%E6%8E%A5%E5%85%A5%20ldap/

分类: Deploy

分类: Deploy

wmht
关注 - 3
粉丝 - 29
+加关注

« 上一篇： Docker 安装 Request Tracker 工单系统
» 下一篇： Rancher 部署 loonflow 工单系统

posted @ 2019-10-14 17:05 wmht 阅读(3859) 评论(0) 编辑 收藏

顶部