Jira 工作流gitlab权限申请自动化

脚本：

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

import requests

import json

from flask import Flask

from flask import request

from flask\_restful import Api, Resource, reqparse, abort

import logging

from requests.auth import HTTPBasicAuth

from datetime import date, timedelta

server = Flask(\_\_name\_\_)

def log\_settings():

log\_path = "/opt/logs/git\_authorize.log"

logger = logging.getLogger(\_\_name\_\_)

logger.setLevel(level=logging.INFO)

handler = logging.FileHandler(log\_path)

handler.setLevel(logging.INFO)

formatter = logging.Formatter(

'%(asctime)s - %(name)s - %(levelname)s - %(message)s')

handler.setFormatter(formatter)

if not logger.handlers:

logger.addHandler(handler)

return logger

class authorize(Resource):

logger = log\_settings()

req\_headers = {'Content-Type': 'application/json', 'PRIVATE-TOKEN': 'PJBkwP6Q8sLdM1q8Ky5K'}

url = "https://gitlab.saicmobility.com/api/v4/"

jira\_api\_url = "http://jira.saicmobility.com:8080/rest/api/2"

jira\_auth = HTTPBasicAuth("autotester", "4UcN1iBvgKHDg")

def get\_user\_id(self, user\_name):

req\_params = {"username": user\_name}

req\_url = "%s/users" % self.url

response = requests.get(req\_url, params=req\_params, headers=self.req\_headers)

user\_id = json.loads(response.text)[0]["id"]

return user\_id

def get\_group\_id(self, group\_name):

req\_url = "%s/groups" % self.url

req\_params = {

'search': group\_name,

"per\_page": 9999

}

response = requests.get(req\_url, params=req\_params, headers=self.req\_headers)

res\_data = json.loads(response.text)

group\_id = res\_data[0]["id"]

return group\_id

def get\_project\_id(self, group\_id, project\_name):

req\_url = "%s/groups/%s/projects" % (self.url, group\_id)

req\_params = {

'search': project\_name,

'simple': 'true',

"per\_page": 100

}

response = requests.get(req\_url, params=req\_params, headers=self.req\_headers)

res\_data = json.loads(response.text)

project\_id = [item['id'] for item in res\_data if item['name'] == project\_name][0]

return project\_id

# 获取某个group下所有的project

def get\_all\_project\_id(self, group\_id):

project\_info\_list = []

for i in range(1, 10000):

req\_params = {

"per\_page": 100,

"page": i

}

req\_url = "%s/groups/%s/projects" % (self.url, group\_id)

response = requests.get(req\_url, params=req\_params, headers=self.req\_headers)

res\_data = json.loads(response.text)

project\_info\_list\_i = [(item['id'], item['name']) for item in res\_data]

if len(project\_info\_list\_i) > 0:

project\_info\_list = project\_info\_list + project\_info\_list\_i

else:

break

return project\_info\_list

def get\_group\_members(self, group\_id):

group\_member\_list = []

for i in range(1, 10000):

req\_params = {

"per\_page": 100,

"page": i

}

req\_url = "%s/groups/%s/members" % (self.url, group\_id)

response = requests.get(req\_url, params=req\_params, headers=self.req\_headers)

group\_user\_id\_list\_i = [item["id"] for item in json.loads(response.text)]

if len(group\_user\_id\_list\_i) > 0:

group\_member\_list = group\_member\_list + group\_user\_id\_list\_i

else:

break

return group\_member\_list

def get\_project\_members(self, project\_id):

project\_member\_list = []

for i in range(1, 10000):

req\_params = {

"per\_page": 100,

"page": i

}

req\_url = "%s/projects/%s/members" % (self.url, project\_id)

response = requests.get(req\_url, params=req\_params, headers=self.req\_headers)

project\_user\_id\_list\_i = [item["id"] for item in json.loads(response.text)]

if len(project\_user\_id\_list\_i) > 0:

project\_member\_list = project\_member\_list + project\_user\_id\_list\_i

else:

break

return project\_member\_list

def add\_user(self, action, user\_id, access\_level, object\_id, expires\_at=None):

if action == 'group':

req\_url = "%s/groups/%s/members" % (self.url, object\_id)

elif action == 'project':

req\_url = "%s/projects/%s/members" % (self.url, object\_id)

request\_data = {

"user\_id": user\_id,

"access\_level": access\_level,

"expires\_at": expires\_at

}

response = requests.post(req\_url, data=json.dumps(request\_data), headers=self.req\_headers)

print(response.text)

def edit\_access\_level(self, action, user\_id, access\_level, object\_id, expires\_at=None):

if action == 'group':

req\_url = "%s/groups/%s/members/%s" % (self.url, object\_id, user\_id)

elif action == 'project':

req\_url = "%s/projects/%s/members/%s" % (self.url, object\_id, user\_id)

params = {

'access\_level': access\_level,

'expires\_at': expires\_at

}

response = requests.put(req\_url, params=params, headers=self.req\_headers)

print(response.text)

# 在授权完成之后，操作jira的api来完成issue的流转

# 对issue进行评论，告知用户授权结果

def make\_comment(self, issue\_id, content):

comment\_url = "%s/issue/%s/comment" % (self.jira\_api\_url, issue\_id)

payload = json.dumps({

"body": content

})

headers = {

"Accept": "application/json",

"Content-Type": "application/json"

}

requests.request(

"POST",

comment\_url,

data=payload,

headers=headers,

auth=self.jira\_auth

)

# 对issue进行流转

def make\_transition(self, issue\_id, status):

if status == 'succ':

target\_code = '31'

elif status == 'fail':

target\_code = '41'

payload = json.dumps({

"transition": {

"id": target\_code

}

})

headers = {

"Accept": "application/json",

"Content-Type": "application/json"

}

transition\_url = "%s/issue/%s/transitions" % (self.jira\_api\_url, issue\_id)

res = requests.request(

"POST",

transition\_url,

data=payload,

headers=headers,

auth=self.jira\_auth

)

self.logger.info("流转结果：{}".format(res.text))

if res.status\_code == 204:

self.logger.info("issue：{} 状态流转成功".format(issue\_id))

else:

self.logger.info("issue：{} 状态流转失败".format(issue\_id))

def return\_msg(self, msg, msg\_code):

res = {"msg": msg, "msg\_code": msg\_code}

return json.dumps(res)

def send\_msg(self, applicant\_name, superior, issue\_id):

jira\_issue\_url = "http://jira.saicmobility.com:8080/browse/{}".format(issue\_id)

url = "http://qa.saicm.local/openapi/message/sendText"

req\_header = {

'secret': "VFuJ5muKYiKw9ufmtGfY"

}

message\_text = "申请类型：无需审批的自动授权，请确认该授权申请是否合理。\n申请人：{}\n申请人上级：{}\njira申请单据链接：{}"\

.format(applicant\_name, superior, jira\_issue\_url)

play\_load = {

"touser": ["heshan", "tuguichang", "sunzhikai", "wuji"],

"toparty": "",

"totag": "",

"content": message\_text

}

requests.post(url, data=play\_load, headers=req\_header)

cmd = authorize()

@server.route("/authorize", methods=["POST"])

def do\_authorize():

access\_level\_dict = {

'guest': 10,

'reporter': 20,

'developer': 30,

'master': 40,

'owner': 50

}

res\_data\_origin = request.data

res\_data = json.loads(res\_data\_origin.decode('utf-8'))

user\_name\_orgin\_data = res\_data['issue']['fields']['customfield\_16400']

if ',' in user\_name\_orgin\_data:

user\_name\_list = user\_name\_orgin\_data.split(',')

else:

user\_name\_list = user\_name\_orgin\_data.split(' ')

cmd.logger.info("获取到的用户列表信息为：{}".format(user\_name\_list))

group\_name\_list = res\_data['issue']['fields']['customfield\_20412']

group\_authorize = res\_data['issue']['fields']['customfield\_19915']['value']

access\_level\_name = res\_data['issue']['fields']['customfield\_19916']['value']

is\_urgent = res\_data['issue']['fields']['customfield\_20804']['value']

superior = res\_data['issue']['fields']['customfield\_21407']['name']

issue\_id = res\_data['issue']['key']

cmd.logger.info("issue: {}\n待授权的用户列表：{}\n授权的目标group：{}\n是否基于组进行授权：{}\n需要分配的权限级别：{}"

.format(issue\_id, user\_name\_list, group\_name\_list, group\_authorize, access\_level\_name))

tomorrow = (date.today() + timedelta(days=2)).strftime("%Y-%m-%d")

for user\_name in user\_name\_list:

if len(user\_name) == 0:

continue

try:

cmd.logger.info("当前读取的用户名是：{}".format(user\_name))

user\_id = cmd.get\_user\_id(user\_name)

except IndexError:

content = ("授权失败！用户: {} 不存在".format(user\_name))

cmd.make\_comment(issue\_id, content)

cmd.make\_transition(issue\_id, 'fail')

msg = cmd.return\_msg(content, 404)

return msg

for group\_name in group\_name\_list:

try:

group\_id = cmd.get\_group\_id(group\_name)

except IndexError:

content = ("授权失败！git group: {} 不存在".format(group\_name))

cmd.make\_comment(issue\_id, content)

cmd.make\_transition(issue\_id, 'fail')

msg = cmd.return\_msg(content, 404)

return msg

access\_level = access\_level\_dict[access\_level\_name]

# 如果只给组授权的情况：

if group\_authorize == 'true':

group\_user\_id\_list = cmd.get\_group\_members(group\_id)

if user\_id in group\_user\_id\_list:

cmd.logger.info("开始为用户：{} 添加gitlab组：{}的权限".format(user\_name, group\_name))

# 判断是否为紧急授权流程，如果是，则仅给用户进行有效期为48小时的授权，同时发送企业微信通知配管组。

if is\_urgent == '紧急授权流程':

cmd.edit\_access\_level('group', user\_id, access\_level, group\_id, expires\_at=tomorrow)

cmd.send\_msg(user\_name, superior, issue\_id)

else:

cmd.edit\_access\_level('group', user\_id, access\_level, group\_id, expires\_at='')

else:

if is\_urgent == '紧急授权流程':

cmd.add\_user('group', user\_id, access\_level, group\_id, expires\_at=tomorrow)

cmd.send\_msg(user\_name, superior, issue\_id)

else:

cmd.add\_user('group', user\_id, access\_level, group\_id, expires\_at='')

elif group\_authorize == 'false':

project\_name\_list = res\_data['issue']['fields']['customfield\_19914'].split(',')

if project\_name\_list is None:

content = "授权失败！缺少参数project\_name"

cmd.make\_comment(issue\_id, content)

cmd.make\_transition(issue\_id, 'fail')

msg = cmd.return\_msg(content, 404)

return msg

for project\_name in project\_name\_list:

cmd.logger.info("开始为用户：" + user\_name + "添加gitlab项目：" + project\_name + "的权限")

try:

project\_id = cmd.get\_project\_id(group\_id, project\_name)

except IndexError:

content = ("授权失败！git project: {} 不存在".format(project\_name))

cmd.make\_comment(issue\_id, content)

cmd.make\_transition(issue\_id, 'fail')

msg = cmd.return\_msg(content, 404)

return msg

project\_user\_id\_list = cmd.get\_project\_members(project\_id)

if user\_id in project\_user\_id\_list:

if is\_urgent == '紧急授权流程':

cmd.edit\_access\_level('project', user\_id, access\_level, project\_id, expires\_at=tomorrow)

cmd.send\_msg(user\_name, superior, issue\_id)

else:

cmd.edit\_access\_level('project', user\_id, access\_level, project\_id, expires\_at='')

else:

if is\_urgent == '紧急授权流程':

cmd.add\_user('project', user\_id, access\_level, project\_id, expires\_at=tomorrow)

cmd.send\_msg(user\_name, superior, issue\_id)

else:

cmd.add\_user('project', user\_id, access\_level, project\_id, expires\_at='')

content = "授权成功"

cmd.make\_comment(issue\_id, content)

cmd.make\_transition(issue\_id, 'succ')

msg = cmd.return\_msg(content, 200)

return msg

server.run(host="0.0.0.0", port=7777, debug=True)

jira工作流：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图示

描述已自动生成