# Python调用OpenStack api

## 1实战案例——使用Python对接API

### 1.1案例目标

（1）了解python-keystoneclient库。

（2）了解Python语法。

（3）使用Python编写代码，利用python-keystoneclient库查询keystone项目。

### 1.2案例分析

#### 1.规划节点

节点规划，见表7-1-1.

表7-1-1节点规划

|  |  |  |
| --- | --- | --- |
| **IP** | **主机名** | **节点** |
| 172.30.18.29 | controller | OpenStack控制节点 |

#### 2.基础准备

使用all-in-one云主机进行实验，安装python3环境，并导入相应的依赖包。

### 1.3案例实施

#### 1.基础环境安装

使用提供的python-3.6.8.tar.gz，解压并配置yum源，安装python3环境。安装完之后查看python版本，命令如下：

[root@openstack ~]# python3 --version

Python 3.6.8

安装完python3环境后，使用提供的whl文件安装依赖，命令如下：

# pip3 install certifi-2019.11.28-py2.py3-none-any.whl

# pip3 install urllib3-1.25.11-py3-none-any.whl

# pip3 install idna-2.8-py2.py3-none-any.whl

# pip3 install chardet-3.0.4-py2.py3-none-any.whl

# pip3 install requests-2.24.0-py2.py3-none-any.whl

安装之后可以使用命令查看是否被成功安装，命令如下：

[root@openstack ~]# pip3 list

DEPRECATION: The default format will switch to columns in the future. You can use --format=(legacy|columns) (or define a format=(legacy|columns) in your pip.conf under the [list] section) to disable this warning.

certifi (2019.11.28)

chardet (3.0.4)

idna (2.8)

pip (9.0.3)

requests (2.24.0)

setuptools (39.2.0)

urllib3 (1.25.11)

至此，基本环境安装完毕。

#### 2.编写python代码创建flavor

编写python代码对接OpenStack API，完成flavor的创建。在all-in-one节点的/root目录下创建create\_flavor.py文件，在该文件中编写python代码对接openstack api，要求在openstack私有云平台上创建一个云主机类型，名字为test、vcpu为1个、内存为1024m、硬盘为20G、ID为199999。执行完代码要求输出“云主机类型创建成功”。create\_flavor.py的文件内容如下：

import requests,json,time

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*全局变量IP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#执行代码前，请修改controller\_ip的IP地址，与指定router，IP可以input，也可以写成静态

controller\_ip = input("请输入访问openstack平台控制节点IP地址：（xx.xx.xx.xx)\n")

try:

url = f"http://{controller\_ip}:5000/v3/auth/tokens"

body = {"auth": {"identity": {"methods": ["password"], "password": {

"user": {"domain": {"name": "demo"}, "name": "admin", "password": "000000"}}},

"scope": {"project": {"domain": {"name": "demo"}, "name": "admin"}}}}

headers = {

"Content-Type": "application/json",

}

Token = requests.post(url, data=json.dumps(body), headers=headers).headers['X-Subject-Token']

headers = {

"X-Auth-Token": Token

}

except Exception as e:

print(f"获取Token值失败，请检查访问云主机控制节点IP是否正确？输出错误信息如下：{str(e)}")

exit(0)

class flavor\_api:

def \_\_init\_\_(self,handers:dict,resUrl:str):

self.headers=handers

self.resUrl=resUrl

#创建flavor类型

def create\_flavor(self,flavor\_name:str,ram,vcpus,disk,id):

self.headers['Content-Type']="application/json"

body={

"flavor":{

"name":flavor\_name,

"ram":ram,

"vcpus":vcpus,

"disk":disk,

"id":id,

}

}

status\_code = requests.post(self.resUrl, data=json.dumps(body), headers=self.headers).text

#获取flavor\_id

def get\_flavor\_id(self,flavor\_name:str):

result = json.loads(requests.get(self.resUrl,headers=self.headers).text)

for item in result['flavors']:

if(item['name']==flavor\_name):

return item['id']

flavor\_api=flavor\_api(headers,f"http://{controller\_ip}:8774/v2.1/flavors")

flavor\_api.create\_flavor(flavor\_name="test",ram=1024,vcpus=1,disk=20,id=199999)

flavor\_id = flavor\_api.get\_flavor\_id(flavor\_name="test")

print("云主机类型创建成功，flavor\_id为:",flavor\_id)

'''

#注意调用方法应该放在最前面！！！

'''

编写完python代码后，使用如下命令执行：

[root@openstack ~]# python3 create\_flavor.py

请输入访问openstack平台控制节点IP地址：（xx.xx.xx.xx)

172.30.18.29

云主机类型创建成功，flavor\_id为: 199999

正确执行，返回flavor的id为199999，使用命令查看flavor是否被成功创建，命令如下：

[root@openstack ~]# source /etc/keystone/admin-openrc.sh

[root@openstack ~]# openstack flavor list

+--------+------+------+------+-----------+-------+-----------+

| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |

+--------+------+------+------+-----------+-------+-----------+

| 199999 | test | 1024 | 20 | 0 | 1 | True |

+--------+------+------+------+-----------+-------+-----------+

确认创建flavor成功。

#### 3.编写python代码创建镜像

编写python代码对接OpenStack API，完成镜像的上传。在all-in-one节点的/root目录下创建create\_image.py文件，在该文件中编写python代码对接openstack api，要求在openstack私有云平台中上传镜像cirros-0.3.4-x86\_64-disk.img，名字为cirros001，disk\_format

为qcow2，container\_format为bare。执行完代码要求输出“镜像创建成功，id为:xxxxxx”。create\_image.py的文件内容如下：

import requests,json,time

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*全局变量IP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#执行代码前，请修改controller\_ip的IP地址，与指定router，IP可以input，也可以写成静态

controller\_ip = input("请输入访问openstack平台控制节点IP地址：（xx.xx.xx.xx)\n")

try:

url = f"http://{controller\_ip}:5000/v3/auth/tokens"

body = {"auth": {"identity": {"methods": ["password"], "password": {

"user": {"domain": {"name": "demo"}, "name": "admin", "password": "000000"}}},

"scope": {"project": {"domain": {"name": "demo"}, "name": "admin"}}}}

headers = {

"Content-Type": "application/json",

}

Token = requests.post(url, data=json.dumps(body), headers=headers).headers['X-Subject-Token']

headers = {

"X-Auth-Token": Token

}

except Exception as e:

print(f"获取Token值失败，请检查访问云主机控制节点IP是否正确？输出错误信息如下：{str(e)}")

exit(0)

class glance\_api:

def \_\_init\_\_(self, headers: dict, resUrl: str):

self.headers = headers

self.resUrl = resUrl

#创建glance镜像

def create\_glance(self, image\_name: str, container\_format="bare", disk\_format="qcow2"):

body = {

"container\_format": container\_format,

"disk\_format": disk\_format,

"name": image\_name,

}

status\_code = requests.post(self.resUrl, data=json.dumps(body), headers=self.headers).status\_code

#获取glance镜像id

def get\_glance\_id(self,image\_name:str):

result = json.loads(requests.get(self.resUrl,headers=self.headers).text)

for item in result['images']:

if(item['name']==image\_name):

return item['id']

#上传glance镜像

def update\_glance(self,image\_name:str,file\_path=""):

self.resUrl=self.resUrl+"/"+self.get\_glance\_id(image\_name)+"/file"

self.headers['Content-Type'] = "application/octet-stream"

seatus\_code = requests.put(self.resUrl,data=open(file\_path,'rb').read(),headers=self.headers).status\_code

glance\_api = glance\_api(headers,f"http://{controller\_ip}:9292/v2/images")

glance\_api.create\_glance(image\_name="cirros001") #调用glance-api中创建镜像方法

print("镜像创建成功，id为: ",glance\_api.get\_glance\_id(image\_name="cirros001"))

glance\_api.update\_glance(image\_name="cirros001",file\_path="./cirros-0.3.4-x86\_64-disk.img")

执行代码，命令如下：

# python3 create\_image.py

请输入访问openstack平台控制节点IP地址：（xx.xx.xx.xx)

172.30.18.29

镜像创建成功，id为: 714882f6-b6d4-4eb3-b655-8b63207755f0

可以看见执行代码成功，使用命令查看镜像是否上传成功，命令如下：

[root@allinone ~]# openstack image list

+--------------------------------------+-----------+--------+

| ID | Name | Status |

+--------------------------------------+-----------+--------+

| 714882f6-b6d4-4eb3-b655-8b63207755f0 | cirros001 | active |

+--------------------------------------+-----------+--------+

镜像是active状态，代码执行成功。

#### 4.编写python代码创建用户

编写python代码对接OpenStack API，完成用户的上传。在all-in-one节点的/root目录下创建create\_user.py文件，在该文件中编写python代码对接openstack api，要求在openstack私有云平台中创建用户chinaskill。具体代码如下：

import requests,json,time,os

#encoding=UTF-8

controller\_ip = '172.30.18.29'

url = f"http://{controller\_ip}:5000/v3/auth/tokens"

body = {"auth": {"identity": {"methods": ["password"], "password": {

"user": {"domain": {"name": "demo"}, "name": "admin", "password": "000000"}}},

"scope": {"project": {"domain": {"name": "demo"}, "name": "admin"}}}}

headers = {

"Content-Type": "application/json",

}

Token = requests.post(url, data=json.dumps(body), headers=headers).headers['X-Subject-Token']

headers = {

"X-Auth-Token": Token

}

class openstack\_role\_api:

def \_\_init\_\_(self, handers: dict, resUrl: str):

self.headers = handers

self.resUrl = resUrl

def create\_roles(self, role\_name):

body = {

"role": {

"description": "My new role",

"name": role\_name

}

}

status\_code = requests.post(self.resUrl, data=json.dumps(body), headers=self.headers)

result = json.loads(requests.get(self.resUrl, headers=self.headers).text)

role\_name = role\_name

for i in result['roles']:

if i['name'] == role\_name:

return f"角色 {role\_name} 创建成功,ID为{i['id']}"

def get\_role\_id(self, role\_name):

result = json.loads(requests.get(self.resUrl, headers=self.headers).text)

role\_name = role\_name

for i in result['roles']:

if i['name'] == role\_name:

return (f"角色 {role\_name} 的ID为{i['id']}")

def delete\_role(self, role\_name):

result = json.loads(requests.get(self.resUrl, headers=self.headers).text)

for i in result['roles']:

if i['name'] == role\_name:

i = i['id']

status\_code = requests.delete(f'http://{controller\_ip}:5000/v3/roles/{i}', headers=self.headers)

return f"角色 {role\_name} 已删除！"

def list\_roles(self):

result = json.loads(requests.get(self.resUrl, headers=self.headers).text)

roles = []

for i in result['roles']:

if i['name'] not in roles:

roles.append(i['name'])

return "该平台的角色为：\n"+'\n'.join(roles)

class openstack\_user\_api:

def \_\_init\_\_(self, handers: dict, resUrl: str):

self.headers = handers

self.resUrl = resUrl

def create\_users(self, user\_name):

body = {

"user": {

"description": "API create user!",

"domain\_id": "fa8baeb025724e1183b0416056378cc3",

"name": user\_name

}

}

status\_code = requests.post(self.resUrl, data=json.dumps(body), headers=self.headers).text

result = json.loads(requests.get(self.resUrl, headers=self.headers).text)

user\_name = user\_name

for i in result['users']:

if i['name'] == user\_name:

return f"用户 {user\_name} 创建成功,ID为{i['id']}"

def list\_users(self):

result = json.loads(requests.get(self.resUrl, headers=self.headers).text)

roles = []

for i in result['users']:

if i['name'] not in roles:

roles.append(i['name'])

return "该平台的用户为：\n"+'\n'.join(roles)

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

result = json.loads(requests.get(self.resUrl, headers=self.headers).text)

user\_name = user\_name

for i in result['users']:

if i['name'] == user\_name:

return (f"用户 {user\_name} 的ID为{i['id']}")

def delete\_user(self, user\_name):

result = json.loads(requests.get(self.resUrl, headers=self.headers).text)

for i in result['users']:

if i['name'] == user\_name:

i = i['id']

status\_code = requests.delete(f'http://{controller\_ip}:5000/v3/users/{i}', headers=self.headers)

return f"用户 {user\_name} 已删除！"

openstack\_role\_api = openstack\_role\_api(headers, f"http://{controller\_ip}:5000/v3/roles")

openstack\_user\_api = openstack\_user\_api(headers, f"http://{controller\_ip}:5000/v3/users")

# print(openstack\_role\_api.create\_roles("xxx"))

# print(openstack\_role\_api.list\_roles())

# print(openstack\_user\_api.create\_users("xxx"))

# print(openstack\_user\_api.list\_users())

# print(openstack\_user\_api.delete\_user("xxx"))

上面的详细代码定义了两个类，一个user类，一个role类，每个类下面有查询，创建，列出，删除四个函数，实例化一个类然后调用函数，例如创建用户chinaskill，就在代码的最下方调用create函数，具体如下：

print(openstack\_user\_api.create\_users("chinaskill"))

执行代码，效果如下：

[root@allinone ~]# python3 create\_user.py

用户 chinaskill 创建成功,ID为76372e3b4ef749b88871a65aa19824fc

使用命令查看用户列表，确定是否创建成功，命令如下：

[root@allinone ~]# openstack user list

+----------------------------------+-------------------+

| ID | Name |

+----------------------------------+-------------------+

| 0f8782af6a654d77b587e25a32f91f28 | cinder |

| 1ab30f77400448eba6b2d47e55084540 | demo |

| 2550fa93b1fe4cb582f1f46353b836d8 | ceilometer |

| 2d2a345336184b1ebbdf022f710084e8 | neutron |

| 48b816f9db9541b4bd9ca49ad453574c | glance |

| 76372e3b4ef749b88871a65aa19824fc | chinaskill |

| 765a16c99d7d42a4b69ff941f7791b54 | aodh |

| 788efa329f324b91a431ad56cd7b9a14 | nova |

| 7ecae98d16d54483b964c9c2548fd7bc | swift |

| 962612a3e7784df38d0c98fea1f30320 | heat |

| 9ee4731c00c24f659b8790be6b77bc8a | admin |

| d6fdd1e5e1a348e0b6c5b8c7f33ba5fa | placement |

| d957a578fed2452ab91bc651f2f1fb97 | heat\_domain\_admin |

| e91070fa751e49689963b566db999bee | gnocchi |

+----------------------------------+-------------------+

可以查看到chinaskill用户，python脚本执行成功。如果要调用其他功能，可按照上面的方法自行实验。

注：代码中的"domain\_id": "fa8baeb025724e1183b0416056378cc3",需要改成实际执行环境中的domain\_id。