Computer Lab 2, Part I

This notebook consists of instructions, exercises and questions that form the practical part of Lab II, Part I. In this assignment, you will learn the basics of the OpenStack Python APIs that can be used to interact directly with the IaaS services Keystone (Identity), Glance (Image) and Nova (compute). Please prepare your solution and answers to questions directly in this notebook, and export it to PDF. Upload that PDF as to the student portal to complete Part I of the Lab.

Task - 1

In [9]:
```
#Raheel Ali
from os import environ as env
import keystoneclient.v3.client as ksclient
```

To establish a client connection, we will need to pass a dictionary with information about the tenant, user, credentials and the API Identity endpoint. Here, I have sourced the "openrc.sh file" obtained from the Horizon dashboard in the underlying shell prior to starting the notebook. Hence, in order to actually run the code below, you would need to do the same with your own credentials.

In [10]:
```
keystone = ksclient.Client(auth_url=env['OS_AUTH_URL'],
                           username=env['OS_USERNAME'],
                           password=env['OS_PASSWORD'],
                           project_name=env['OS_PROJECT_NAME'],
                           project_domain_name=env['OS_USER_DOMAIN_NAME'],
                           project_id=env['OS_PROJECT_ID'],
                           version=env['OS_IDENTITY_API_VERSION'],
                           user_domain_name=env['OS_USER_DOMAIN_NAME'],
                           region_name=env['OS_REGION_NAME'])
```

Next command will collect all the available endpoints in keystone.

In [11]:
```
endpoints = keystone.service_catalog.get_endpoints()
```

Next section will print the services with associated infromation.

In [12]:
```
for endpoint in endpoints:
    for edp in endpoints[endpoint]:
        if edp['interface'] == 'public':
            print ('service: ', endpoint, ', region: ', edp['region'], ', public end
```

```
service:  compute , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:8774/v2.1
service:  identity , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:5000
service:  cloudformation , region:  east-1 , public endpoint:  https://east-1.cloud.
snic.se:8000/v1
service:  orchestration , region:  east-1 , public endpoint:  https://east-1.cloud.s
nic.se:8004/v1/fc1aade83c2e49baa7498b3918560d9f
service:  network , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:9696
service:  volumev3 , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:8776/v3/fc1aade83c2e49baa7498b3918560d9f
service:  image , region:  east-1 , public endpoint:  https://east-1.cloud.snic.se:9
292
service:  metric , region:  east-1 , public endpoint:  https://130.238.28.5:8041
service:  placement , region:  east-1 , public endpoint:  https://east-1.cloud.snic.
se:8780
service:  volumev2 , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:8776/v2/fc1aade83c2e49baa7498b3918560d9f
```

Questions:

1 - Explain the working of the code? 2 - Why we need openrc file to run this code? 3 - The code shows information about the "public" endpoints. Modify the code to show both the "public" and "internal" endpoints? 4 - What is the difference between "internal" and "public" endpoints?

In [13]:
```
#Task 1.3
for endpoint in endpoints:
    for edp in endpoints[endpoint]:
        if edp['interface'] == 'public':
            print ('service: ', endpoint, ', region: ', edp['region'], ', public end
        elif edp['interface'] == 'internal':
            print ('service: ', endpoint, ', region: ', edp['region'], ', internal e
```

```
service:  compute , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:8774/v2.1
service:  compute , region:  east-1 , internal endpoint:  http://172.29.236.9:8774/v
2.1
service:  identity , region:  east-1 , internal endpoint:  http://172.29.236.9:5000
service:  identity , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:5000
service:  cloudformation , region:  east-1 , internal endpoint:  http://172.29.236.
9:8000/v1
service:  cloudformation , region:  east-1 , public endpoint:  https://east-1.cloud.
snic.se:8000/v1
service:  orchestration , region:  east-1 , internal endpoint:  http://172.29.236.9:
8004/v1/fc1aade83c2e49baa7498b3918560d9f
service:  orchestration , region:  east-1 , public endpoint:  https://east-1.cloud.s
nic.se:8004/v1/fc1aade83c2e49baa7498b3918560d9f
service:  network , region:  east-1 , internal endpoint:  http://172.29.236.9:9696
service:  network , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:9696
service:  volumev3 , region:  east-1 , internal endpoint:  http://172.29.236.9:8776/
v3/fc1aade83c2e49baa7498b3918560d9f
service:  volumev3 , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:8776/v3/fc1aade83c2e49baa7498b3918560d9f
service:  image , region:  east-1 , internal endpoint:  http://172.29.236.9:9292
service:  image , region:  east-1 , public endpoint:  https://east-1.cloud.snic.se:9
292
service:  metric , region:  east-1 , public endpoint:  https://130.238.28.5:8041
service:  metric , region:  east-1 , internal endpoint:  http://172.29.236.9:8041
service:  placement , region:  east-1 , internal endpoint:  http://172.29.236.9:8780
service:  placement , region:  east-1 , public endpoint:  https://east-1.cloud.snic.
se:8780
service:  volumev2 , region:  east-1 , public endpoint:  https://east-1.cloud.snic.s
e:8776/v2/fc1aade83c2e49baa7498b3918560d9f
service:  volumev2 , region:  east-1 , internal endpoint:  http://172.29.236.9:8776/
v2/fc1aade83c2e49baa7498b3918560d9f
```

Answer:

Task 1.1) This program prints all the public end points, it prints service, endpoint, region and url.

Task 1.2) Openrc file is a configuration file which we need here to set the environment variables t run this code.

Task 1.4) The internal endpoints can be accessed inside a cloud environment and can't be accessible outside of the deployment network while public endpoints can be accessed publically outside the cloud.

Task - 2

In this task you need to write a small python program using Keystone and Nova APIs to list all

the available VMs in the project.

Use the following links and the code available in Task-1.

https://docs.openstack.org/python-novaclient/pike/ https://docs.openstack.org/python-novaclient/pike/reference/api/index.html

Following are the functions required to accomplish the task:

Load the required plugin:
loader = loading.get_plugin_loader(...)

Create the auth object:
auth = loader.load_from_options(....)

Create session object using "auth":
sess = session.Session(....)

Create Nova Client Object:
nova = client.Client(...)

Print the Vms:
nova.servers.list():

In [16]:
```
from keystoneauth1 import loading
from keystoneauth1 import session
from novaclient import client
loader = loading.get_plugin_loader('password')
auth = loader.load_from_options(auth_url=env['OS_AUTH_URL'],
                                username=env['OS_USERNAME'],
                                password=env['OS_PASSWORD'],
                                project_name=env['OS_PROJECT_NAME'],
                                project_domain_name=env['OS_USER_DOMAIN_NAME'],
                                project_id=env['OS_PROJECT_ID'],
                                user_domain_name=env['OS_USER_DOMAIN_NAME'])
sess = session.Session(auth=auth)
nova = client.Client(2.3, session=sess)
nova.servers.list()
```

Out[16]:
```
[<Server: docker_cloud>,
 <Server: Julie-Stack-C2-my_instance1-3vfb7agisyq6>,
 <Server: Julie-Stack-C2-my_instance0-fuzcedhhg67y>,
 <Server: vm_py_script_kev_cloud>,
 <Server: vm1>,
 <Server: s3>,
 <Server: wezh_lab2_scripted_instance>,
 <Server: sai_l2_task2>,
 <Server: ah_docker>,
 <Server: jovi3089_C2>,
 <Server: lisa_C2>,
 <Server: akshai_cmd_C1>,
 <Server: aham_docker>,
 <Server: vm1_alex>,
 <Server: girish2>,
 <Server: akshai_C1>,
 <Server: girish1>,
 <Server: julie-vm2>,
 <Server: s1>,
 <Server: Sotiris_notebook>,
 <Server: vm1>,
 <Server: yudulab2>,
 <Server: Julie-Lab-2>,
```

```
<Server: alex-lab2-test>,
<Server: Henkeinst-WSL>,
<Server: a_hameed>,
<Server: maha_script>,
<Server: c-2>,
<Server: shqi>,
<Server: RaheelInit>,
<Server: test_lab3>,
<Server: Aneysha-LAb2>,
<Server: emgo2250_L2>,
<Server: dapi_vm1>,
<Server: test_cont_Ellinor_C2>,
<Server: marcus_test>,
<Server: c2_kevaja>,
<Server: instance-kev>,
<Server: Ego_C2>,
<Server: Raheel_Docker>,
<Server: com_w1>,
<Server: com_m>,
<Server: alex-lab2>,
<Server: shreyas>,
<Server: G5_W1>,
<Server: G5_M>,
<Server: Sotiris_C2>,
<Server: Ellinor_c2_2>,
<Server: sai_lab2>,
<Server: Vera_C2>,
<Server: dapi1>,
<Server: JS>,
<Server: Marcus_L_C2>,
<Server: meritony>,
<Server: wezh_lab2>,
<Server: Stina_2>,
<Server: Ellinor_c2>,
<Server: dani_lab2_inst3>,
<Server: max_soneback_docker>,
<Server: vmTabeaDocker>,
<Server: maha_c2>]
```

Task - 3:

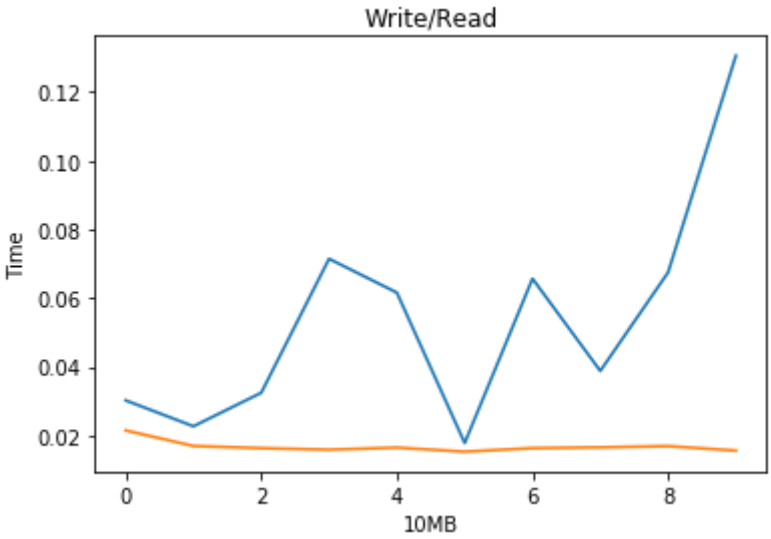
Try to measure the speed with which you can put and get files to and from volumes. Conduct your experiment several times to gather statistic and plot a) A estimated distribution of the time taken (in wall clock) to write and read a file of size 10MB in your volume and b) vary the size of the file from 10kB to 100MB and plot the read and write throughput (in MB/s) times as a function of file size (for the smaller data sizes, you might need to repeat the experiment many times and obtain a statistical average). Use "Root" disk as a reference point. Include the resulting graphs and a description of your experiment in the report.

In [3]:

```
#Task3
# a) A estimated distribution of the time taken (in wall clock) to write and read a

import sys
import time
import shutil
import os
import matplotlib.pyplot as plt

StartTime = time.time()
writeTime = [0] * 10
readTime = [0] * 10
i = 0

for i in range(10):
```

```
    StartTime = time.time()

    #Copying the file to the volume
    shutil.copy('/home/ubuntu/Timeplot/file_10mb.txt', '/data/www')
    writeTime[i] = time.time()-StartTime
    print('Write time is ' + str(writeTime[i]))
    #Reading the file from the volume
    StartTime = time.time()
    with open('/home/ubuntu/Timeplot/file_10mb.txt', 'r') as file:
        file.read()
    readTime[i] = time.time()-StartTime
    print('Read time is ' + str(readTime[i]))
    i = i+1


plt.title('Write/Read')
plt.plot(writeTime)
plt.ylabel('Time')
plt.xlabel('10MB')
plt.plot(readTime)
plt.show()
```

```
Write time is 0.030262231826782227
Read time is 0.02152085304260254
Write time is 0.0227205753326416
Read time is 0.01699209213256836
Write time is 0.03247499465942383
Read time is 0.01635599136352539
Write time is 0.07146072387695312
Read time is 0.01591014862060547
Write time is 0.061614274978637695
Read time is 0.01651620864868164
Write time is 0.017850637435913086
Read time is 0.015342473983764648
Write time is 0.06568598747253418
Read time is 0.016364336013793945
Write time is 0.0388336181640625
Read time is 0.01659107208251953
Write time is 0.06751179695129395
Read time is 0.016951560974121094
Write time is 0.13059544563293457
Read time is 0.015644550323486328
```



In [15]:

```
#b) vary the size of the file from 10kB to 100MB and plot the read and write through
# file from 10kB to 100MB

import shutil
import os
import sys
```

```
import time
import matplotlib.pyplot as plt

bytes = 1000000
StartTime = time.time()
x = 0
WriteTime= [0] * 8
ReadTime= [0] * 8
WriteThroughput = [0] * 8
ReadThroughput = [0] * 8

files = ['file_10kb', 'file_500kb', 'file_1mb', 'file_5mb', 'file_10mb', 'file_50mb'
for i in range (len(files)):
    StartTime = time.time()
    print(i)
    shutil.copy("/home/ubuntu/Timeplot/"+str(files[i])+".txt", "/data/www")


    WriteTime[x] = time.time()-StartTime

    StartTime = time.time()
    with open("/data/www/"+str(files[i])+".txt", 'r') as file:
        file.read()

    pathx = os.path.getsize("/data/www/" + str(files[i])+".txt")
    NewPath = pathx/bytes
    print(NewPath)
    throughputWrite = NewPath/WriteTime[i]
    WriteThroughput[i] = throughputWrite
    print("Throughput for write is " + str(throughputWrite))

    ReadTime[x] = time.time()-StartTime
    throughputRead = NewPath/ReadTime[i]
    ReadThroughput[i] = throughputRead
    print("Throughput read is " + str(throughputRead))
    x=x+1

print(files)
print(ReadTime)

plt.title('Write/Read')
plt.plot(files, WriteThroughput)
plt.plot(files, ReadThroughput)

plt.show()
```

```
0
0.01024
Throughput for write is 6.3619719982224865
Throughput read is 14.59384062521237
1
0.512
Throughput for write is 327.76001953601957
Throughput read is 403.5864777297501
2
1.048576
Throughput for write is 391.8081524368819
Throughput read is 528.4843200076905
3
5.24288
Throughput for write is 463.7528481909824
Throughput read is 641.7320616196341
4
10.48576
Throughput for write is 409.71516909226415
Throughput read is 705.1089414025075
```

```
5
52.4288
Throughput for write is 224.96447619862138
Throughput read is 613.0091645300313
6
83.88608
Throughput for write is 337.8399184294989
Throughput read is 452.86237151507277
7
104.8576
Throughput for write is 237.0187533367932
Throughput read is 557.0645001297017
['file_10kb', 'file_500kb', 'file_1mb', 'file_5mb', 'file_10mb', 'file_50mb', 'file_
80mb', 'file_100mb']
[0.0007016658782958984, 0.001268625259399414, 0.001984119415283203, 0.00816988945007
3242, 0.01487112045288086, 0.08552694320678711, 0.18523526191711426, 0.188232421875]
```

Don't foget to terminate resources after your experiments.

In [ ]:
```
# Clean up volumes
```