# ASSIGNMENT 8 – LAB 10: OPENSTACK

**MAIN TOPICS: Openstack, VMs, API**

First of all, I visit the web page: 137:204.107.63:22080

User: dist-sys-user1

PW: openstack4DistSys

(On my terminal the command to access with SSH (with the correct security groups to check for each instance after) is: **ssh -P 52022 dist-sys-user1@137.204.107.63**

**With the psswd: openstack4DistSys1**

The dashboard will open up. On the left side we have important components of the dashboard such as Compute and Network. From Compute we can see the instances that is our VM that runs in the program and that we have to create from existing images in it. The Network part is not so about the creation of a VM but more focused on the management of the network we are creating and simulating basing on the previous instance.

**Assignment**

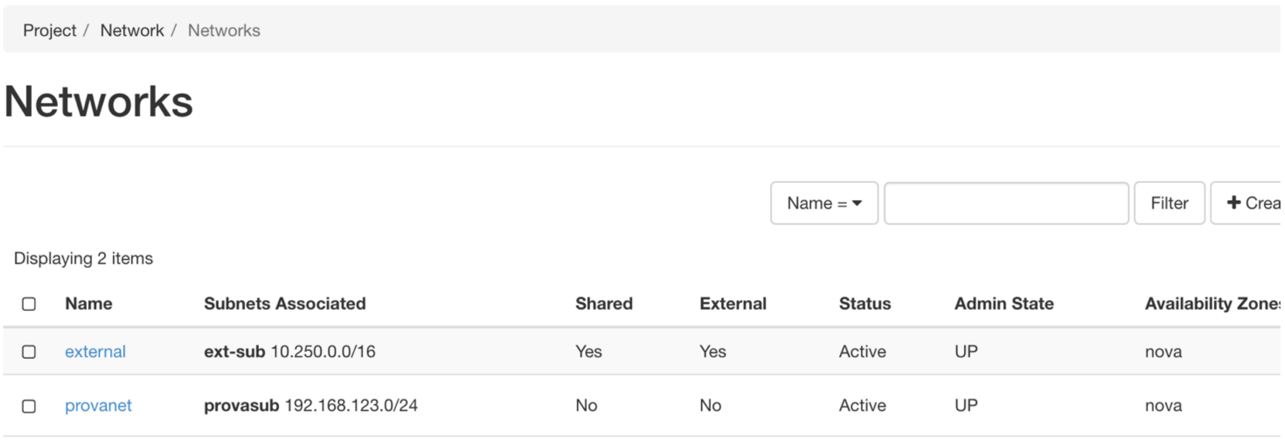
1) During the exam you should demonstrate you know how to create networks and VMs on OpenStack.

2) If the API Endpoint is at 10.15.2.1 (reachable from the VM) can we interact with the cluster via REST API?

Build a Python script that automates the creation of a network and a VM connected to it with REST API calls. The VM should have your name as ssh password.

1) Let's build a network and VM on Openstack:

I create a network called "provanet" with the following IP subnet: 192.168.123.0/24 from the Network>Networks>Create Network page. During the creation, I will have to associate a name to the subnet, too. ("provasub")



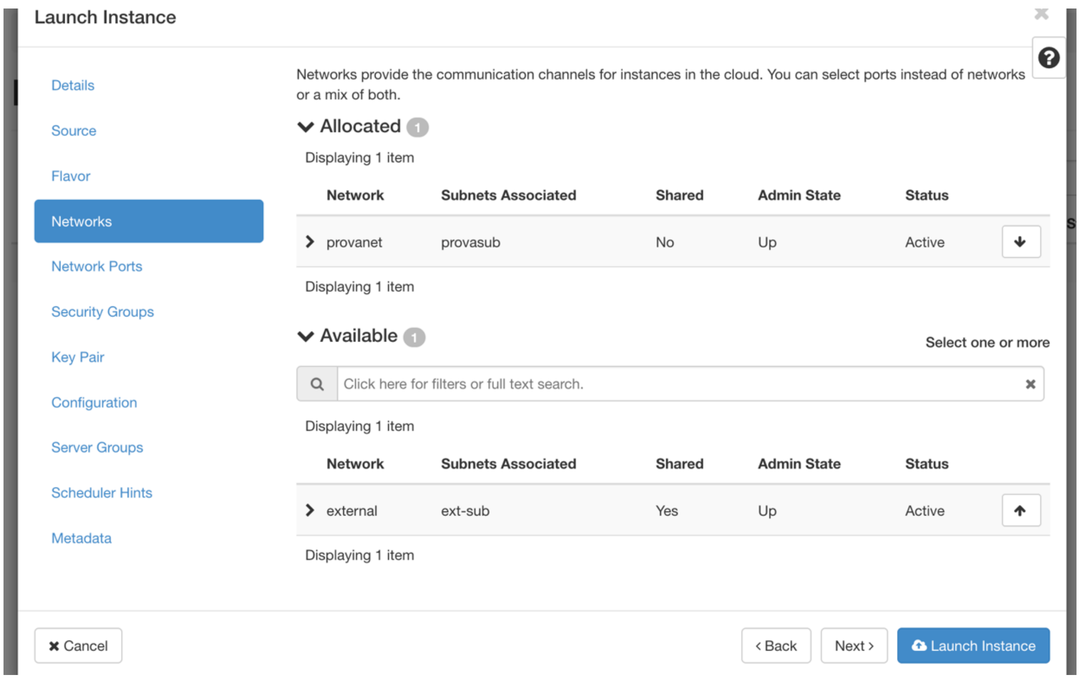
Displaying 2 items

| Name | Subnets Associated | Shared | External | Status | Admin State | Availability Zone |
|---|---|---|---|---|---|---|
| external | **ext-sub** 10.250.0.0/16 | Yes | Yes | Active | UP | nova |
| provanet | **provasub** 192.168.123.0/24 | No | No | Active | UP | nova |

Then I go to Compute>Instances>Launch Instance, to create an istance to be linked to my network as requested.

The instance page requires a lot of details to specify:

- Details: I give a name to the instance (e.g. “provainstance”)
- Source: it gives to the instance the disk to boot from. (e.g. Ubuntu22)
- Flavor: manages the size to compute (e.g. m1.small)
- Networks: it defines which networks should we link to the instance to be managed properly.



Launch Instance

Networks provide the communication channels for instances in the cloud. You can select ports instead of networks or a mix of both.

Allocated 1

Displaying 1 item

| Network | Subnets Associated | Shared | Admin State | Status |
|---|---|---|---|---|
| provanet | provasub | No | Up | Active |

Displaying 1 item

Available 1

Select one or more

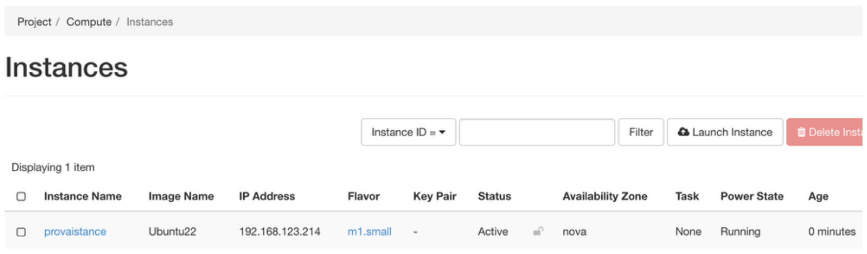
Click here for filters or full text search.

Displaying 1 item

| Network | Subnets Associated | Shared | Admin State | Status |
|---|---|---|---|---|
| external | ext-sub | Yes | Up | Active |

Displaying 1 item

- Configuration: I paste the script that gives the standard configuration to the cloud we’re working on (vd. Github) The file is called Cloud Init to pass a list of commands the VM will execute after the boot.



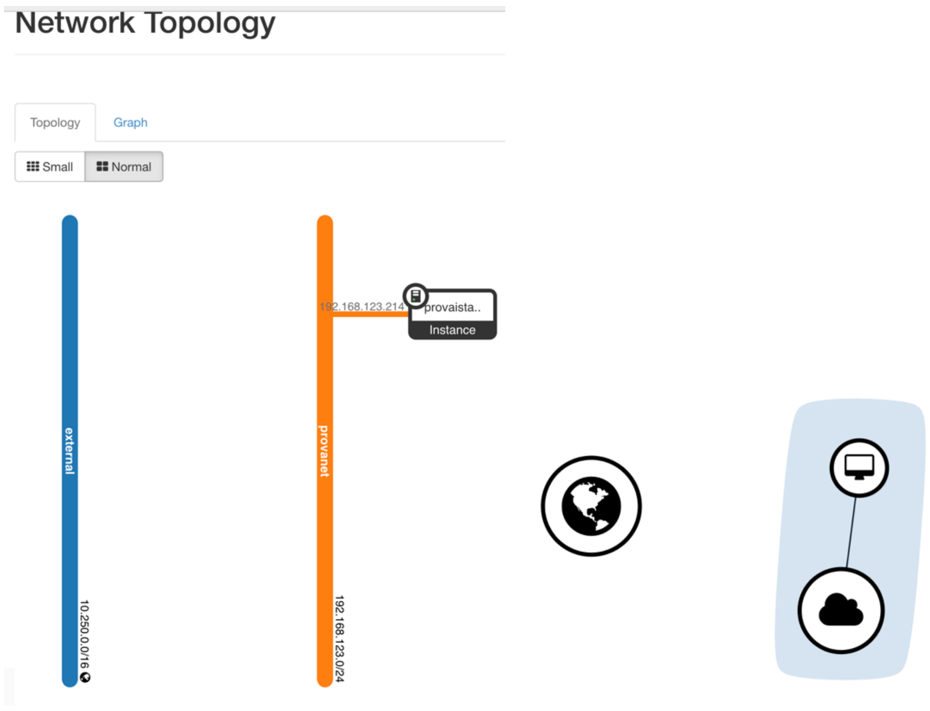
Project / Compute / Instances

Instances

Displaying 1 item

| Instance Name | Image Name | IP Address | Flavor | Key Pair | Status | Availability Zone | Task | Power State | Age |
|---|---|---|---|---|---|---|---|---|---|
| provaistance | Ubuntu22 | 192.168.123.214 | m1.small | - | Active | nova | None | Running | 0 minutes |

As we can check from the network topology the new network and instance have been created. To the instance, a random IP from the subnet chosen for the network has been decided by the programme itself, according to the configuration we gave to it.

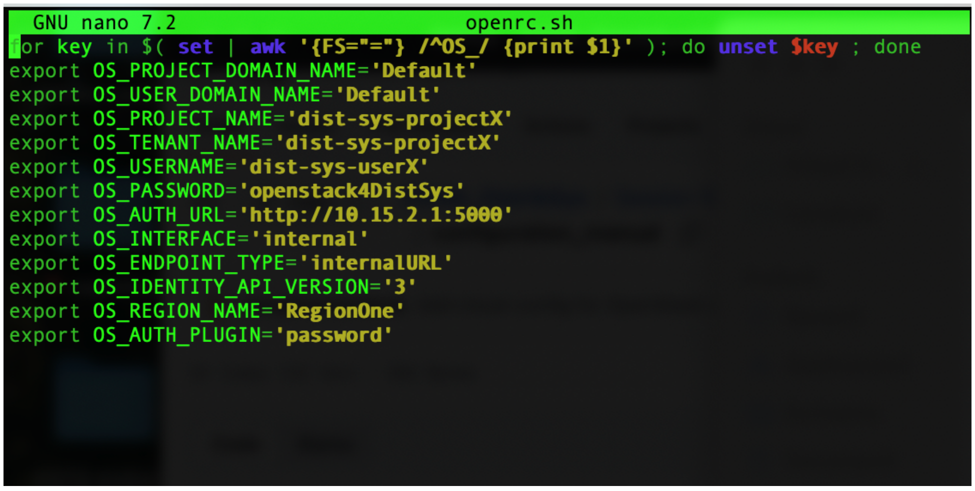


After having done this we can then go to the second part of the exercise.

2) If the API Endpoint is at 10.15.2.1 (reachable from the VM) can we interact with the cluster via REST API?

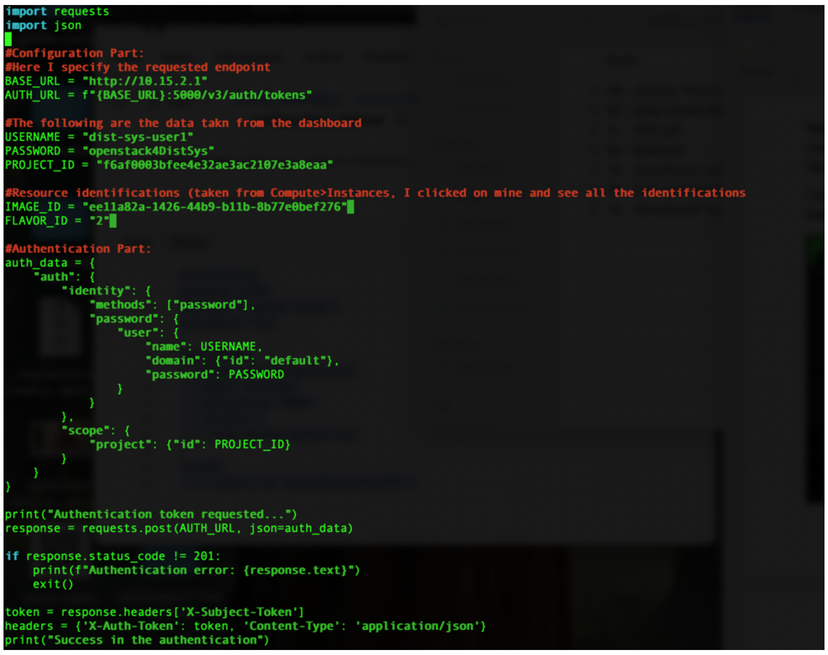
We can reach the API Endpoint from the VM by creating a Python file that automates the configuration of the network. I'll work from my terminal to link the VM to the http of OpenStack.

I've previously created two files, one (openrc.sh) that contains the exports for the connection between my VM and the URL of OpenStack. This file is used to start the OpenStack CLI from the SSH VM

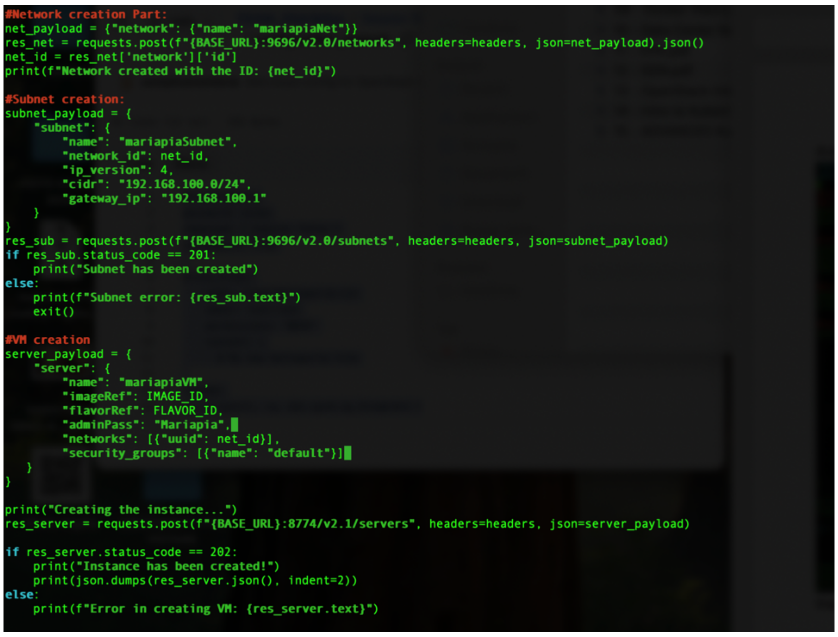


```
  GNU nano 7.2                              openrc.sh
for key in $( set | awk '{FS="="} /^OS_/ {print $1}' ); do unset $key ; done
export OS_PROJECT_DOMAIN_NAME='Default'
export OS_USER_DOMAIN_NAME='Default'
export OS_PROJECT_NAME='dist-sys-projectX'
export OS_TENANT_NAME='dist-sys-projectX'
export OS_USERNAME='dist-sys-userX'
export OS_PASSWORD='openstack4DistSys'
export OS_AUTH_URL='http://10.15.2.1:5000'
export OS_INTERFACE='internal'
export OS_ENDPOINT_TYPE='internalURL'
export OS_IDENTITY_API_VERSION='3'
export OS_REGION_NAME='RegionOne'
export OS_AUTH_PLUGIN='password'
```

And the other one contains the file python.

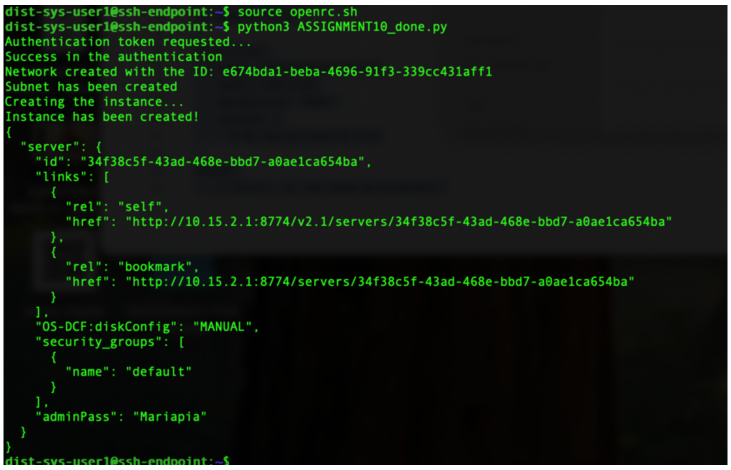


```
import requests
import json

#Configuration Part:
#Here I specify the requested endpoint
BASE_URL = "http://10.15.2.1"
AUTH_URL = f"{BASE_URL}:5000/v3/auth/tokens"

#The following are the data takn from the dashboard
USERNAME = "dist-sys-user1"
PASSWORD = "openstack4DistSys"
PROJECT_ID = "f6af0003bfee4e32ae3ac2107e3a8eaa"

#Resource identifications (taken from Compute>Instances, I clicked on mine and see all the identifications
IMAGE_ID = "ee11a82a-1426-44b9-b11b-8b77e0bef276"
FLAVOR_ID = "2"

#Authentication Part:
auth_data = {
    "auth": {
        "identity": {
            "methods": ["password"],
            "password": {
                "user": {
                    "name": USERNAME,
                    "domain": {"id": "default"},
                    "password": PASSWORD
                }
            }
        },
        "scope": {
            "project": {"id": PROJECT_ID}
        }
    }
}

print("Authentication token requested...")
response = requests.post(AUTH_URL, json=auth_data)

if response.status_code != 201:
    print(f"Authentication error: {response.text}")
    exit()

token = response.headers['X-Subject-Token']
headers = {'X-Auth-Token': token, 'Content-Type': 'application/json'}
print("Success in the authentication")
```



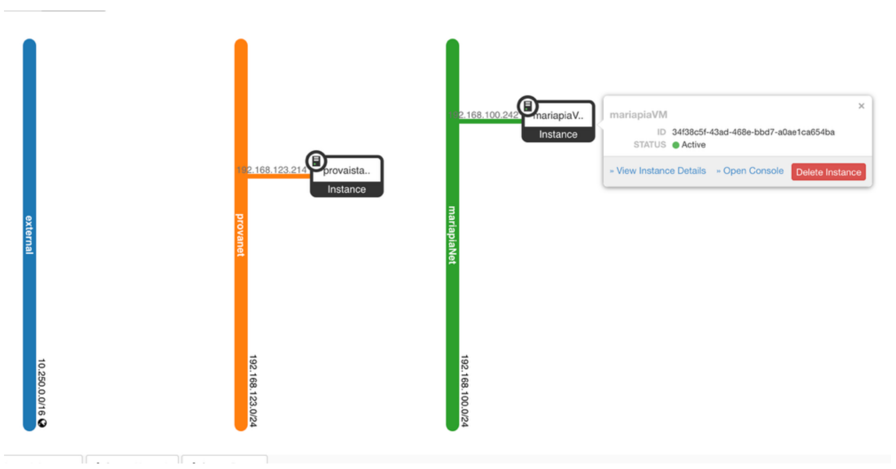
```
#Network creation Part:
net_payload = {"network": {"name": "mariapiaNet"}}
res_net = requests.post(f"{BASE_URL}:9696/v2.0/networks", headers=headers, json=net_payload).json()
net_id = res_net['network']['id']
print(f"Network created with the ID: {net_id}")

#Subnet creation:
subnet_payload = {
    "subnet": {
        "name": "mariapiaSubnet",
        "network_id": net_id,
        "ip_version": 4,
        "cidr": "192.168.100.0/24",
        "gateway_ip": "192.168.100.1"
    }
}
res_sub = requests.post(f"{BASE_URL}:9696/v2.0/subnets", headers=headers, json=subnet_payload)
if res_sub.status_code == 201:
    print("Subnet has been created")
else:
    print(f"Subnet error: {res_sub.text}")
    exit()

#VM creation
server_payload = {
    "server": {
        "name": "mariapiaVM",
        "imageRef": IMAGE_ID,
        "flavorRef": FLAVOR_ID,
        "adminPass": "Mariapia",
        "networks": [{"uuid": net_id}],
        "security_groups": [{"name": "default"}]
    }
}

print("Creating the instance...")
res_server = requests.post(f"{BASE_URL}:8774/v2.1/servers", headers=headers, json=server_payload)

if res_server.status_code == 202:
    print("Instance has been created!")
    print(json.dumps(res_server.json(), indent=2))
else:
    print(f"Error in creating VM: {res_server.text}")
```

I upload the variables in the file openrc.sh and then I run my python script.



```
dist-sys-user1@ssh-endpoint:~$ source openrc.sh
dist-sys-user1@ssh-endpoint:~$ python3 ASSIGNMENT10_done.py
Authentication token requested...
Success in the authentication
Network created with the ID: e674bda1-beba-4696-91f3-339cc431aff1
Subnet has been created
Creating the instance...
Instance has been created!
{
  "server": {
    "id": "34f38c5f-43ad-468e-bbd7-a0ae1ca654ba",
    "links": [
      {
        "rel": "self",
        "href": "http://10.15.2.1:8774/v2.1/servers/34f38c5f-43ad-468e-bbd7-a0ae1ca654ba"
      },
      {
        "rel": "bookmark",
        "href": "http://10.15.2.1:8774/servers/34f38c5f-43ad-468e-bbd7-a0ae1ca654ba"
      }
    ],
    "OS-DCF:diskConfig": "MANUAL",
    "security_groups": [
      {
        "name": "default"
      }
    ],
    "adminPass": "Mariapia"
  }
}
dist-sys-user1@ssh-endpoint:~$
```

Let's check if all went right also in the OpenStack dashboard...



mariapiaVM
ID 34f38c5f-43ad-468e-bbd7-a0ae1ca654ba
STATUS ● Active

The green one will be my new instance + network. As long as we could create it, I have put also the Security groups (in particular ssh to access from the secure shell protocol and the “All ICMP” to work on it through the web dashboard) by default (as seen in the output of the terminal, too)

If we wanted to connect it also to the external network, I should have added

EXTERNAL_NET_ID = "ID_netfromthedashboard" in the first part of the code.

```
server_payload = {
"server": {
        "name": "mariapiaVM",
        "imageRef": IMAGE_ID,
        "flavorRef": FLAVOR_ID,
        "adminPass": "Mariapia",
        "networks": [
                {"uuid": net_id},
                {"uuid": EXTERNAL_NET_ID} #here I should have put the ID of the external
                ],
                 "security_groups": [{"name": "default"}]
        }
 }
```