CSCI 5380

Network Virtualization and

Orchestration

Lab 2

OpenStack: Multi-tenants

University of Colorado Boulder

Department of Computer Science

Network Engineering

Professor Levi Perigo, Ph.D.

# Objective 2 – Auto-scaling application using Python

1. Scenario:

You are working in a cloud firm that has a single instance of an application running on OpenStack cloud platform. The firm is planning to add a functionality to the single running instance of the application that can autoscale/replicate itself to multiple instances whenever the compute capacity (eg. CPU cycles or memory) reaches a pre-defined threshold. Since you are familiar with the Python programming and REST API, you are being assigned a following task:

* 1. Write a simple Python application that can ssh into the available “cirros” instance that was created in the above objective and extract the CPU utilization information. [As an alternative, you may use ceilometer service for retrieving this telemetry data]
  2. If the CPU utilization or memory usage exceeds a threshold value, for example 20%, spin up additional instances of cirros. The creation of cirros instances should be triggered whenever the usage of CPU or memory exceeds a predefined threshold. Select CPU or memory usage to your interest to define your condition to trigger the creation of additional instances. In order to collect the utilization data, you’ll have to monitor its usage using appropriate commands.
  3. The Python application can use Nova REST API to create additional “Cirros” instances whenever the above condition occurs.
  4. The auto scaling of the instances should be handled considering following requirements:

**Max scaling size: 4** (this value denotes the maximum number of instances that should be spun)

**Increment size: 1** (this value denotes the number of instances that should be spun whenever CPU utilization exceeds threshold)

**Evaluation period: 40** (this value denotes the time period in seconds for monitoring CPU usage)

1. You can use the [Linux stress tool](https://www.tecmint.com/linux-cpu-load-stress-test-with-stress-ng-tool/) to raise the CPU utilization of an instance above the threshold.

To achieve this objective, I am monitoring the CPU usage for 40 seconds on instance in openstack via login into it using Netmiko. I am using the “top -bn1 | grep CPU:” to monitor the CPU usage. By using “if int(CPUuse.group(1)) > 20:” condition, I am comparing the current CPU usage with the predefine threshold value of 20. If condition is matched, I am executing “authenticate\_and\_create\_instance(auth\_options, instance\_details)” function in which I am creating the new instance named “New\_Lab2\_Obj1\_1”, flavor “m1.nano”, and, default image “cirros-0.6.2-x86\_64-disk”.

*Code snippet for creating new instance:*

A screenshot of a computer program

Description automatically generated

# Objective 4 – Network policies in OpenStack

# Summary:

In this objective, you will manage network policies within OpenStack.

# Section 1: VM and Virtual Network Setup

First you will need to create the configuration as is shown in Figure 1 above.

# Section 2: Achieve Inter-VN Communication

The default policy allows only intra-VN communication.

1. Figure out how to ping between VM’s in VN-A and VM’s in VN-B.
2. Figure out how to ping from the VM’s out to the Internet.

# Section 3: Network Policy Management

Now manage the network policy inside OpenStack, such that:

1. VN-A\_VM-1 can ping VN-B\_VM-1, but VN-A\_VM-2 cannot ping VN-B\_VM-1.
2. VN-B\_VM-1 can go out to the Internet, but VN-A\_VM-1 and VN-A\_VM-2 cannot.

A screenshot of a computer

Description automatically generated

Security rules can be added under Project > Network > Security Groups

A screenshot of a computer

Description automatically generated

In the above screenshot, rules are added on VN-A\_VM-1 to reach only VN-B\_VM-1 and VN-A\_VM-2 by allowing only the specific remote IP address.

A screenshot of a computer

Description automatically generated

In the above screenshot, rules are added on VN-A\_VM-2 to reach only VN-A\_VM-1 not VN-B\_VM-1 by allowing only the VN-A\_VM-1’s IP address.

A screenshot of a computer

Description automatically generated

In the above screenshot, rules are added on VN-B\_VM-1 to reach Internet and only VN-

A\_VM-1.

After creating all the security groups, they need to be associated with the instances.

SGA\_VM-1 assigned to VN-A\_VM-1.

A screenshot of a security group

Description automatically generated

SGA\_VM-2 assigned to VN-A\_VM-2.

A screenshot of a security group

Description automatically generated

SGB\_VM-1 assigned to VN-B\_VM-1.

A screenshot of a security group

Description automatically generated

Ping from VN-A\_VM-1 to VN-A\_VM-2, VN-B\_VM-1 and Internet.

A screenshot of a computer

Description automatically generated

Ping from VN-A\_VM-2 to VN-A\_VM-1, VN-B\_VM-1 and Internet.

A screenshot of a computer

Description automatically generated

Ping from VN-B\_VM-1 to VN-A\_VM-1, VN-A\_VM-2 and Internet.