Intelligent Cloud Operations

OpenStack Cloud Operating System

Large-scale AIOps Lab / SRE Dept.
Ireland and Munich Research Centers
Prof. Jorge Cardoso
jorge.cardoso@huawei.com

22.11.2020

1 Table of Contents

2 Setup the Infrastructure

2.1 Using VirtualBox

Download Oracle VM VirtualBox for your platform at:

- https://www.virtualbox.org/wiki/Downloads

VirtualBox enables creating, managing and running virtual machines (VMs) in your local environment.

Download Ubuntu 18.04.3 Bionic Beaver from:

- https://www.osboxes.org/ubuntu/

Since the file is large (>1G), it will take some time.

Start the Ubuntu VM in VirtualBox.

2.2 Technical University of Munich (TUM)

To deploy our OpenStack cloud we will use nested virtualization. We will deploy OpenStack virtualization inside the already virtualized environment OpenNebula. In other words, we run a hypervisor inside of a virtual machine (VM), which itself runs on a hypervisor.

Information on how to start a VM in OpenNebula provided by the LRZ Compute Cloud of the Leibniz Supercomputing Centre can be found at:

- https://www.lrz.de/services/compute/cloud_en/

Create a VM with the following characteristics:

- Memory size: 16 GB
- OS image: Ubuntu_20G
- CPU: 1 vCPU (or more)
- Network: connection to the Internet (optional)
  - Write down the public IP address, e.g.: 141.40.254.130
- Access:
  - Generate ssh key pairs
    - $ ssh-keygen –t rsa
  - Save the keys to files, e.g., as lrz and lrz.pub
  - Copy the public key lrz.pub to OpenNebula dashboard Context\SSH contextualization
- More information at:
  - https://www.lrz.de/services/compute/cloud_en/cloud-tutorial_en/

Test the access to your VM:

- ssh -i lrz -v root@141.40.254.130

3 Install Openstack

We will setup a complete non-production OpenStack environment using DevStack (another option would be to use PackStack). DevStack is composed by several scripts to install an all-in-one OpenStack environment on a single VM.

It is tested for most recent releases of Ubuntu, Fedora and CentOS/RHEL 7. We will use Ubuntu 18.04 since it is the most tested.

The services installed and configured by DevStack are keystone, glance, cinder, nova, neutron, and horizon.

The detailed instructions to follow are available at:

- https://docs.openstack.org/devstack/latest/

While the instructions are available in the previous link, we will repeat them here for completeness reasons.

We will add a non-root user, with sudo enabled, to run the install scripts:

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
$ sudo su - stack
```

DevStack can be easily downloaded from git:

```
$ git clone https://git.openstack.org/openstack-dev/devstack
$ cd devstack
```

The configuration of DevStack is done using the file local.conf. Create the file local.conf and write the passwords to be used for the deployment:

```
$ vi local.conf
    [[local|localrc]]
    ADMIN_PASSWORD=my_secret_password
    DATABASE_PASSWORD=$ADMIN_PASSWORD
    RABBIT_PASSWORD=$ADMIN_PASSWORD
    SERVICE_PASSWORD=$ADMIN_PASSWORD
```

To enable tracing using OSprofiler and Redis add the following line to local.conf:

```
$ vi local.conf
    enable_plugin osprofiler https://git.openstack.org/openstack/osprofiler
    master
    OSPROFILER_COLLECTOR=redis
```

To enable tracing using OSprofiler and Jaeger add the following line to local.conf:

```
$ vi local.conf
    enable_plugin osprofiler https://git.openstack.org/openstack/osprofiler
    refs/changes/67/611067/4
    OSPROFILER_BRANCH=refs/changes/67/611067/4
    OSPROFILER_COLLECTOR=jaeger
```

Start the installation process. This will take 15-20 minutes depending on your internet connection speed since many git trees and packages need to be download during the installation.

```
$ ./stack.sh
```

You can check the status of the OpenStack services running (e.g., glance, cinder, neutron, nova, and keystone) using the status command:

```
$ sudo systemctl status "devstack@*"
```

You can also inspect the code of OpenStack services, it is stored in the following directory:

```
$ ls -al /opt/stack/
```

You can access the Horizon dashboard and use the web interface to manage virtual machines, networks, volumes, and images:

- http://141.40.254.130/
- User Name: admin
- Password: admin (specified in the file local.conf)

To quickly understand how to use the UI to launch an instance, read the documentation available at:

- https://docs.openstack.org/horizon/latest/user/launch-instances.html

To shutdown DevStack, run:

```
$ ./unstack.sh
$ ./clean.sh
```

4 Prepare the CLI

Currently, there are two types of Command Line Interface (CLI) available:

- Integrated openstack CLI
- Native Client CLI for each openstack project

We will use the integrated CLI, set up the environment variables to access OpenStack, and take the role of admin by sourcing the openrc file with the parameters admin and admin:

```
$ cd /opt/stack/devstack
$ source openrc admin admin
```

As a first exercise, let us show the list of hypervisors (compute nodes) available in our deployment and display their characteristics (CPU and MEM):

```
$ openstack hypervisor list
$ openstack host show vm-141-40-254-130
$ openstack hypervisor show vm-141-40-254-130
```

Take the role of the demo user by souring openrc with the demo user:

```
$ source openrc demo demo
```

5 Launch Instances

A VM is a structure which brings together resources from several OpenStack services:

- Keypairs from Nova
  - https://docs.openstack.org/python-openstackclient/pike/cli/command-objects/keypair.html
- Security groups from Nova
  - https://docs.openstack.org/nova/latest/admin/security-groups.html#list-and-view-current-security-groups
- Flavors from Nova
  - https://docs.openstack.org/nova/pike/admin/flavors2.html
- Images from Glance
  - https://docs.openstack.org/glance/latest/admin/manage-images.html
- Networks from Neutron
  - https://docs.openstack.org/python-openstackclient/latest/cli/command-objects/network.html

Since most cloud images support public key authentication instead of password authentication, we will create a pair public-private SSH keys to access the VM to launch. OpenStack will inject the public key into the instance at boot time.

Create an SSH key-pair:

$ ssh-keygen
  - Use the file name: os_key
  - Passphrase: (empty)

Add the public key to OpenStack:

$ openstack keypair create --public-key ~/.ssh/os_key.pub os_key

Verify that the key was successfully added by listing the key registered:

$ openstack keypair list

VM instances have a firewall controlled by a security group, a container for a group of access rules that let specific networks from specific places through the firewall. By default, the security group applies to all instances and includes firewall rules that deny remote access to instances. We will a create a new security group named os_sg and apply it to the instance to launch.

$ openstack security group create os_sg
$ openstack security group list

To allow ICMP (ping) and secure shell (SSH) access to the VM, create two security rules:

$ openstack security group rule create --proto icmp os_sg
$ openstack security group rule create os_sg --protocol tcp --dst-port 22:22 --remote-ip 0.0.0.0/0
$ openstack security group rule list os_sg

Flavors define the compute, memory, and storage needed by instances. They represent an available hardware configuration for a VM. We can find the list of flavours available using the following command:

- $ openstack flavor list
  - Select the flavor m1.tiny

An image is a bootable disk that can be used to create a VM. It typically includes a OS kernel (Linux, Windows), libraries, utilities and configurations. Images are managed by the Glance service. To find an image use the command:

- $ openstack image list
  - Select image cirros-0.3.5-x86_64-disk

Select a private network. Every VM is connected to a private network which enables to communicate with other VMs and connect to the Internet through a router. Nonetheless, it is not possible for computers located outside the network (e.g., in the Internet) to initialize a connection to those instances:

- $ openstack network list
  - Select a private network, e.g., 9c5ae54a-5486-4e29-bb55-d4425c9837fa

Using the flavour, image, network, security group, and keypair previously identified, run the following command to create a VM and inspect its private IP:

- $ openstack server create --flavor m1.tiny --image cirros-0.3.5-x86_64-disk --nic net-id=9c5ae54a-5486-4e29-bb55-d4425c9837fa --security-group os_sg --key-name os_key os_vm01
- $ openstack server list
  - Take a note of the private IP: e.g., 10.0.0.6

Once the VM is running, access the instance using the Horizon dashboard via the console using VNC:

- $ openstack console url show os_vm01

Instances can also have a public, or floating IP address. Public addresses are used for communication with networks outside the cloud platforms such as the Internet. Create a new floating IP address:

- $ openstack network list
  - Take a note of the public network id: e.g., 2ea58d3b-f569-4474-bf28-f13ccc758eda
- $ openstack floating ip create 2ea58d3b-f569-4474-bf28-f13ccc758eda
  - Take a note of the floating IP created: e.g., 172.24.4.8
  - You can also refer to the network using its name
    - $ openstack floating ip create public

Assign the floating IP to the VM:

- $ openstack server add floating ip fc5b336f-e7ee-4f6f-8d52-2d3166b49ec6 172.24.4.8
- Using a most elaborate procedure:
  - $ openstack port list | grep 10.0.0.6

- $ openstack floating ip set --port 4e46b973-3746-4307-855a-75baa7bc659e --fixed-ip-address 10.0.0.6 172.24.4.8

Access the instance from the Internet using a floating IP and the private key:

```
$ ping -c 4 172.24.4.8
$ ssh -i ~/.ssh/os_key cirros@172.24.4.8
```

6 Attaching a Volume

The Cinder service provides volume storage. Network accessible volumes or virtual disks can be attached to a VM. Five steps are involved:

- Create the volume
- Attach the volume
- Format the volume
- Mount the volume
- Mount on reboot

To create the new volume named os_vol with 1Gb, run the following command:

```
$ openstack volume create --size 1 os_vol
```

The volume should now show up in OpenStack:

```
$ openstack volume list
```

To use the volume, it needs to be attached to a running VM. Attach the volume os_vol to os_vm01.

```
$ openstack server add volume os_vm01 os_vol
$ openstack volume list
```

The new volume needs a file system to store data. You can use ext3 or ext4 file systems (ext4 was introduced in 2008 with Linux Kernel 2.6.19 to replace ext3 and overcomes its limitations such as file size and number subdirectories).

Find the device name for the new volume. Generally, /dev/vda is the boot drive. Most likely, /dev/vdb will be the new volume.

```
$ ssh -i ~/.ssh/os_key cirros@172.24.4.8
$ ls /dev/vd*
```

Check that the size of the new volume matches the device:

```
$ sudo fdisk -l dev/vdb
```

Create a file system on the drive by running mkfs:

```
$ sudo mkfs.ext4 /dev/vdb
$ sudo lsblk -f
```

Create a directory /mnt/os_vol to mount the new file system /dev/vdb:

```
$ sudo mkdir -p /mnt/os_vol
$ sudo mount /dev/vdb /mnt/os_vol
$ df -h /mnt/os_vol
```

You can also make the mount permanent to survive reboots. Identify the UUID of the volume and modify /etc/fstab to have it auto mounted and execute:

$ sudo blkid /dev/vdb
$ sudo mount -a

For more information:

- https://docs.openstack.org/mitaka/install-guide-obs/launch-instance-cinder.html

7 Create a Network

You can access the Network Topology of the dashboard to displays a graphical representation of the interaction of your routers, networks, and instances.

- Horizon Dashboard \ Project \ Network \ Network Topology

To create a new empty private network named os_net to accept a subnet in the future, run the following command:

$ openstack network create os_net
$ openstack network list

Create a new subnet on top of the network created above. The subnet is named os_subnet, the DNS server to use is Google DNS, and the DHCP is enable.

$ openstack subnet create --subnet-range 192.168.101.0/24 --dhcp --gateway 192.168.101.1 --dns-nameserver 8.8.8.8 --network os_net os_subnet
$ openstack network list
    - Take a note of the os_net id: e.g., 6f81d5d5-435c-4bf8-bde3-3bbad634e5c9

Create a new instance on the private network. Identify the flavor, security group, image, key-pair, and network to use.

$ openstack flavor list
$ openstack security group list
$ openstack image list
$ openstack keypair list
$ openstack network list

Create an instance named os_vm02:

$ openstack server create --flavor m1.tiny --image cirros-0.3.5-x86_64-disk --nic net-id=6f81d5d5-435c-4bf8-bde3-3bbad634e5c9 --security-group os_sg --key-name os_key os_vm02

Add the subnet to an existing router:

$ openstack server list
    - Take a note of private IP: e.g., 192.168.101.9
$ openstack router list
    - Take a note of the router name, e.g., router1
$ openstack subnet list

    - Take a note of the os_net id: e.g., 91a77e1c-99f8-4bcc-bf91-dc2c3763f99a

$ openstack router add subnet router1 91a77e1c-99f8-4bcc-bf91-dc2c3763f99a

    - or: $ openstack router add subnet router1 os_subnet

Add a public floating IP address to the VM. The private IP will be used for communication between instances and the public address to communicate with networks outside the cloud platform. To list all floating IP addresses allocated to your project, run:

$ openstack floating ip list

    - Take a note of a free floating IP: e.g., 172.24.4.14
    - Or: $ openstack floating ip create public

Associate a floating IP address with the instance os_vm02 and access the VM:

$ openstack server add floating ip os_vm02 172.24.4.14
$ ssh -i ~/.ssh/os_key cirros@172.24.4.14

The instance is now associated with two IP addresses:

$ openstack server list --ip 10.0.0.

Delete the new VM, floating ip, and network:

$ openstack server remove floating ip os_vm02 172.24.4.14
$ openstack floating ip delete 172.24.4.14
$ openstack router remove subnet router1 91a77e1c-99f8-4bcc-bf91-dc2c3763f99a
$ openstack server delete os_vm02
$ openstack network delete os_net
$ openstack router show router1

More examples are available at:

- https://developer.openstack.org/firstapp-libcloud/networking.html

8 Distributed Tracing

8.1 Redis

Add to the file …./stack/devstack/local.conf

- enable_plugin osprofiler https://opendev.org/openstack/osprofiler master
- OSPROFILER_COLLECTOR=redis

You can also install RedisDesktopManager to manage the traces which will be installed stored in Redis.

8.2 Jaeger

Add to the file …./stack/devstack/local.conf

- enable_plugin osprofiler https://git.openstack.org/openstack/osprofiler refs/changes/67/611067/4
- OSPROFILER_BRANCH=refs/changes/67/611067/4

- OSPROFILER_COLLECTOR=jaeger

8.3 Generate traces

Run the commands from the previous sections by appending --os-profile SECRET_KEY at the end of each command. For example:

```
$ openstack image list --os-profile SECRET_KEY
$ openstack hypervisor list --os-profile SECRET_KEY
$ openstack host show <hypervisor hostname>
$ openstack volume list --os-profile SECRET_KEY
$ openstack server create --flavor m1.tiny --image cirros-0.3.5-x86_64-disk --nic net-id=<network_id> --security-group os_sg --key-name os_key os_vm01 --os-profile SECRET_KEY
$ openstack server add volume os_vm01 os_vol --os-profile SECRET_KEY
```

OS profiler is not currently working with networking:

```
$ openstack subnet create --subnet-range 192.168.101.0/24 --dhcp --gateway 192.168.101.1 --dns-nameserver 8.8.8.8 --network os_net os_subnet
```

8.4 Display traces

osprofiler CLI can be used to list traces or get single traces into, e.g., html or json:

```
$ osprofiler trace list --connection-string redis://localhost:6379
$ osprofiler trace show --connection-string redis://localhost:6379 --html <trace-id> --out <some_name>.html
$ osprofiler trace show --connection-string redis://localhost:6379 --json <trace-id> --out <some_name>.html
```

To display a trace using Jaeger UI:

- http://localhost:16686

Use the shortened trace id printed in the search box of Jaeger UI and search for the trace.

9 Jupyter

Install and run jupyter:

```
$ sudo apt install jupyter jupyter-notebook
$ jupyter notebook
```

10 Further Reading

- Become an expert at using the CLI
  - https://developer.openstack.org/api-guide/quick-start/
- Writing Your First OpenStack Application using an SDK
  - https://developer.openstack.org/firstapp-libcloud/
- Try various SDKs
  - Python LibCloud

        - https://libcloud.apache.org
    - Go Gophercloud
        - https://github.com/gophercloud/gophercloud
- User stories
    - https://www.openstack.org/user-stories/