Lab 9.2: Using the OpenStack SDK
IN720 Virtualisation

September 21, 2018

Introduction

In this lab we will use the OpenStack SDK to access services in our and launch a VM instances. Our process will be to

1. Get a connection
2. Identify our image
3. Identify our flavour
4. Find a suitable network
5. Identify a keypair
6. Launch an instance
7. Associate a floating IP address with the instance.

1 Preliminary steps

Before you start coding, create a subdirectory of your home directory to hold your work, unless you have already done so. In that directory, create a `clouds.yml` file similar to the one we looked at in the last lab. Activate your Python virtualenv and make sure you can create a working connection object like we saw last time. In the code below, we will assume that you have created an OpenStack connection object and associated it with the variable `conn`.

2 Identify resources

Before we can launch our instances, we need to find the needed OpenStack resources. Start by creating variables in your code with the resource names.

```
IMAGE = 'ubuntu-16.04-x86_64'
FLAVOUR = 'c1.c1r1'
NETWORK = 'private-net'
KEYPAIR = '<your keypair name>'
```

Next, we will use our connection object to get the resources with these names from our cloud.

```
image = conn.compute.find_image(IMAGE)
flavour = conn.compute.find_flavor(FLAVOUR)
network = conn.network.find_network(NETWORK)
keypair  = conn.compute.find_keypair(KEYPAIR)
```

Now we have objects that represent the specific resources we need to launch an instance. Note that if you don't have a keypair ready to use on the cloud, then you can use the SDK to create one. These calls generally raise `ResourceNotFound` exception if the requested resource is not found.

3 Launch an instance

With our resources in hand, launching an instance is straightforward:

```
SERVER = '<your username>-server'
server = conn.compute.create_server(
    name=SERVER, image_id=image.id, flavor_id=flavour.id,
    networks=[{"uuid": network.id}], key_name=keypair.name)
```

Notice that the `networks` argument is specified slightly differently than the others. This is because we can connect a server to multiple networks, so we pass in a list of them.

The call is completed *asynchronously*, meaning it may take some time before the instance is actually launched. We cannot proceed to the next step until this is done, so we will wait:

```
server = conn.compute.wait_for_server(server)
```

4 Associate a floating IP address

Floating IP addresses are available on our public network, so we will access the network and get an ip address from it:

```
public_net = conn.network.find_network('public-net')
floating_ip = conn.network.create_ip(floating_network_id=public_net.id)
```

Now we can associate the address with the server

```
conn.compute.add_floating_ip_to_server(server, floating_ip.floating_ip_address)
```

5 Cleaning up

Don't forget to remove your instance from the cloud. Seeing how to do this with Python is left as an exercise. See

`https://docs.openstack.org/openstacksdk/latest/user/index.html`

for more information.