

OpenStack Debugging Training – Student Lab Book

Setup

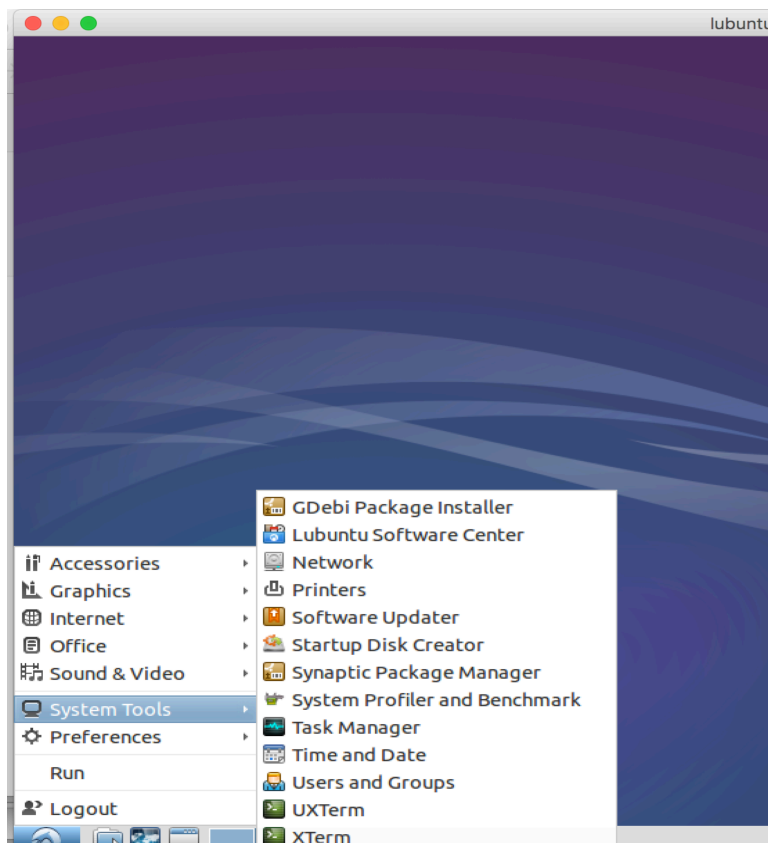
1. Setup Devstack VM prior to the class using instructions from <https://github.com/txdev/OpenStack-Debugging>

Using Devstack

Start the VM from VirtualBox by clicking on the “Start” button. The VM will boot up and automatically logs you in.

Starting Devstack

- Open *xterm* (start menu -> system tools -> xterm)



- Run following commands from the *xterm*
 - `cd ~/devstack`
 - `./restart-devstack.sh`

Navigating Devstack Screens

In devstack, each OpenStack process is started in a virtual terminal called *screen*. You can navigate among screens using following keystrokes:
(Eg: ctrl A + N means hold “control A” and press n)

- Go to next screen - *ctrl A + N*
- Go to previous screen - *ctrl A + P*
- List all screens - *ctrl A + “*
- Detach from screen - *ctrl A + D*
- Go to screen 9 - *ctrl A + 9*

Using Horizon

- Open Firefox
- Goto URL - <http://localhost/horizon>
- Login with admin/welcome (Login could be slow, be patient)
- Try options such as images, networks, instances

Using CLI

Openstack CLI provides tools to interact with various sub-systems

- Open a terminal and try commands such as
 - `nova list`
 - `neutron net-list`
 - `glance image-list`

Using PyCharm

Open Terminal and run following commands to start PyCharm.

- `cd ~/pycharm*`
- `bin/pycharm.sh &`

PyCharm already has two projects configured – Neutron and Horizon. You can create a new project by simply opening the correct OpenStack folder. As an example, to create “Nova” project, run these commands from PyCharm

- File -> Open
- Select `/opt/stack/nova` folder
-

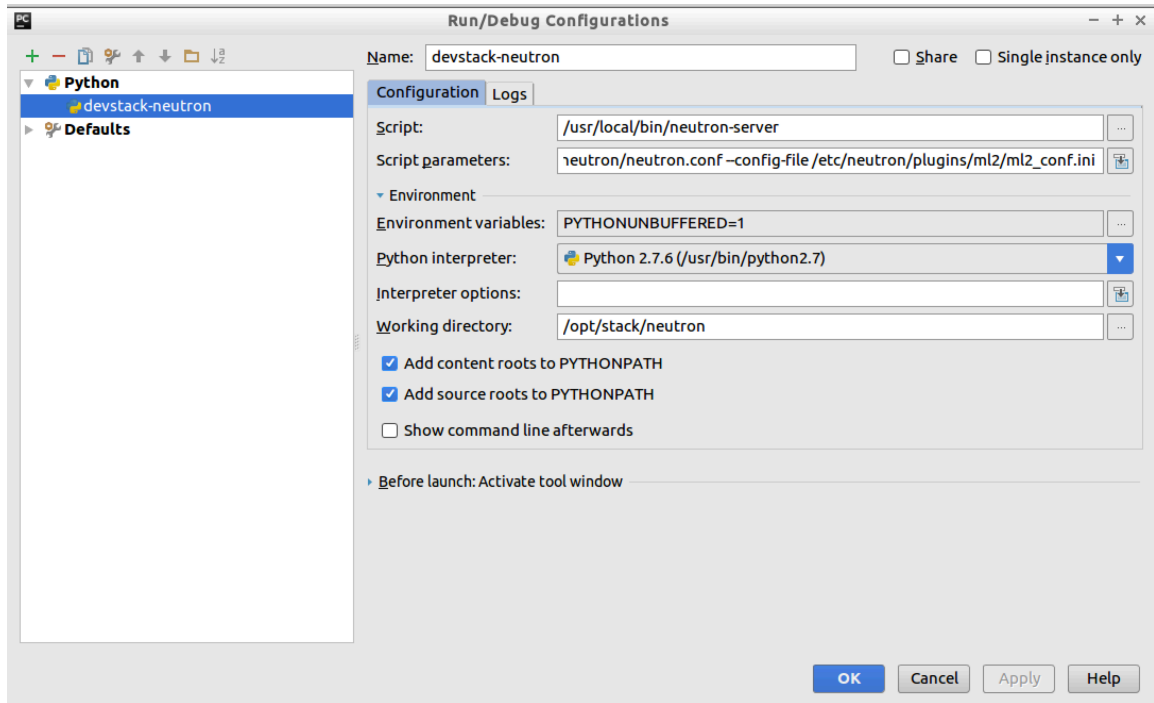
It may take a while to open the new project since PyCharm has to index all files.

- Open Neutron and Horizon projects

- Setting up debug configuration

Exercise 1 - Debug Neutron “Create Network”

- Open Neutron project in PyCharm
- Make sure that following debug configuration is present. If not, create new one.



Configuration details:

```
script -> /usr/local/bin/neutron-server
script parameters -> --config-file /etc/neutron/neutron.conf --
config-file /etc/neutron/plugins/ml2/ml2_conf.ini
Working directory -> /opt/stack/neutron
```

- Discuss the monkey patch (check the file neutron/common/eventlet_utils.py line number 32)
- Put a break point at neutron/neutron/plugins/ml2/plugin.py in update_port() method

Create a VM:

A VM can be created using Horizon GUI or OpenStack command line.

To create from command line, gather information about the following:

- Image name
 - Run “glance image-list” to get list of images
- Flavor name
 - Run “nova flavor-list” to get list of flavors
- Network id
 - Run “neutron net-list” to get list of networks

Run the following command to start a VM:

```
> nova boot --image <image name> --flavor <flavor name> --nic net-id <network-id>
<VM Name>
```

Work through the PyCharm debugger

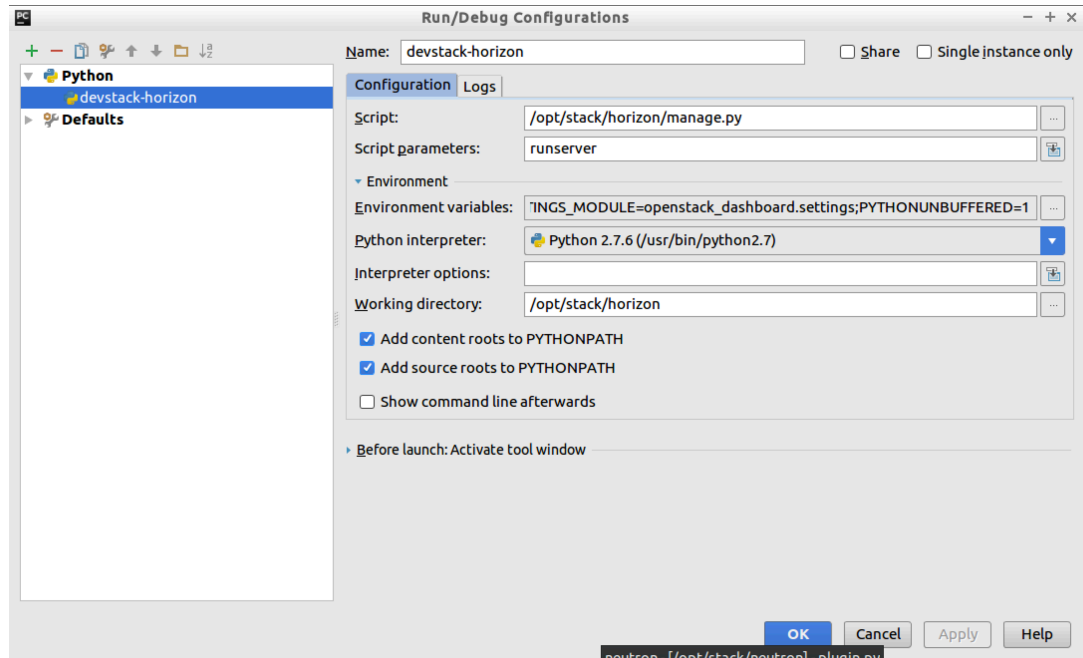
Exercise 2 - Debug Neutron “list network”

- Put a break point at neutron/neutron/plugins/ml2/plugin.py in get_networks() method
- List network from Horizon or CLI
- Work through the PyCharm debugger

Exercise 3 - Debug Horizon

When you start Horizon debugging from PyCharm, the webserver will be started on port 8000. However, you still have to login to <http://localhost> to establish the session.

- Open Horizon project in PyCharm
- Check the debug configuration



Configuration details:

Script -> /opt/stack/horizon/manage.py

Script parameters -> runserver

Env variables ->

SETTINGS_MODULE=openstack_dashboard.settings;PYTHONUNBUFFERED=1

Python interpreter -> /usr/bin/python2.7

Working directory -> /opt/stack/horizon

- Put a break point at
/opt/stack/horizon/openstack_dashboard/dashboards/project/networks/t
ables.py
- Start the debugger
- Point the browser to <http://localhost> and login with username/password of
admin/welcome to create the session.
- Point the browser to <http://localhost:8000>
- Work through the debugger

Exercise 4 – Debug using pdb

Check l option

- Stop the Neutron server from the screen
- Edit file *neutron/neutron/plugins/ml2/plugin.py* and add this code in the
get_networks() method


```
import pdb
pdb.set_trace()
```

- Start the neutron from the screen by recalling the previous command

Exercise 5 – Adding logging statements

OpenStack services use standard logging levels – DEBUG, INFO, AUDIT, WARNING, ERROR, CRITICAL and TRACE.

To disable DEBUG-level logging, edit `/etc/nova/nova.conf` and add

```
debug=false
```

Adding a debug statement

Add a custom debug statement in `neutron/neutron/plugins/ml2/plugin.py` in `get_networks()` method:

```
LOG.debug("entering get_networks")
```

Restart the neutron from the Screen and run the command “`neutron net-list`”. Observe the console of screen to see the above message getting printed.

Exercise 6 – Development workflow

There are many ways to setup a development workflow, where you can test your code changes using Devstack. As an easy solution, you can make the code changes directly under `/opt/stack` and restart services.

A better option is setting up your own local repository and pushing the changes to Devstack. Please see these instructions for one possible workflow.

1. Remove `/opt/stack/horizon` directory
 - a. `rm -r /opt/stack/horizon`
2. Create `horizon.git` repository
 - a. `mkdir horizon.git`
 - b. `cd horizon.git`
 - c. `git init --bare`
3. Clone horizon
 - a. `git clone https://git.openstack.org/openstack/horizon`
4. Add remote
 - a. `git remote add remote file:///home/foundry/horizon.git`
5. Make code changes and commit
 - a. `cd horizon`

- b. `git commit -am "comments"`
 - c. `git push remote master`
- 6. Clone your repository
 - a. `cd /opt/stack`
 - b. `git clone file:///home/foundry/horizon.git`
- 7. Get updates
 - a. `cd /opt/stack`
 - b. `git pull`