

OpenStack Debugging Training – Student Lab Book

Setup

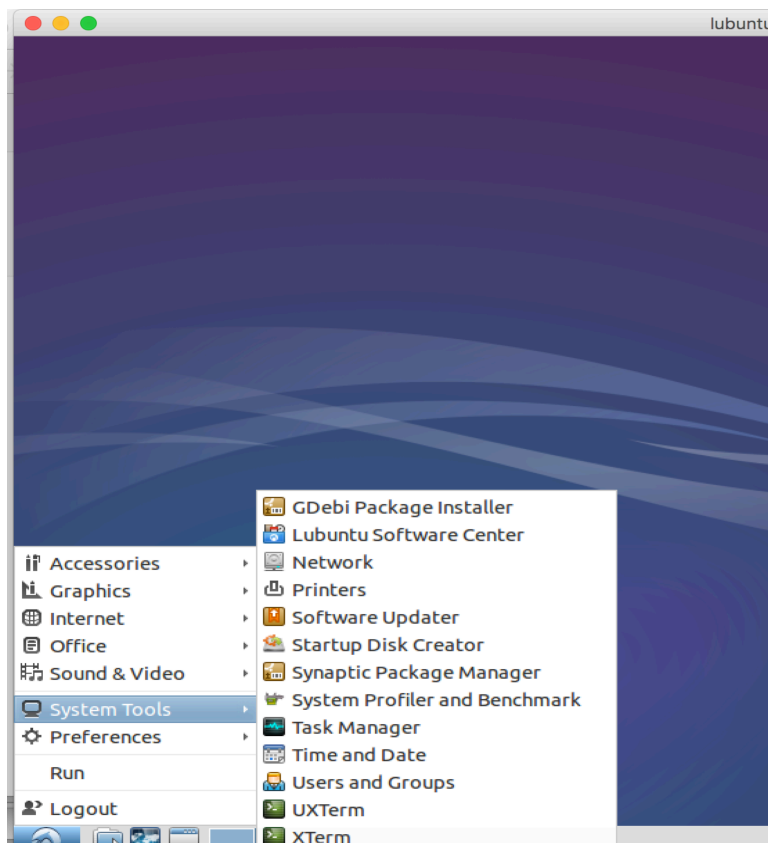
Setup Devstack VM prior to the class using instructions from <https://github.com/txdev/OpenStack-Debugging>

Using Devstack

Start the VM from VirtualBox by clicking on the “Start” button.
The VM will boot up and automatically log you in. If the screen gets locked, use the password ‘foundry123’.

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/foundry123 (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 &`

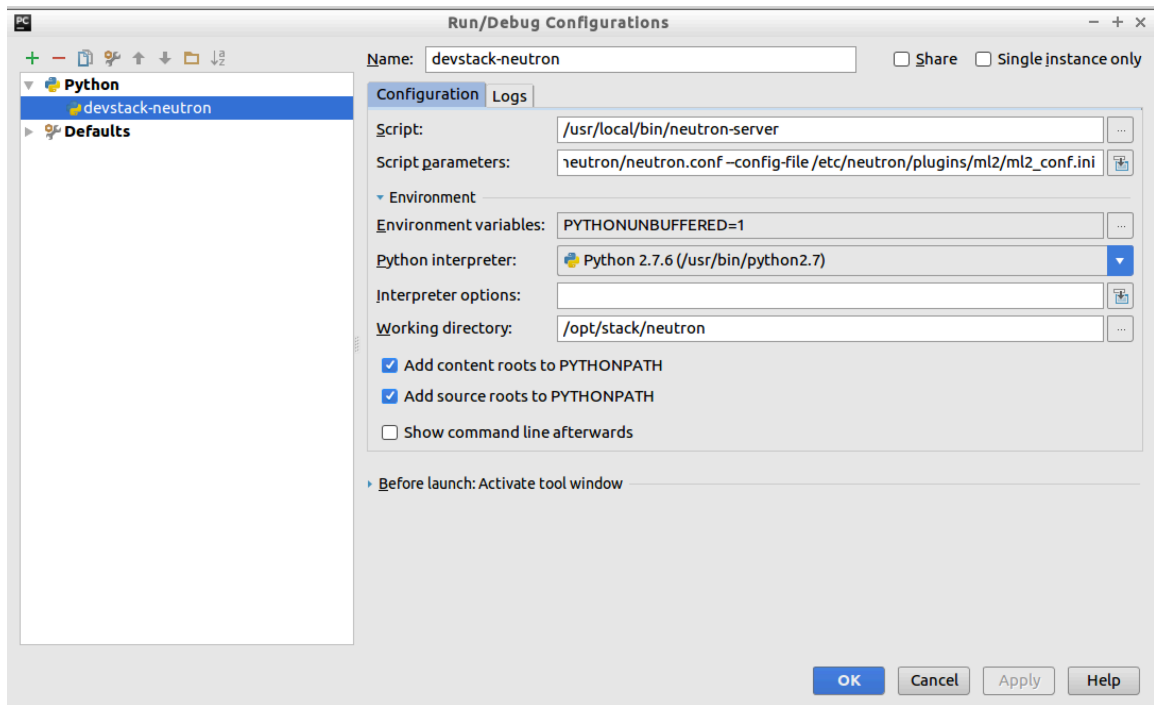
PyCharm already has two projects configured – Neutron and Horizon. You can create a new project by simply opening the correct OpenStack folder. As an example, to create “Nova” project, run these commands from PyCharm

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

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

Exercise 1 - Debug neutron “port binding”

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



Configuration details:

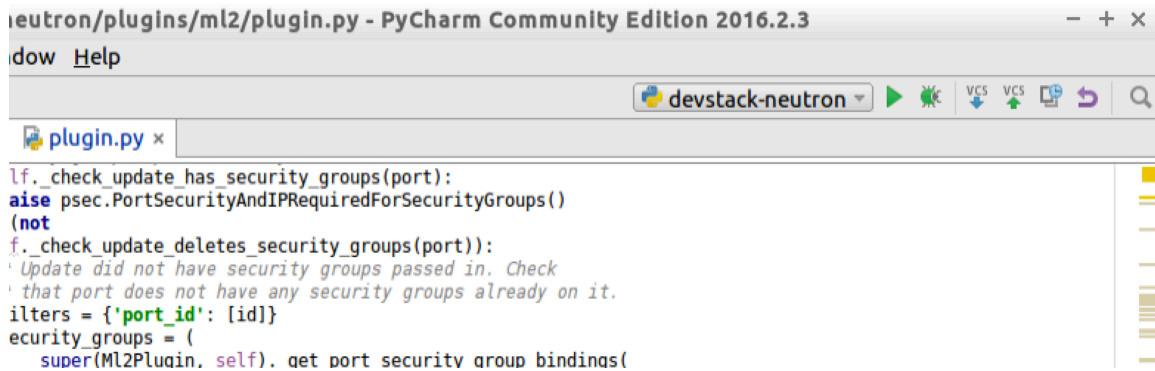
```
script ->/usr/local/bin/neutron-server
script parameters -> --config-file /etc/neutron/neutron.conf --
config-file /etc/neutron/plugins/ml2/ml2_conf.ini
Working directory -> /opt/stack/neutron
```

- 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

Start Debugging

Go to the screen for “q-svc” and do Ctrl +C to stop the neutron server.

Start Neutron Debugging from PyCharm by clicking on the “Debug devstack-neutron” icon.



```
neutron/plugins/ml2/plugin.py - PyCharm Community Edition 2016.2.3
File Edit View Window Help
devstack-neutron
plugin.py x
if _check_update_has_security_groups(port):
    raise psec.PortSecurityAndIPRequiredForSecurityGroups()
    (not
    f._check_update_deletes_security_groups(port)):
    'Update did not have security groups passed in. Check
    ' that port does not have any security groups already on it.
    filters = {'port_id': [id]}
    security_groups = (
        super(Ml2Plugin, self).get_port_security_group_bindings(
```

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>
```

Example:

```
>nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny
--nic net-id=01328cal-0c25-42ab-8da1-49c9f927aadae TestVM1
```

Work through the PyCharm debugger.

Exercise 2 - Debug Neutron “list network”

- Remove all existing break points by clicking “Run -> View Breakpoints” and unchecking all existing breakpoints
- Put a break point at `get_networks()` method in the file `neutron/neutron/plugins/ml2/plugin.py`
- Stop the existing debugging session and start new (Alternatively, you can try “Rerun Devstack” option)
- List network from Horizon or CLI (`neutron net-list`)
- Work through the PyCharm debugger

Cleanup:

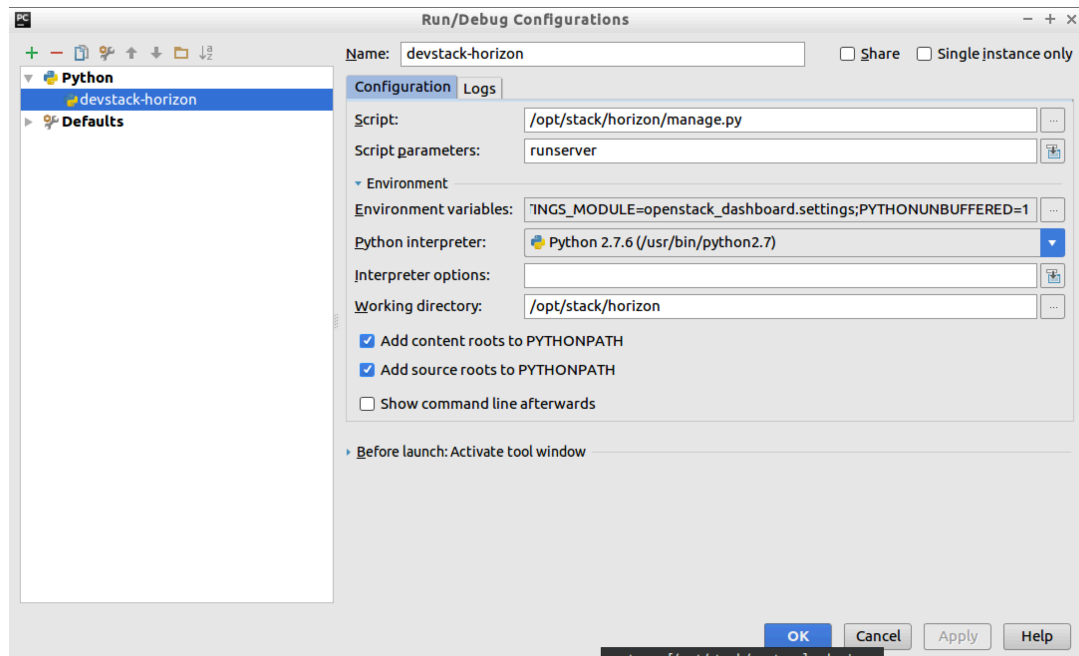
- Stop the Neutron debugger and close Neutron project
- From the neutron screen, restart the neutron service (Recall the previous command using up arrow key and hit enter)

Exercise 3 - Debug Horizon

Open Horizon project from PyCharm.

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 `get_subnets()` function in `/opt/stack/horizon/openstack_dashboard/dashboards/project/networks/tables.py` file
- Start the debugger
- Point the browser to <http://localhost> and login with username/password of admin/foundry123 to create the session.
- Point the browser to <http://localhost:8000>
- From the Horizon dashboard, click on Project -> Network -> Networks
- You will be automatically switched to the PyCharm debugger. Work through the debugger.

Cleanup

- Close Horizon project from PyCharm by choosing File->Close Project

Exercise 4 – Debug using pdb

- 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
- Run the following command
 - `neutron net-list`
- Observe the console of Neutron screen and you may see many debug statements. Press “Enter” and you will be placed on pdb prompt.
- Common pdb commands
 - Navigating the code
 - `l(list)` – list 11 lines surrounding the current line
 - `w(here)` – display file and line number of current line
 - `n(ext)` – execute current line
 - `s(tep)` – step into function
 - `r(eutrnr)` – return from the current function
 - Controlling execution
 - `b <line no>` - create breakpoint at line number
 - `b` – list break points
 - `c(ontinue)` – execute until a breakpoint is encountered
 - `clear [#]` – clear break point of index #
 - Others
 - `p <name>` - print value of the variable name
 - `!<expr>` - execute the expression
 - `run args` – restarts the debugger with arguments
 - `q(uit)` – quit the debugger

Exercise 5 – Adding logging statements

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

Adding a debug log

Add a custom debug statement at `get_networks()` in the file `neutron/neutron/plugins/ml2/plugin.py` in 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 to 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. `mv /opt/stack/horizon /opt/stack/horizon.old`
2. Create horizon.git repository
 - a. `mkdir horizon.git`
 - b. `cd horizon.git`
 - c. `git init --bare`
3. Clone horizon
 - a. `cd`
 - b. `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. Make code changes
 - c. `git commit -am "comments"`
 - d. `git push remote master`
6. Clone your repository
 - a. `cd /opt/stack`
 - b. `git clone file:///home/foundry/horizon.git`

7. On-going code updates
 - a. Update code using step 5
 - b. `cd /opt/stack`
 - c. `git pull`