Table of Contents

[Devstack setup steps: 3](#_Toc423504708)

[OS 3](#_Toc423504709)

[Install git 3](#_Toc423504710)

[Get Devstack branch 3](#_Toc423504711)

[Configure 3](#_Toc423504712)

[Local.conf file 3](#_Toc423504713)

[Using CLI 4](#_Toc423504714)

[Modification to a running OpenStack environment 4](#_Toc423504715)

[Cleanup 4](#_Toc423504716)

[Maintenance 5](#_Toc423504717)

[Scripts 5](#_Toc423504718)

[stack.sh 5](#_Toc423504719)

[unstack.sh 5](#_Toc423504720)

[rejoin-stack.sh 5](#_Toc423504721)

[clean.sh 6](#_Toc423504722)

[OpenDaylight 6](#_Toc423504723)

[Install ODL Helium 6](#_Toc423504724)

[Restart ODL 6](#_Toc423504725)

[OpenVSwitch 6](#_Toc423504726)

[Restart 6](#_Toc423504727)

[List Flows 6](#_Toc423504728)

[Configuring Internet access for Deployed VMs 6](#_Toc423504729)

[Setup Configuration 7](#_Toc423504730)

[Host configuration 7](#_Toc423504731)

[Launch Instance 7](#_Toc423504732)

[Potential issue with connecting to VMs via SSH 8](#_Toc423504733)

[Multi-Node/Multi-Host Devstack setup steps: 8](#_Toc423504734)

[Cluster Controller Node (Master Host) 8](#_Toc423504735)

[Compute Node 8](#_Toc423504736)

# Devstack setup steps:

## OS

Use Ubuntu 14.04 LTS Server. If installed on a VM, make sure to allocate lots of resources. Recommendation is 8 cores, 32G ram, 200G disk.

## Install git

Make sure git is installed on your machine:

sudo apt-get install git

## Get Devstack branch

Clone the Juno release into your local machine:

git clone <https://github.com/openstack-dev/devstack.git> -b stable/juno

## Configure

See <http://docs.openstack.org/developer/devstack/configuration.html> for more details.

Create local.conf under devstack with the following services if you intend to use Neutron as the networking service and would like to integrate OpenDayLight with Neutron:

Note: There is a default local.conf under devstack/samples/ which can be used as a template. The following services can then be added to it.

## Local.conf file

[[local|localrc]]

DEST=/opt/stack

# By default ``stack.sh`` output only goes to the terminal where it runs.  It can  
# be configured to additionally log to a file by setting ``LOGFILE`` to the full  
# path of the destination log file.  A timestamp will be appended to the given name.  
LOGFILE=$DEST/logs/stack.sh.log  
  
# Old log files are automatically removed after 7 days to keep things neat.  Change  
# the number of days by setting ``LOGDAYS``.  
LOGDAYS=7  
  
# Nova logs will be colorized if ``SYSLOG`` is not set; turn this off by setting  
# ``LOG\_COLOR`` false.  
LOG\_COLOR=False

RECLONE=yes

# If the ``\*\_PASSWORD`` variables are not set here you will be prompted to enter  
# values for them by ``stack.sh`` and they will be added to ``local.conf``.

ADMIN\_PASSWORD=admin123

MYSQL\_PASSWORD=$ADMIN\_PASSWORD

RABBIT\_PASSWORD=$ADMIN\_PASSWORD

SERVICE\_PASSWORD=$ADMIN\_PASSWORD

SERVICE\_TOKEN=$ADMIN\_PASSWORD

HOST\_IP=<your host IP>

Q\_PLUGIN=ml2

Q\_ML2\_PLUGIN\_MECHANISM\_DRIVERS=opendaylight,logger

Q\_ML2\_TENANT\_NETWORK\_TYPE=gre

LIBVIRT\_TYPE=kvm

DISABLED\_SERVICES=n-net

ENABLED\_SERVICES+=,q-svc,q-agt,q-dhcp,q-l3,q-meta,neutron,odl-server,odl-compute

Remember that you need to make sure the HOST\_IP is the current IP of your machine.

1. Run ./stack.sh
2. Open a web browser and connect to the following URL to browse OpenStack Dashboard:

http://<your controller ip address>

1. Open a web browser and connect to the following URL to browse OpenDayLight console:

http://<your controller ip address>:8080

# Using CLI

To use the CLI you need to set some environment variables. Create a file openrc.sh with the following content. Make sure you replace your host ip

#!/bin/bash

export OS\_AUTH\_URL=http:// <your host IP>:5000/v2.0

export OS\_TENANT\_NAME="demo"

export OS\_USERNAME="admin"

export OS\_PASSWORD=admin123

To use the variables run the script, or run source openrc.sh

You should be able to run openstack service command lines like nova/glance etc

Alternatively, you can run “source eucarc [username] [tenantname]”

# Modification to a running OpenStack environment

When there is an already running environment, for any modification of the configuration (local.conf, etc.) to take effect, the following script should be executed: ./rejoin-stack.sh

## Cleanup

One step command to re-stack:

./unstack.sh && ./clean.sh && sudo rm -rf /opt/stack && ./stack.sh

In order to undo the devstack installation, one should run ./unstack.sh script. Furthermore to clean the database, etc completely, running ./clean.sh is recommended.

You can also remove the /opt/stack folder to completely clean openstack

Note: Running unstack.sh and clean.sh will wipe whatever resources that exist in your working configuration (such as VMs, networks, etc) Therefore if that is not intended and only a change in the config is desired, ./rejoin-stack.sh should be invoked.

## Maintenance

The OpenStack repos on Github get updated all the time. If you want to stay up-to-date with these changes, I recommend periodically doing a git pull request in each of the directories that get stuffed into /opt/stack/. You can kill your instance of OpenStack before you do this by doing a killall screen.

You can also use the rejoin-stack.sh script if you've already run the script and are wanting to fire it back up again after killing it.

## Scripts

<http://docs.openstack.org/developer/devstack/#scripts>

stack.sh  
We have run this script initially to install Openstack. This script allows you to specify configuration options of what git repositories to use, enabled services, network configuration and various passwords. You need to run this script everytime you make any change to the OpenStack configuration or enable any new service.

unstack.sh   
Stops all the services started by stack.sh (mostly) mysql and rabbit are left running as OpenStack code refreshes do not require them to be restarted. Note that any changes made to the code should remain intact after running unstack.sh unless set otherwise.

rejoin-stack.sh   
This script rejoins an existing screen, or re-creates a screen session from a previous run of stack.sh. This means that if you have previously run stack.sh and you want to rejoin the session you run rejoin-stack.sh. Doing this restores all the data from the previous sessions. So do not be scared of losing your data after rebooting your machine.   
Just run ./unstack.sh   
then, reboot your machine  
when you want to join the session again run ./rejoin-stack.sh  
And you have all your data in place! You can check by accessing the dashboard on the same URL as before.

Dev stack not accessible after restart

ip addr del “ip-addr/mask” (ex:10.71.85.111/24) dev “interface”(ex:eth0)

ip addr add “ip-addr/mask” (ex:10.71.85.111/24) dev “interface”(ex:br-ex)

clean.sh  
Does its best to eradicate traces of a Grenade run except for the following: - both base and target code repos are left alone - packages (system and pip) are left alone. This is used if you want to erase all the OpenStack related files from your system.

# OpenDaylight

## Install ODL Helium

Full Instructions to install Helium is [here](https://wiki.opendaylight.org/view/OVSDB:Helium_and_Openstack_on_Fedora20). Adding just the key steps in this document.

First get Helium ODL by using the below command in /home/vmidc

Once stack completes, attach to the screen session by using

screen –r

Goto ODL screen by using CTRL + A + N until you get to “odl-server” screen and type

feature:install odl-base-all odl-aaa-authn odl-restconf odl-nsf-all odl-adsal-northbound odl-mdsal-apidocs odl-ovsdb-openstack odl-ovsdb-northbound odl-dlux-core

At this point you should be able to navigate to <http://HOSTIP:8181> to get to ODL.

Detach from the screen session by pressing CTRL + A + D

## Restart ODL

cd /opt/stack/opendaylight/distribution-karaf-0.2.0-Helium/bin

./stop

./start

./status

# OpenVSwitch

## Restart

sudo service openvswitch-switch restart

## List Flows

sudo ovs-ofctl -O OpenFlow13 dump-flows br-int

# Configuring Internet access for Deployed VMs

To enable internet access from the deployed server instances in devstack you need to add the following lines to local.conf.

This basically configures a private network, connects it to router and connects this router to a public network specified in the configuration file.

## Setup Configuration

# Everything below till the Q\_FLOATING\_ALLOCATION\_POOL is optional

# this is needed only for enabling connectivity from the VM’s to the internet

PUBLIC\_NETWORK\_GATEWAY=<your gateway IP>Example: 10.71.85.252

FLOATING\_RANGE=<your external network> Example: 10.71.85.0/24

FIXED\_RANGE=<your fixed internal>(optional) Example: 10.0.0.0/24

FIXED\_NETWORK\_SIZE=<your fixed internal range size>(optional) Example: 256

NETWORK\_GATEWAY=<your internal gateway>(optional) Example: 10.0.0.1

Q\_FLOATING\_ALLOCATION\_POOL=start=<exernal ip pool start>,end=<exernal ip pool start>

After stack.sh completes, you need to do the following.

## Host configuration

The Openstack host needs to have uplink interface which is connected to the internet. Make sure this interface does NOT have an ip address set.

You will need to run the following command. This command basically add a port on the vswitch to forward traffic from the br-ex interface to your uplink interface

sudo ovs-vsctl add-port br-ex <your uplink interface name>

To build up the arp table, you need to ping from the router in openstack to the external gateway.

run

vmidc@ubuntu-openstack-85-arv:~/devstack$ sudo ip netns

qrouter-79d63b08-d181-4c3d-8c8e-3b7c81ac1228

Using the qrouter id in the above command run the following

sudo ip netns exec qrouter-79d63b08-d181-4c3d-8c8e-3b7c81ac1228 ping <your gateway IP>

Note: The above steps have been verified on a multi node (specifically 2-node) setup as well. These steps need to be applied **only to the controller node** and no change is needed on the compute node. All the VMs regardless of what node they reside on will get have the external connectivity. For the basic multi node setup refer to a following section of this document.

## Launch Instance

Launch a server instance with the server connected to the private network.

Open the instance options menu and select “Associate Floating Ip”.

Click on the “+” then in the next dialog select the “public” combobox item.

Click close.

The instance now should have a private ip and a floating public ip associated with it.

You should be able to ping the internet from the VM and ping the VM from the outside network

## Potential issue with connecting to VMs via SSH

We need to make sure that our VM allows external ssh with clear password authentication. To verify this look at the following file (assuming your VM is running on Ubuntu): /etc/ssh/sshd\_config

In this file make sure PasswordAuthentication is set to yes.

If you ended up modifying this file, restart the sshd deamon:

sudo /etc/init.d/ssh stop

sudo /etc/init.d/ssh start

# Multi-Node/Multi-Host Devstack setup steps:

## Cluster Controller Node (Master Host)

This node will contain all the required services as mentioned in the above normal single-host setup environment. Basically, this “master” node will mainly have Neutron, Nova, and ODL Controller services.

## Compute Node

This node will have Nova and “agent” services that point back to the “master” services. Following is a snapshot of the local.conf file configured for this node:

Q\_PLUGIN=ml2  
Q\_ML2\_PLUGIN\_MECHANISM\_DRIVERS=opendaylight  
Q\_ML2\_TENANT\_NETWORK\_TYPE=gre

SERVICE\_HOST=<IP address of master host>  
Q\_HOST=<IP address of master host>

ENABLED\_SERVICES=cinder,neutron,n-cpu,rabbit,n-novnc  
VNCSERVER\_PROXYCLIENT\_ADDRESS=<IP address of Compute Node>  
VNCSERVER\_LISTEN=0.0.0.0

enable\_service q-agt  
enable\_service odl-compute