# Install OpenStack Folsom

Bilel Msekni (bilel.msekni@telecom-sudparis.eu)

October 10, 2012

## Contents

# 1 Initialize

**Note:** Some commands were too long so i had to split them into two lines. Beware when you copy paste

Go into *sudo* mode

```
sudo su
```

Add the OpenStack Folsom repositories to your ubuntu repositories:

```
##### The following command is splitted to two, reform before executing #####
echo deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/folsom main
>> /etc/apt/sources.list.d/folsom.list
apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 5EDB1B62EC4926EA
```

Update your system:

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

Install the mySQL along with other stuffs:

```
apt-get install vlan bridge-utils ntp mysql-server python-mysqldb
```

Configure mySQL to receive all incoming requests:

```
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
service mysql restart
```

Configure the NTP server to synchronize between your compute nodes and the controller node:

```
nano /etc/ntp.conf

#Replace server ntp.ubuntu.com with :

ntp.ubuntu.com iburst
nserver 127.127.1.0
nfudge 127.127.1.0 stratum 10

#Restart the service
service ntp restart
```

Enable IPv4 Forwarding by uncommenting the line *net.ipv4.ip_forward=1*

```
nano /etc/sysctl.conf
#Uncomment net.ipv4.ip\_forward=1
```

## 2 Prepare for networking

Give your two NICs static addresses by modifying */etc/network/interfaces*:

```
auto eth0
 iface eth0 inet static
 address 157.159.100.232
 netmask 255.255.255.0
 gateway 157.159.100.1

auto eth1
 iface eth1 inet static
 address 10.0.0.3
```

## 3 Keystone

This is how we install OpenStack's identity service:

```
apt-get install keystone python-keystone python-keystoneclient
```

remove the default database:

```
rm /var/lib/keystone/keystone.db
```

Create a new MySQL database for keystone:

```
mysql -u root -p
CREATE DATABASE keystone;
GRANT ALL ON keystone.* TO 'keystoneUser'@'%' IDENTIFIED BY 'keystonePass';
quit;
```

Adapt the *connection* attribute in the */etc/keystone/keystone.conf* to the new database

```
connection = mysql://keystoneUser:keystonePass@157.159.100.232/keystone
```

Restart the identity service then synchronize the database:

```
service keystone restart
keystone-manage db_sync
```

Fill up the keystone database using the two scripts available. Beware that you MUST modify the *HOST_IP* variable before executing the scripts:

```
chmod +x keystone_basic.sh
chmod +x keystone_endpoints_basic.sh
./keystone_basic.sh
./keystone_endpoints_basic.sh
```

Create a simple credential file and load it so you won't be bothered later:

```
nano creds
#Paste the following:
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_pass
export OS_AUTH_URL="http://157.159.100.232:5000/v2.0/"
# Load it:
source creds
```

To test Keystone, we use a simple curl request:

```
apt-get install curl openssl
curl http://157.159.100.232:35357/v2.0/endpoints -H 'x-auth-token: ADMIN'
```

# 4 Glance

After installing Keystone, we continue with installing image storage service a.k.a Glance:

```
apt−get install glance python−glance python−glanceclient
```

Delete the default database and create a new MySQL database for Glance:

```
rm /var/lib/glance/glance.sqlite
mysql −u root −p
CREATE DATABASE glance;
GRANT ALL ON glance.∗ TO ’glanceUser’@’%’ IDENTIFIED BY ’glancePass’;
quit;
```

Update */etc/glance/glance-api-paste.ini* with:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 157.159.100.232
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = service_pass
```

Update the */etc/glance/glance-registry-paste.ini* with:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 157.159.100.232
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = service_pass
```

Update */etc/glance/glance-api.conf* with:

```
sql_connection = mysql://glanceUser:glancePass@157.159.100.232/glance
```

and

```
[paste_deploy]
flavor = keystone
```

Update the */etc/glance/glance-registry.conf* with:

```
sql_connection = mysql://glanceUser:glancePass@157.159.100.232/glance
```

and

```
[paste_deploy]
flavor = keystone
```

Restart the glance-api and glance-registry services:

```
service glance−api restart; service glance−registry restart
```

Synchronize the glance database:

```
glance−manage db_sync
```

Restart the services again to take into account the new modifications:

```
service glance−registry restart; service glance−api restart
```

To test Glance's well installation, we upload a new image to the store: start by downloading an ubuntu cloud image to your node and then uploading it to Glance

```
mkdir images
cd images
wget http://uec−images.ubuntu.com/releases/precise/release/ubuntu−12.04−server−cloudimg−amd64.tar.gz
tar xzvf ubuntu−12.04−server−cloudimg−amd64.tar.gz
##### The following command is splitted to two, reform before executing #####
glance add name="Ubuntu" is_public=true container_format=ovf disk_format=qcow2
< precise−server−cloudimg−amd64.img
```

Now list the images to see what you have just uploaded:

```
glance image−list
```

## 5 KVM

KVM is needed as the hypervisor that will be used to create virtual machines. Before you install KVM, make sure that your hardware enables virtualization:

```
apt−get install cpu−checker
kvm−ok
```

Normally you would get a good response. Now, move to install kvm and configure it:

```
apt−get install −y kvm libvirt−bin pm−utils
```

Edit the */etc/libvirt/qemu.conf file* and uncomment:

```
cgroup_device_acl = [
"/dev/null", "/dev/full", "/dev/zero",
"/dev/random", "/dev/urandom",
"/dev/ptmx", "/dev/kvm", "/dev/kqemu",
"/dev/rtc", "/dev/hpet","/dev/net/tun"
]
```

Delete default virtual bridge :

```
virsh net−destroy default
virsh net−undefine default
```

Enable live migration by updating */etc/libvirt/libvirtd.conf* file :

```
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"
```

Edit libvirtd_opts variable in */etc/init/libvirt-bin.conf* file

```
 env libvirtd_opts="−d␣−l"
```

Edit */etc/default/libvirt-bin* file :

```
libvirtd_opts="−d␣−l"
```

Restart the libvirt service to load the new values:

```
service libvirt-bin restart
```

## 6 OpenVSwitch

Install the openVSwitch:

```
apt-get install -y openvswitch-switch openvswitch-datapath-dkms
```

Create the bridges:

```
ovs-vsctl add-br br-int
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-eth1 eth1
ovs-vsctl add-br br-ex
ovs-vsctl add-port br-ex eth2
```

## 7 Quantum

Instead of diving into the dark world of networking, the quantum project enables rich networking topologies with minimal configuration overhead:

Start by installing the rabbitMQ server:

```
apt-get install rabbitmq-server
```

Install the Quantum server and the Quantum OVS plugin:

```
apt-get install quantum-server python-cliff python-pyparsing
apt-get install quantum-plugin-openvswitch
```

Create a database:

```
mysql -u root -p
CREATE DATABASE quantum;
GRANT ALL ON quantum.* TO 'quantumUser'@'%' IDENTIFIED BY 'quantumPass';
quit;
```

Edit the OVS plugin configuration file:

```

  nano /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini

[DATABASE]
sql_connection = mysql://quantumUser:quantumPass@157.159.100.232/quantum
[OVS]
tenant_network_type=vlan
network_vlan_ranges = physnet1:1:4094
bridge_mappings = physnet1:br-eth1
```

Restart the quantum server:

```
service quantum-server restart
```

Install the OVS plugin agent

```
apt-get install quantum-plugin-openvswitch-agent
```

Intall quantum DHCP and l3 agents:

```
apt-get -y install quantum-dhcp-agent
apt-get -y install quantum-l3-agent
```

Edit */etc/quantum/api-paste.ini*

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 157.159.100.232
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = quantum
admin_password = service_pass
```

In addition, update the */etc/quantum/l3_agent.ini*

```
auth_url = http://157.159.100.232:35357/v2.0
auth_region = RegionOne
admin_tenant_name = service
admin_user = quantum
admin_password = service_pass
```

Restart all the services:

```
service quantum-server restart
service quantum-plugin-openvswitch-agent restart
service quantum-dhcp-agent restart
service quantum-l3-agent restart
```

## 8 Nova

Start by installing nova components:

```
apt-get install -y nova-api nova-cert nova-common novnc nova-compute-kvm
apt-get install -y nova-consoleauth nova-scheduler nova-novncproxy
```

Prepare a Mysql database for Nova:

```

rm /var/lib/nova/nova.sqlite
mysql -u root -p
CREATE DATABASE nova;
GRANT ALL ON nova.* TO 'novaUser'@'%' IDENTIFIED BY 'novaPass';
quit;
```

Now modify authtoken section in the */etc/nova/api-paste.ini* file to this:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 157.159.100.232
auth_port = 35357
auth_protocol = http
```

```
admin_tenant_name = service
admin_user = nova
admin_password = service_pass
signing_dirname = /tmp/keystone-signing-nova
```

Modify the *nova.conf* like this:

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
scheduler_driver=nova.scheduler.simple.SimpleScheduler
s3_host=157.159.100.232
ec2_host=157.159.100.232
ec2_dmz_host=157.159.100.232
rabbit_host=157.159.100.232
cc_host=157.159.100.232
nova_url=http://157.159.100.232:8774/v1.1/
sql_connection=mysql://novaUser:novaPass@157.159.100.232/nova
ec2_url=http://157.159.100.232:8773/services/Cloud
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
use_deprecated_auth=false
auth_strategy=keystone
keystone_ec2_url=http://157.159.100.232:5000/v2.0/ec2tokens
# Imaging service
glance_api_servers=157.159.100.232:9292
image_service=nova.image.glance.GlanceImageService


# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://157.159.100.232:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=127.0.0.1
vncserver_listen=0.0.0.0

# Network settings
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://157.159.100.232:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=service_pass
quantum_admin_auth_url=http://157.159.100.232:35357/v2.0
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver

# Compute #
compute_driver=libvirt.LibvirtDriver
```

```

# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
```

Don't forget to update the ownership rights of the nova directory:

```
chown −R nova. /etc/nova
chmod 644 /etc/nova/nova.conf
```

Add this line to the sudoers file:

```
sudo visudo
#Paste this line anywhere you like:
nova ALL=(ALL) NOPASSWD:ALL
```

Synchronize your database:

```
nova−manage db sync
```

Restart nova-* services and start the iscsi service:

```
cd /etc/init.d/; for i in $( ls nova−∗ ); do sudo service $i restart; done
service novnc restart
```

Check for the smiling faces on nova-* services to validate your installation:

```
 nova−manage service list
```

# 9 Cinder

Cinder is the newest OpenStack project and it aims at managing the volumes for VMs. Although Cinder is a replacement of the old nova-volume service, its installation is now a seperated from the nova install process.

```
apt−get install cinder−api cinder−scheduler cinder−volume iscsitarget
apt−get install open−iscsi iscsitarget−dkms
```

Prepare a Mysql database for Cinder:

```
rm /var/lib/cinder/cinder.sqlite
mysql −u root −p
CREATE DATABASE cinder;
GRANT ALL ON cinder.∗ TO 'cinderUser'@'%' IDENTIFIED BY 'cinderPass';
quit;
```

Configure */etc/cinder/api-paste.init* like the following:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_protocol = http
service_host = 157.159.100.232
service_port = 5000
auth_host = 157.159.100.232
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = service_pass
```

and edit the */etc/cinder/cinder.conf* to:

```
[DEFAULT]
rootwrap_config=/etc/cinder/rootwrap.conf
sql_connection = mysql://cinderUser:cinderPass@157.159.100.232/cinder
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper=ietadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
#osapi_volume_listen_port=5900
```

Finally, synchronize your database

```
cinder-manage db sync
```

Don't forget to create a volumegroup and name it *cinder-volumes*.

```
dd if=/dev/zero of=cinder-volumes bs=1 count=0 seek=2G
losetup /dev/loop2 cinder-volumes
fdisk /dev/loop2
#Type in the followings:
n
p
1
ENTER
ENTER
t
8e
write
```

Proceed to create the physical volume then the volume group:

```
pvcreate /dev/loop2
vgcreate cinder-volumes /dev/loop2
```

# 10 Horizon

To install horizon, install these packages:

```
apt-get install openstack-dashboard memcached
```

I had some issues with the OpenStack ubuntu theme so i disabled it to go back to the default look:

```
nano /etc/openstack-dashboard/localsettings.py
#Comment these lines
#Enable the Ubuntu theme if it is present.
#try:
# from ubuntu_theme import *
#except ImportError:
# pass
```

Edit */etc/apache2/apache2.conf* to add this line

```
ServerName localhost
```

Reload Apache and memcached:

```
service apache2 restart; service memcached restart
```

You can now access your OpenStack **@157.159.100.232/horizon** with credentials **admin:admin_pass**.

## 11 Your first VM

To start your first VM, you will need to create networks for it. This is easy using the new Quantum project but we first need to create a new tenant as it is not recommended to play with the admin tenant.

Create a new tenant:

```
keystone tenant−create −−name project_one
```

Create a new user and assign the admin role to it in the new tenant:

```
keystone user−create −−name=user_one −−pass=user_one −−tenant−id $put_id_of_project_one −−email=user
keystone user−role−add −−tenant−id $put_id_of_project_one −−user−id $put_id_of_user_one −−role−id $put
```

Create a new network for the tenant:

```
quantum net−create −−tenant−id $put_id_of_project_one net_proj_one −−provider:network_type vlan −−provic
```

Create a new subnet inside the new tenant network:

```
quantum subnet−create −−tenant−id $put_id_of_project_one net_proj_one 10.10.10.0/24
```

* Create a router for the new tenant:

```
quantum router−create −−tenant_id $put_id_of_project_one router_proj_one
```

Add the router to the subnet:

```
quantum router−interface−add $put_router_id_here $put_subnet_id_here
```

You can now start creating VMs but they will not be accessible from the internet. If you like them to be so, perform the following:

Create your external network with the tenant id belonging to the service tenant:

```
quantum net−create ext_net −−tenant−id $SERVICE_TENANT_ID −−router:external=True
```

Create a subnet containing your floating IPs:

```
quantum subnet−create ext_net 192.168.100.10/28 −− −−enable_dhcp=False
```

Set the router for the external network:

```
quantum router−gateway−set $ROUTER_ID $EXT_NET_ID
```

**This is it !**, You can now login to your OpenStack dashboard and start creating internet accessible VMs.

I Hope you enjoyed this guide, please if you have any feedbacks, don't hesitate.

## 12 Adding a compute node

This part is comming soon (Testing Stage)

## 13 Licensing

## 14 References

- `http://docs.openstack.org/trunk/openstack-compute/install/apt/content/`
- `https://github.com/EmilienM/openstack-folsom-guide`
- `http://docs.openstack.org/trunk/openstack-network/admin/content/ch_install.html`