

Install OpenStack Folsom

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

October 19, 2012

Contents

1	Initialize	3
2	Prepare for networking	3
3	Keystone	4
4	Glance	5
5	KVM	6
6	OpenVSwitch	7
7	Quantum	7
8	Nova	8
9	Cinder	10
10	Horizon	11
11	Your first VM	12
12	Adding a compute node	12
13	Licensing	13
14	References	13

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

```
1 sudo su
```

Add the OpenStack Folsom repositories to your ubuntu repositories:

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

Update your system:

```
1 apt-get update
2 apt-get upgrade
3 apt-get dist-upgrade
```

Install the mySQL along with other stuffs:

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

Configure mySQL to receive all incoming requests:

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

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

```
1 nano /etc/ntp.conf
2
3 #Replace server ntp.ubuntu.com with :
4
5 ntp.ubuntu.com iburst
6 nserver 127.127.1.0
7 nfudge 127.127.1.0 stratum 10
8
9 #Restart the service
10 service ntp restart
```

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

```
1 nano /etc/sysctl.conf
2 #Uncomment net.ipv4.ip_forward=1
```

2 Prepare for networking

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

```
1 auto eth0
2 iface eth0 inet static
3 address 157.159.100.232
4 netmask 255.255.255.0
```

```
5 gateway 157.159.100.1
6
7 auto eth1
8 iface eth1 inet static
9 address 10.0.0.3
```

3 Keystone

This is how we install OpenStack's identity service:

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

remove the default database:

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

Create a new MySQL database for keystone:

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

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

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

Restart the identity service then synchronize the database:

```
1 service keystone restart
2 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:

```
1 chmod +x keystone_basic.sh
2 chmod +x keystone_endpoints_basic.sh
3 ./keystone_basic.sh
4 ./keystone_endpoints_basic.sh
```

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

```
1 nano creds
2 #Paste the following:
3 export OS_TENANT_NAME=admin
4 export OS_USERNAME=admin
5 export OS_PASSWORD=admin_pass
6 export OS_AUTH_URL="http://157.159.100.232:5000/v2.0/"
7 # Load it:
8 source creds
```

To test Keystone, we use a simple curl request:

```
1 apt-get install curl openssl
2 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:

```
1 apt-get install glance python-glance python-glanceclient
```

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

```
1 rm /var/lib/glance/glance.sqlite
2 mysql -u root -p
3 CREATE DATABASE glance;
4 GRANT ALL ON glance.* TO 'glanceUser'@'%' IDENTIFIED BY 'glancePass';
5 quit;
```

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

```
1 [filter:authtoken]
2 paste.filter_factory = keystone.middleware.auth_token:filter_factory
3 auth_host = 157.159.100.232
4 auth_port = 35357
5 auth_protocol = http
6 admin_tenant_name = service
7 admin_user = glance
8 admin_password = service_pass
```

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

```
1 [filter:authtoken]
2 paste.filter_factory = keystone.middleware.auth_token:filter_factory
3 auth_host = 157.159.100.232
4 auth_port = 35357
5 auth_protocol = http
6 admin_tenant_name = service
7 admin_user = glance
8 admin_password = service_pass
```

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

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

and

```
1 [paste_deploy]
2 flavor = keystone
```

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

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

and

```
1 [paste_deploy]
2 flavor = keystone
```

Restart the glance-api and glance-registry services:

```
1 service glance-api restart; service glance-registry restart
```

Synchronize the glance database:

```
1 glance-manage db_sync
```

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

```
1 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

```
1 mkdir images
2 cd images
3 wget https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img
4
5 ##### The following command is splitted to two, reform before executing #####
6 glance image-create --name myFirstImage --is-public true --container-format bare
7 --disk-format qcow2 < cirros-0.3.0-x86_64-disk.img
```

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

```
1 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:

```
1 apt-get install cpu-checker
2 kvm-ok
```

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

```
1 apt-get install -y kvm libvirt-bin pm-utils
```

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

```
1 cgroup_device_acl = [
2 "/dev/null", "/dev/full", "/dev/zero",
3 "/dev/random", "/dev/urandom",
4 "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
5 "/dev/rtc", "/dev/hpet", "/dev/net/tun"
6 ]
```

Delete default virtual bridge :

```
1 virsh net-destroy default
2 virsh net-undefine default
```

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

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

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

```
1 env libvirtd_opts="--d_l"l"
```

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

```
1 libvirtd_opts="--d_l"l"
```

Restart the libvirt service to load the new values:

```
1 service libvirt-bin restart
```

6 OpenVSwitch

Install the openVSwitch:

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

Create the bridges:

```
1 ovs-vsctl add-br br-int
2 ovs-vsctl add-br br-eth1
3 ovs-vsctl add-port br-eth1 eth1
4 ovs-vsctl add-br br-ex
5 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:

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

Install the Quantum server and the Quantum OVS plugin:

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

Create a database:

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

Edit the OVS plugin configuration file:

```
1
2 nano /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini
3
4 [DATABASE]
5 sql_connection = mysql://quantumUser:quantumPass@157.159.100.232/quantum
6 [OVS]
7 tenant_network_type=vlan
8 network_vlan_ranges = physnet1:1:4094
9 bridge_mappings = physnet1:br-eth1
```

Restart the quantum server:

```
1 service quantum-server restart
```

Install the OVS plugin agent

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

Intall quantum DHCP and l3 agents:

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

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

```
1 [filter:authtoken]
2 paste.filter_factory = keystone.middleware.auth_token:filter_factory
3 auth_host = 157.159.100.232
4 auth_port = 35357
5 auth_protocol = http
6 admin_tenant_name = service
7 admin_user = quantum
8 admin_password = service_pass
```

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

```
1 auth_url = http://157.159.100.232:35357/v2.0
2 auth_region = RegionOne
3 admin_tenant_name = service
4 admin_user = quantum
5 admin_password = service_pass
```

Restart all the services:

```
1 service quantum-server restart
2 service quantum-plugin-openvswitch-agent restart
3 service quantum-dhcp-agent restart
4 service quantum-l3-agent restart
```

8 Nova

Start by installing nova components:

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

Prepare a Mysql database for Nova:

```
1
2 rm /var/lib/nova/nova.sqlite
3 mysql -u root -p
4 CREATE DATABASE nova;
5 GRANT ALL ON nova.* TO 'novaUser'@'%' IDENTIFIED BY 'novaPass';
6 quit;
```

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

```
1 [filter:authtoken]
2 paste.filter_factory = keystone.middleware.auth_token:filter_factory
3 auth_host = 157.159.100.232
4 auth_port = 35357
5 auth_protocol = http
```



```
6admin_tenant_name = service
7admin_user = nova
8admin_password = service_pass
9signing_dirname = /tmp/keystone—signing—nova
```

Modify the */etc/nova/nova.conf* like this:

```
1[DEFAULT]
2logdir=/var/log/nova
3state_path=/var/lib/nova
4lock_path=/run/lock/nova
5verbose=True
6api_paste_config=/etc/nova/api—paste.ini
7scheduler_driver=nova.scheduler.simple.SimpleScheduler
8s3_host=157.159.100.232
9ec2_host=157.159.100.232
10ec2_dmz_host=157.159.100.232
11rabbit_host=157.159.100.232
12cc_host=157.159.100.232
13nova_url=http://157.159.100.232:8774/v1.1/
14sql_connection=mysql://novaUser:novaPass@157.159.100.232/nova
15ec2_url=http://157.159.100.232:8773/services/Cloud
16root_helper=sudo nova—rootwrap /etc/nova/rootwrap.conf
17
18# Auth
19use_deprecated_auth=false
20auth_strategy=keystone
21keystone_ec2_url=http://157.159.100.232:5000/v2.0/ec2tokens
22# Imaging service
23glance_api_servers=157.159.100.232:9292
24image_service=nova.image.glance.GlanceImageService
25
26
27# Vnc configuration
28novnc_enabled=true
29novncproxy_base_url=http://157.159.100.232:6080/vnc_auto.html
30novncproxy_port=6080
31vncserver_proxycient_address=127.0.0.1
32vncserver_listen=0.0.0.0
33
34# Network settings
35network_api_class=nova.network.quantumv2.api.API
36quantum_url=http://157.159.100.232:9696
37quantum_auth_strategy=keystone
38quantum_admin_tenant_name=service
39quantum_admin_username=quantum
40quantum_admin_password=service_pass
41quantum_admin_auth_url=http://157.159.100.232:35357/v2.0
42libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
43linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
44firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
45
46# Compute #
47compute_driver=libvirt.LibvirtDriver
```

```

48
49# Cinder #
50volume_api_class=nova.volume.cinder.API
51osapi_volume_listen_port=5900

```

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

```

1  chown -R nova. /etc/nova
2  chmod 644 /etc/nova/nova.conf

```

Add this line to the sudoers file:

```

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

```

Synchronize your database:

```

1  nova-manage db sync

```

Restart nova-* services and start the iscsi service:

```

1  cd /etc/init.d/; for i in $(ls nova-* ); do sudo service $i restart; done
2  service novnc restart

```

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

```

1  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 separated from the nova install process.

```

1  apt-get install cinder-api cinder-scheduler cinder-volume iscsitarget
2  apt-get install open-iscsi iscsitarget-dkms

```

Prepare a Mysql database for Cinder:

```

1  rm /var/lib/cinder/cinder.sqlite
2  mysql -u root -p
3  CREATE DATABASE cinder;
4  GRANT ALL ON cinder.* TO 'cinderUser'@'%' IDENTIFIED BY 'cinderPass';
5  quit;

```

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

```

1 [filter:authtoken]
2 paste.filter_factory = keystone.middleware.auth_token:filter_factory
3 service_protocol = http
4 service_host = 157.159.100.232
5 service_port = 5000
6 auth_host = 157.159.100.232
7 auth_port = 35357
8 auth_protocol = http
9 admin_tenant_name = service
10 admin_user = cinder
11 admin_password = service_pass

```

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

```
1 [DEFAULT]
2 rootwrap_config=/etc/cinder/rootwrap.conf
3 sql_connection = mysql://cinderUser:cinderPass@157.159.100.232/cinder
4 api_paste_conf = /etc/cinder/api-paste.ini
5 iscsi_helper=ietadm
6 volume_name_template = volume-%s
7 volume_group = cinder-volumes
8 verbose = True
9 auth_strategy = keystone
10 #osapi_volume_listen_port=5900
```

Finally, synchronize your database

```
1 cinder-manage db sync
```

Don't forget to create a volume group and name it `cinder-volumes`. Beware, this volume group gets lost after a system reboot.

```
1 dd if=/dev/zero of=cinder-volumes bs=1 count=0 seek=2G
2 losetup /dev/loop2 cinder-volumes
3 fdisk /dev/loop2
4 #Type in the followings:
5 n
6 p
7 1
8 ENTER
9 ENTER
10 t
11 8e
12 write
```

Proceed to create the physical volume then the volume group:

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

restart your services:

```
1 service cinder-volume restart
2 service cinder-api restart
```

10 Horizon

To install horizon, install these packages:

```
1 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:

```
1 nano /etc/openstack-dashboard/local_settings.py
2 #Comment these lines
3 #Enable the Ubuntu theme if it is present.
4 #try:
5 # from ubuntu_theme import *
6 #except ImportError:
7 # pass
```

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

```
1 ServerName localhost
```

Reload Apache and memcached:

```
1 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:

```
1 keystone tenant-create --name project_one
```

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

```
1 keystone user-create --name=user_one --pass=user_one --tenant-id $put_id_of_project_one --email=user_one
2 keystone user-role-add --tenant-id $put_id_of_project_one --user-id $put_id_of_user_one --role-id $put_id_of_admin_role
```

Create a new network for the tenant:

```
1 quantum net-create --tenant-id $put_id_of_project_one net_proj_one --provider:network_type vlan --provider:physical_network phys_net
```

Create a new subnet inside the new tenant network:

```
1 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:

```
1 quantum router-create --tenant-id $put_id_of_project_one router_proj_one
```

Add the router to the subnet:

```
1 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:

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

Create a subnet containing your floating IPs:

```
1 quantum subnet-create ext_net 192.168.100.10/28 --enable_dhcp=False
```

Set the router for the external network:

```
1 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 coming soon (Testing Stage)

13 Licensing

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit:

- <http://creativecommons.org/licenses/by-sa/3.0/>
- <https://github.com/mseknibilel/OpenStack-Folsom-Install-guide/blob/master/licence.png>

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