OpenStack Folsom Guide

Guide for Ubuntu Precise

Emilien Macchi

Table of Contents

Introduction

I'm writing this document a few weeks before Folsom stable release. I could not resist to share my experience with the community.

This document helps anyone who wants to deploy Folsom of OpenStack for development purpose.

Table 1. Architecture and informations

	controller	**compute**
Managment Network	192.168.0.1/24	192.168.0.2/24
Hostname	folsom-controller	folsom-compute
Services	MySQL, RabbitMQ, Nova, Cinder, Glance, Keystone, Quantum, Open-vSwitch	nova-compute, KVM, nova-api, Quantum Agent with Open-vSwitch

Note

That's a basic architecture, of course **you can add many compute nodes as you want**.

Since Folsom code has not been release into stable Ubuntu Packages, we are going to use "Folsom Testing Packages [https://launchpad.net/~openstack-ubuntu-testing/+archive/folsom-trunk-testing]" which are built from master for each component.

Requirements

You need at least two machines (virtual or physical) with 3 NIC (Managment Network + VMs Traffic in tunnel mode + Public Network) for controller node and 2 NIC (Managment Network + VMs Traffic in tunnel mode) for compute node. You need also to download Ubuntu 12.04 (LTS).

Note

Run all commands as the root user

Controller Node

Operating System

1. Install Ubuntu with this parameters :

   - Time zone : **UTC**
   - Hostname : **folsom-controller**
   - Packages : **OpenSSH-Server**

   After OS Installation, reboot the server .

2. Add the repository and upgrade Ubuntu :

```
apt-get install -y python-software-properties
add-apt-repository ppa:openstack-ubuntu-testing/folsom-trunk-testing
add-apt-repository ppa:openstack-ubuntu-testing/folsom-deps-staging
apt-get update && apt-get -y dist-upgrade
```

   Reboot the server.

3. Configure the network :

   - Edit **/etc/network/interfaces** file :

```
# Management Network
auto eth0
    iface eth0 inet static
    address 192.168.0.1
    netmask 255.255.255.0
    gateway 192.168.0.254
    dns-nameservers 8.8.8.8

# VMs Networks with OVS in tunnel mode
auto eth1
    iface eth1 inet static
    address 10.0.0.3
    netmask 255.255.255.0

# Public Bridge
auto eth2
    iface eth2 inet manual
    up ifconfig $IFACE 0.0.0.0 up
    up ip link set $IFACE promisc on
    down ip link set $IFACE promisc off
```

```
    down ifconfig $IFACE down
```

Then, restart network service :

```
service networking restart
```

- Enable **IP forwarding** since this node will be a gateway between external & internal network :

```
sed -i -r 's/^\s*#(net\.ipv4\.ip_forward=1.*)/\1/' /etc/sysctl.conf
echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Edit the **/etc/hosts** file and add **folsom-controller** & **folsom-compute** hostnames with correct IP.

4. Install Configure NTP :

   - Install the package :

     ```
     apt-get install -y ntp
     ```

   - Configure **/etc/ntp.conf** file :

     ```
     server ntp.ubuntu.com iburst
     server 127.127.1.0
     fudge 127.127.1.0 stratum 10
     ```

   - Restart the service :

     ```
     service ntp restart
     ```

MySQL

1. Install the packages :

   ```
   apt-get -y install mysql-server python-mysqldb
   ```

2. Allow connection from the network :

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

3. Restart the service :

   ```
   service mysql restart
   ```

4. Create Databases, Users, Rights :

   ```
   mysql -u root -ppassword <<EOF
   CREATE DATABASE nova;
   GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
       IDENTIFIED BY 'password';
   GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'192.168.0.1' \
       IDENTIFIED BY 'password';
   GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'192.168.0.2' \
       IDENTIFIED BY 'password';
   CREATE DATABASE cinder;
   GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
       IDENTIFIED BY 'password';
   CREATE DATABASE glance;
   GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
       IDENTIFIED BY 'password';
   CREATE DATABASE keystone;
   ```

```
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
    IDENTIFIED BY 'password';
CREATE DATABASE quantum;
GRANT ALL PRIVILEGES ON quantum.* TO 'quantum'@'localhost' \
    IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON quantum.* TO 'quantum'@'192.168.0.2' \
    IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
EOF
```

RabbitMQ

1. Install the packages :

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

2. Change the default password :

```
rabbitmqctl change_password guest password
```

Keystone

1. Install the packages :

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

2. Edit **/etc/keystone/keystone.conf** :

```
[DEFAULT]
admin_token = password
bind_host = 0.0.0.0
public_port = 5000
admin_port = 35357
compute_port = 8774
verbose = True
debug = True
log_file = keystone.log
log_dir = /var/log/keystone
log_config = /etc/keystone/logging.conf

[sql]
connection = mysql://keystone:password@localhost:3306/keystone
idle_timeout = 200

[identity]
driver = keystone.identity.backends.sql.Identity

[catalog]
driver = keystone.catalog.backends.sql.Catalog

(...)
```

3. Restart Keystone and create the tables in the database :

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

4. Load environment variables :

- Create **novarc** file :

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL="http://localhost:5000/v2.0/"
export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
export SERVICE_TOKEN=password
```

- Export the variables :

```
source novarc
echo "source novarc">>.bashrc
```

5. Download the data script [https://github.com/EmilienM/openstack-folsom-guide/blob/master/scripts/keystone-data.sh] and fill Keystone database with datas :

```
./keystone-data.sh
```

6. Download the endpoint script [https://github.com/EmilienM/openstack-folsom-guide/blob/master/scripts/keystone-endpoints.sh] and create the endpoints :

```
./keystone-endpoints.sh
```

Glance

1. Install the packages :

```
apt-get -y install glance glance-api python-glanceclient glance-common
```

2. Configure Glance :

- Edit **/etc/glance/glance-api.conf** and **/etc/glance/glance-registry.conf** files and modify :

```
sql_connection = mysql://glance:password@localhost/glance
admin_tenant_name = service
admin_user = glance
admin_password = password
```

For **glance-api.conf**, modify :

```
notifier_strategy = rabbit
rabbit_password = password
```

- Restart Glance services :

```
service glance-api restart && service glance-registry restart
```

- Create Glance tables into the database :

```
glance-manage db_sync
```

- Download and import Ubuntu 12.04 UEC Image [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
glance image-create name="Ubuntu" is_public=true container_format=ovf \
    disk_format=qcow2 < precise-server-cloudimg-amd64.img
```

- Check if the image has been introduced in the index :

```
glance image-list

+--------+---------+-------------+------------------+------------+--------+
| ID     | Name    | Disk Format | Container Format | Size       | Status |
+--------+---------+-------------+------------------+------------+--------+
| 9a17961| Ubuntu  | qcow2       | ovf              | 1476395008 | active |
+--------+---------+-------------+------------------+------------+--------+
```

- If you want to install Glance Replicator (new in Folsom) :

  ```
  https://review.openstack.org/#/c/7615/
  ```

  More informations about it here [http://www.stillhq.com/openstack/000007.html].

Nova

1. Install the packages :

```
apt-get -y install nova-api nova-cert nova-common \
    nova-scheduler python-nova python-novaclient nova-consoleauth novnc
```

2. Configure Nova :

   - Edit **/etc/nova/api-paste.ini** file and modify :

     ```
     admin_tenant_name = service
     admin_user = nova
     admin_password = password
     ```

     You should also **delete** each composite with "**volume**".

     We can do that manually or with this command :

     ```
     sed -i '/volume/d' /etc/nova/api-paste.ini
     ```

   - Edit **/etc/nova/nova.conf** file and modify :

     ```
     [DEFAULT]

     # MySQL Connection #
     sql_connection=mysql://nova:password@192.168.0.1/nova

     # nova-scheduler #
     rabbit_password=password
     scheduler_driver=nova.scheduler.simple.SimpleScheduler

     # nova-api #
     cc_host=192.168.0.1
     auth_strategy=keystone
     s3_host=192.168.0.1
     ec2_host=192.168.0.1
     nova_url=http://192.168.0.1:8774/v1.1/
     ec2_url=http://192.168.0.1:8773/services/Cloud
     keystone_ec2_url=http://192.168.0.1:5000/v2.0/ec2tokens
     api_paste_config=/etc/nova/api-paste.ini
     allow_admin_api=true
     use_deprecated_auth=false
     ec2_private_dns_show_ip=True
     dmz_cidr=169.254.169.254/32
     ```

```
ec2_dmz_host=192.168.0.1
metadata_host=192.168.0.1
metadata_listen=0.0.0.0
enabled_apis=ec2,osapi_compute,metadata

# Networking #
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.168.0.1:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=password
quantum_admin_auth_url=http://192.168.0.1:35357/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

# Cinder #
volume_api_class=nova.volume.cinder.API

# Glance #
glance_api_servers=192.168.0.1:9292
image_service=nova.image.glance.GlanceImageService

# novnc #
novnc_enable=true
novncproxy_base_url=http://192.168.0.1:6080/vnc_auto.html
vncserver_proxyclient_address=127.0.0.1
vncserver_listen=0.0.0.0

# Misc #
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
verbose=true
```

- Create Nova tables into the database :

```
nova-manage db sync
```

- Restart Nova services :

```
service nova-api restart
service nova-cert restart
service nova-consoleauth restart
service nova-scheduler restart
service novnc restart
```

Open-vSwitch

1. Install the packages :

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

2. Start Open-vSwitch service & restart the agent :

```
/etc/init.d/openvswitch-switch start
restart quantum-plugin-openvswitch
```

3. Configure virtual bridging :

```
ovs-vsctl add-br br-int
ovs-vsctl add-br br-ex
ovs-vsctl br-set-external-id br-ex bridge-id br-ex
ovs-vsctl add-port br-ex eth2
```

Quantum

1. Install the packages :

```
apt-get -y install quantum-server python-cliff \
    quantum-plugin-openvswitch-agent \
    quantum-l3-agent quantum-dhcp-agent \
    python-pyparsing
```

2. Configure Quantum services :

   - Edit **/etc/quantum/quantum.conf** file and modify :

```
core_plugin = \
    quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
auth_strategy = keystone
fake_rabbit = False
rabbit_password = password
```

   - Edit **/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini** file and modify :

```
[DATABASE]
sql_connection = mysql://quantum:password@localhost:3306/quantum
reconnect_interval = 2
[OVS]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
integration_bridge = br-int
tunnel_bridge = br-tun
local_ip = 10.0.0.3
enable_tunneling = True
[AGENT]
root_helper = sudo /usr/bin/quantum-rootwrap /etc/quantum/rootwrap.conf
```

Note

It's more handy to choose **tunnel mode** since you don't have to configure your physical switchs for VLANs.

   - Edit **/etc/quantum/l3_agent.ini** file and modify :

```
[DEFAULT]
debug = True
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
auth_url = http://localhost:35357/v2.0
auth_region = RegionOne
admin_tenant_name = service
admin_user = quantum
admin_password = password
root_helper = sudo quantum-rootwrap /etc/quantum/rootwrap.conf
metadata_ip = 192.168.0.1
use_namespaces = False
```

- Edit **/etc/quantum/dhcp_agent.ini** file and add :

```
use_namespaces = False
```

- Edit **/etc/quantum/api-paste.ini** file and modify :

```
admin_tenant_name = service
admin_user = quantum
admin_password = password
```

3. Start the services :

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

4. Download my Quantum script [https://github.com/EmilienM/openstack-folsom-guide/blob/master/scripts/quantum-networking.sh]. Before launching it, you should modify networking informations inside the script. All is commented and you can customize belong your needs. In this script, we actually create one tenant network with its router, and one external network connected to the tenant router. We are using the "**Per-tenant Routers with Private Networks**" use-case.

```
./quantum-networking.sh
```

Cinder

1. Install the packages :

```
apt-get install -y cinder-api cinder-scheduler cinder-volume iscsitarget \
    open-iscsi iscsitarget-dkms python-cinderclient
```

2. Configure & start the iSCSI services :

```
sed -i 's/false/true/g' /etc/default/iscsitarget
service iscsitarget start
service open-iscsi start
```

3. Configure Cinder :

- Edit **/etc/cinder/cinder.conf** file and modify :

```
[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
sql_connection = mysql://cinder:password@localhost:3306/cinder
iscsi_helper = ietadm
volume_group = cinder-volumes
rabbit_password = password
logdir = /var/log/cinder
verbose = true
auth_strategy = keystone
```

- Edit **/etc/cinder/api-paste.ini** file and modify :

```
admin_tenant_name = service
admin_user = cinder
admin_password = password
```

- Create the volume :

```
fdisk /dev/sdb

[Create a Linux partition]

pvcreate /dev/sdb1
vgcreate cinder-volumes /dev/sdb1
```

- Create Cinder tables into the database :

```
cinder-manage db sync
```

- Restart the services : :

```
service cinder-api restart
service cinder-scheduler restart
service cinder-volume restart
```

Horizon

Install the packages :

```
apt-get -y install apache2 libapache2-mod-wsgi openstack-dashboard \
    memcached python-memcache
```

You can now login with **admin** / **password** credentials or **demo** / **password**.

Compute Node

Operating System

1. Install Ubuntu with this parameters :
   - Time zone : **UTC**
   - Hostname : **folsom-compute**
   - Packages : **OpenSSH-Server**

   After OS Installation, reboot the server .
2. Add the repository and upgrade Ubuntu :

   ```
   apt-get install -y python-software-properties
   add-apt-repository ppa:openstack-ubuntu-testing/folsom-trunk-testing
   add-apt-repository ppa:openstack-ubuntu-testing/folsom-deps-staging
   apt-get update && apt-get -y dist-upgrade
   ```

   Reboot the server.
3. Configure the network :
   - Edit **/etc/network/interfaces** file :

   ```
   # Management Network
   auto eth0
       iface eth0 inet static
       address 192.168.0.2
       netmask 255.255.255.0
   ```

```
    gateway 192.168.0.254
    dns-nameservers 8.8.8.8

# VMs Networks with OVS in tunnel mode
auto eth1
    iface eth1 inet static
    address 10.0.0.4
    netmask 255.255.255.0
```

Then, restart network service :

```
service networking restart
```

Note

If **eth1** is connected to a Switch, it should be in tagged mode.

- Enable **IP forwarding** :

```
sed -i -r 's/^\s*#(net\.ipv4\.ip_forward=1.*)/\1/' /etc/sysctl.conf
echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Edit the **/etc/hosts** file and add **folsom-controller** & **folsom-compute** hostnames with correct IP.

4. Install & Configure NTP :

- Install the package :

```
apt-get install -y ntp
```

- Configure **/etc/ntp.conf** file :

```
server 192.168.0.1
```

- Restart the service :

```
service ntp restart
```

Hypervisor

1. Install the packages that we need :

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

2. Configure libvirt :

- Edit **/etc/libvirt/qemu.conf** file and add :

```
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",
]
```

- Disable **KVM default virtual bridge** to avoid any confusion :

```
virsh net-destroy default
virsh net-undefine default
```

- Allow **Live Migrations** :

Edit **/etc/libvirt/libvirtd.conf** file :

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

Modify 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"
```

3. • Restart libvirt :

```
service libvirt-bin restart
```

Nova

1. Install the packages :

```
apt-get -y install nova-api-metadata nova-compute-kvm novnc
```

2. Configure Nova :

   • Edit **/etc/nova/api-paste.ini** file and modify :

   ```
   admin_tenant_name = service
   admin_user = nova
   admin_password = password
   ```

   • Edit **/etc/nova/nova-compute.conf** file and modify :

   ```
   [DEFAULT]
   libvirt_type=kvm
   libvirt_ovs_bridge=br-int
   libvirt_vif_type=ethernet
   libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
   libvirt_use_virtio_for_bridges=True
   ```

   • Edit **/etc/nova/nova.conf** file and modify :

   ```
   [DEFAULT]

   # MySQL Connection #
   sql_connection=mysql://nova:password@192.168.0.1/nova

   # nova-scheduler #
   rabbit_host=192.168.0.1
   rabbit_password=password
   scheduler_driver=nova.scheduler.simple.SimpleScheduler

   # nova-api #
   cc_host=192.168.0.1
   auth_strategy=keystone
   s3_host=192.168.0.1
   ec2_host=192.168.0.1
   nova_url=http://192.168.0.1:8774/v1.1/
   ec2_url=http://192.168.0.1:8773/services/Cloud
   ```

```
keystone_ec2_url=http://192.168.0.1:5000/v2.0/ec2tokens
api_paste_config=/etc/nova/api-paste.ini
allow_admin_api=true
use_deprecated_auth=false
ec2_private_dns_show_ip=True
dmz_cidr=169.254.169.254/32
ec2_dmz_host=192.168.0.1
metadata_host=192.168.0.2
metadata_listen=0.0.0.0
enabled_apis=metadata

# Networking #
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.168.0.1:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=password
quantum_admin_auth_url=http://192.168.0.1:35357/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

# Glance #
glance_api_servers=192.168.0.1:9292
image_service=nova.image.glance.GlanceImageService

# novnc #
novnc_enable=true
novncproxy_base_url=http://192.168.0.2:6080/vnc_auto.html
vncserver_proxyclient_address=127.0.0.1
vncserver_listen=0.0.0.0

# Misc #
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
verbose=true
```

- Restart Nova services :

```
service nova-api-metadata restart
service nova-compute restart
```

Open-vSwitch

1. Install the packages :

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

2. Edit **/etc/quantum/quantum.conf** file and modify :

```
core_plugin = \
    quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
auth_strategy = keystone
fake_rabbit = False
rabbit_host = 192.168.0.1
rabbit_password = password
```

3. Start Open-vSwitch service & restart the agent :

```
/etc/init.d/openvswitch-switch start
restart quantum-plugin-openvswitch-agent
```

4. Configure virtual bridging :

```
ovs-vsctl add-br br-int
```

5. Edit **/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini** file and modify :

```
[DATABASE]
sql_connection = mysql://quantum:password@192.168.0.1:3306/quantum
reconnect_interval = 2
[OVS]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
integration_bridge = br-int
tunnel_bridge = br-tun
local_ip = 10.0.0.4
enable_tunneling = True
[AGENT]
root_helper = sudo /usr/bin/quantum-rootwrap /etc/quantum/rootwrap.conf
```

6. Start the Agent :

```
service quantum-plugin-openvswitch-agent restart
```

Create your first VM

This section is going to be written very soon.

Credits

Thank's to ...

John Griffith - SolidFire

Martin Loschwitz - Hastexo

Adam Gandelman - Canonical

Dan Wendlandt - Nicira / VMware

License

```
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

About the author

Emilien Macchi, Junior System Engineer at StackOps Technologies [http://www.stackops.com/].