OpenStack Installation

(Ubuntu 16.04 TLS)

RegionOne

---

1. บทนำ

เอกสารฉบับนี้เป็นเอกสารแนะนำขั้นตอนการติดตั้ง ระบบคลาวด์ (Cloud) ด้วย OpenStack (http://www.openstack.org) โดยมีเนื้อหาเฉพาะส่วนของขั้นตอนการดำเนิน ไม่มีเนื้อหาเกี่ยวกับทฤษฎีหรือหลักการเลย :) ขั้นตอนในเอกสารนั้น ผ่านการทดลองด้วยระบบจำลองบนโปรแกรม Virtual โดยพยายามกำหนดคุณสมบัติของเครื่องให้เป็นไปตามความต้องการพื้นฐานของ OpenStack แต่ก็จะมีบางส่วนที่ VirtualBox ไม่สามารถทำงานได้เสมือนกับเครื่องจริง

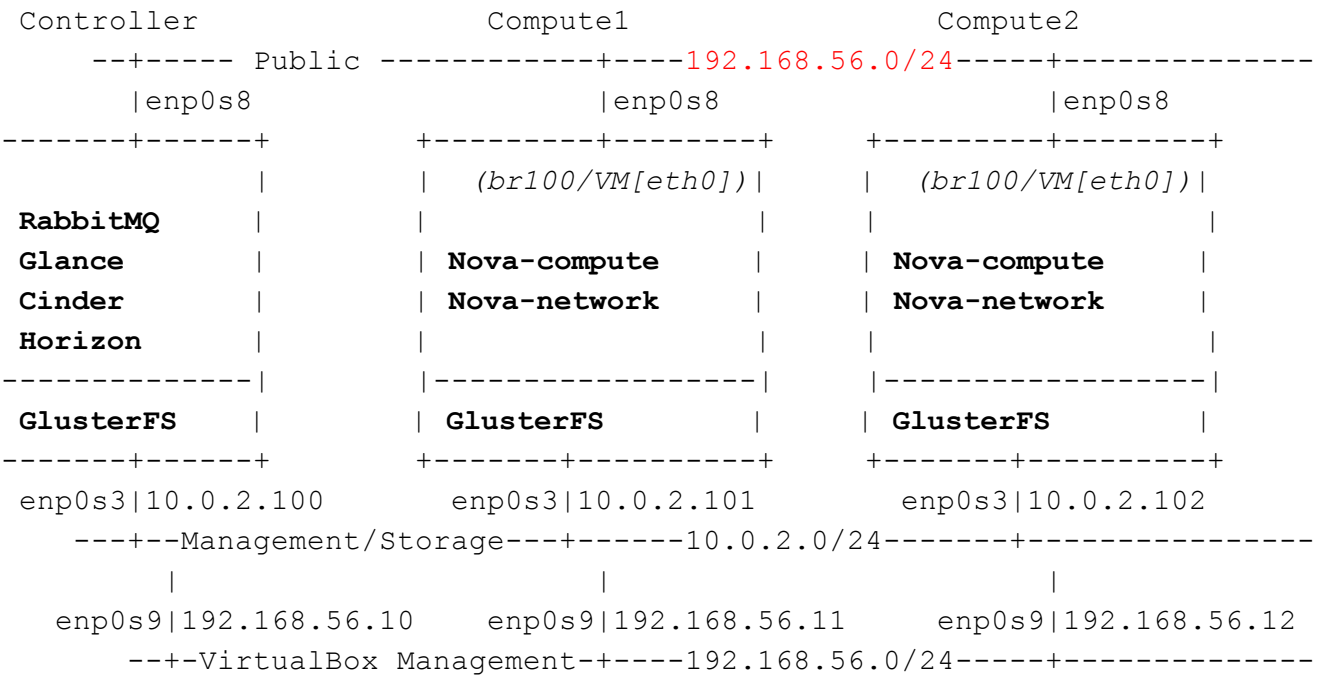
สภาพแวดล้อมของการทดลองคือ เป็นการติดตั้ง OpenStack โดยใช้ระบบปฏิบัติการ Ubuntu 16.04 TLS ดำเนินการเทียบกับขั้นตอนการติดตั้งจากเว็บ https://docs.openstack.org/ocata/install-guide-ubuntu และมีเนื้อหาเสริมบางส่วนจากการนำ GlusterFS มาติดตั้งใช้งานร่วมกับ OpenStack

1.1. คุณสมบัติฐานข้อมูล ผู้ใช้และรหัสผ่าน

คุณสมบัติทั้งหมดต่อไปนี้ ใช้สำหรับการตั้งค่าและกำหนดเป็นรหัสผ่านของการทำงานของแต่ละส่วนของ OpenStack

```
MySQL database system user: root
MySQL database system password: openstack;pass

RABBIT_USER: openstack
RABBIT_PASS: rabbit;pass

KEYSTONE_DBUSER, KEYSTONE_USER: keystone
KEYSTONE_DBPASS, KEYSTONE_PASS: keystone;pass

GLANCE_DBUSER, GLANCE_USER: glance
GLANCE_DBPASS, GLANCE_PASS: glance;pass

NOVA_DBUSER, NOVA_USER: nova
NOVA_DBPASS, NOVA_PASS: nova;pass

NEUTRON_DBUSER, NEUTRON_USER: neutron
NEUTRON_DBPASS, NEUTRON_PASS: neutron;pass

CINDER_DBUSER, CINDER_USER: cinder
CINDER_DBPASS, CINDER_PASS: cinder;pass
```

1.2. ผังการเชื่อมต่อเครือข่าย



---

2. Network Configuration

Controller & Compute Node

Install packages

```
_# apt-get install bridge-utils
```

Controller

Edit network configuration file /etc/network/interfaces

```
# Management/Storage network interface
auto enp0s3
iface enp0s3 inet static
      address 10.0.2.100
      netmask 255.255.255.0

# Public/Provider network interface
auto enp0s8
iface enp0s8 inet manual
      up ip link set dev $IFACE up
      down ip link set dev $IFACE down
```

Restart networking

```
controller# service networking restart
```

Compute Node

Edit network configuration file /etc/network/interfaces

```
# Management/Storage network interface
auto enp0s3
iface enp0s3 inet static
      address 10.0.2.101 (...)
      netmask 255.255.255.0

# Public/Provider network interface
auto enp0s8
iface enp0s8 inet manual
      up ip link set dev $IFACE up
      down ip link set dev $IFACE down
```

Restart networking

```
computeX# service networking restart
```

Controller & Compute Node

Edit file /etc/hosts

```
# ...
10.0.2.100  controller
10.0.2.101  compute1
10.0.2.102  compute2

10.0.2.100   block0
10.0.2.101   block1
10.0.2.102   block2
```

---

3. Base Services

Controller & Compute Node

Manually install ssh service

```
_# apt install ssh
```

Controller

Install NTP package

```
controller# apt install ntp
```

Edit NTP configuration file /etc/ntp.conf (Optional)

```
#...
server 203.158.192.11 iburst
restrict 10.0.2.0 netmask 255.255.255.0 noburst
```

Restart NTP service

```
controller# service ntp restart
```

Compute Node

Install NTP package

```
computeX# apt install ntp
```

Edit NTP configuration file /etc/ntp.conf (Optional)

```
# ...
server 10.0.2.100 iburst
server 203.158.192.11 iburst
```

Restart NTP service

```
computeX# service ntp restart
```

---

4. OpenStack Packages

Controller & Compute Node

Enable the OpenStack repository

```
_# apt install software-properties-common
_# add-apt-repository cloud-archive:ocata
```

Upgrade the packages on your host

```
_# apt update && apt dist-upgrade
```

Install the OpenStack client

```
_# apt install python-openstackclient
```

4.1. Database - MySQL

Controller

Install MySQL packages

```
controller# apt install mariadb-server python-pymysql
```

Edit MySQL configuration file /etc/mysql/mariadb.conf.d/99-openstack.cnf

```
[mysqld]

bind-address = 10.0.2.100

default-storage-engine = innodb
innodb_file_per_table
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

Restart MySQL service

```
controller# service mysql restart
```

Re-initialize MySQL security (Optional)

```
controller# mysql_secure_installation
Enter current password for root (enter for none):

Set root password? [Y/n] y
New password: <MySQL root password>
Re-enter new password: <MySQL root password>

Remove anonymous users? [Y/n] y

Disallow root login remotely? [Y/n] n

Remove test database and access to it? [Y/n] y

Reload privilege tables now? [Y/n] y
```

Note: *MySQL root password* is: *openstack;pass*

4.2. Messaging server - RabbitMQ

Controller

Install RabbitMQ message broker service

```
controller# apt install rabbitmq-server
```

Configure the message broker service

```
controller# rabbitmqctl add_user openstack '<RABBIT_PASS>'
```

Note: *<RABBIT_PASS>* is: *rabbit;pass*

Permit configuration, write, and read access for the openstack user

```
controller# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Restart the message broker service

```
controller# service rabbitmq-server restart
```

4.3. Caching service - Memcached

Controller

Install memcached service

```
controller# apt-get install memcached python-memcache
```

Edit memcached configuration file /etc/memcached.conf

```
# ...
-l 127.0.0.1,controller
```

Restart the message broker service

```
controller# service memcached restart
```

---

5. Add Identity service

5.1. Create Identity service database

Controller

Connect to MySQL database

```
controller# mysql -u root -p
Enter password: <MYSQL_PASSWORD>
```

Note: *<MYSQL_PASSWORD>* is: *openstack;pass*

Create keystone database

```
MariaDB [(none)]> CREATE DATABASE keystone;
```

Create and grant keystone user to keystone database

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO \
  'keystone'@'localhost' IDENTIFIED BY '<KEYSTONE_DBPASS>';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO \
  'keystone'@'%' IDENTIFIED BY '<KEYSTONE_DBPASS>';
```

Note: *<KEYSTONE_DBPASS>* is: *keystone;pass*

Apply new privileges

```
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit
```

5.2. Install and configure the components

Controller

Install packages

```
controller# apt install keystone python-keystoneclient
```

Edit keystone configuration file /etc/keystone/keystone.conf

```
[database]
# ...
connection = mysql+pymysql://keystone:<KEYSTONE_DBPASS>@controller/keystone
# ...

[token]
# ...
provider = fernet
# ...
```

Note: *<KEYSTONE_DBPASS>* is: *keystone;pass*

Populate the Identity service database

```
controller# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

Initialize Fernet key repositories

```
controller# keystone-manage fernet_setup \
    --keystone-user keystone --keystone-group keystone
controller# keystone-manage credential_setup \
    --keystone-user keystone --keystone-group keystone
```

Bootstrap the Identity service

```
controller# keystone-manage bootstrap --bootstrap-password '<ADMIN_PASS>' \
    --bootstrap-admin-url http://controller:35357/v3/ \
    --bootstrap-internal-url http://controller:5000/v3/ \
    --bootstrap-public-url http://controller:5000/v3/ \
    --bootstrap-region-id 'RegionOne'
```

Note: *<ADMIN_PASS>* is: *cloud;admin*

5.3. Create OpenStack client environment scripts for user admin and demo

Controller

Create the /root/admin-rc.sh file and add the following content

```
export OS_USERNAME=admin
export OS_PASSWORD='<ADMIN_PASS>'
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_DOMAIN_NAME=default
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
```

Note: *<ADMIN_PASS>* is: *cloud;admin*

Edit file /root/demo-rc.sh and add the following content

```
export OS_USERNAME=demo
export OS_PASSWORD='demo'
export OS_PROJECT_NAME=demo
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_DOMAIN_NAME=default
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

5.4. Create a domain, projects, users, and roles

Controller

Excute script to define environment variables

```
controller# source admin-rc.sh
```

Create the *service* project

```
controller# openstack project create --domain default \
    --description "Service Project" service
```

Create the *demo* project

```
controller# openstack project create --domain default \
    --description "Demo Project" demo
```

Create the *demo* user

```
controller# openstack user create --domain default \
  --password-prompt demo
User Password: demo
Repeat User Password: demo
```

Create the *user* role

```
controller# openstack role create user
```

Add the *user* role to the *demo* project and user

```
controller# openstack role add --project demo --user demo user
```

5.5. Verify operation

Controller

Unset the temporary OS_AUTH_URL and OS_PASSWORD environment variables

```
controller# unset OS_AUTH_URL OS_PASSWORD
```

As the *admin* user, request an authentication token

```
controller# openstack --os-auth-url http://controller:35357/v3 \
    --os-project-domain-name default --os-user-domain-name default \
    --os-project-name admin --os-username admin token issue
Password: <ADMIN_PASS>
```

Note: *<ADMIN_PASS>* is: *cloud;admin*

As the *demo* user, request an authentication token

```
controller# openstack --os-auth-url http://controller:5000/v3 \
    --os-project-domain-name default --os-user-domain-name default \
    --os-project-name demo --os-username demo token issue
Password: demo
```

6. Add the Image service

6.1. Create Image service database

Controller

Connect to MySQL database

```
controller# mysql -u root -p
Enter password: <MYSQL_PASSWORD>
```

Note: *<MYSQL_PASSWORD>* is: *openstack;pass*

Create glance database

```
MariaDB [(none)]> CREATE DATABASE glance;
```

Create and grant glance user to glance database

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO \
  'glance'@'localhost' IDENTIFIED BY '<GLANCE_DBPASS>';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO \
  'glance'@'%' IDENTIFIED BY '<GLANCE_DBPASS>';
```

Note: *<GLANCE_PASS>* is: *glance;pass*

Apply new privileges

```
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit
```

6.2. Install and configure

Controller

Source the admin credentials to gain access to admin-only CLI commands

```
controller# source admin-rc.sh
```

Create the glance user

```
controller# openstack user create --domain default \
    --password-prompt glance
User Password: <GLANCE_PASS>
Repeat User Password: <GLANCE_PASS>
```

Note: <GLANCE_PASS> is: glance;pass

Add the admin role to the glance user and service project

```
controller# openstack role add --project service --user glance admin
```

Create the glance service entity

```
controller# openstack service create --name glance \
    --description "OpenStack Image" image
```

Create the Image service API endpoints

```
controller# openstack endpoint create --region 'RegionOne' \
    image public http://controller:9292

controller# openstack endpoint create --region 'RegionOne' \
    image internal http://controller:9292

controller# openstack endpoint create --region 'RegionOne' \
    image admin http://controller:9292
```

Install the packages

```
controller# apt install glance
```

Edit file /etc/glance/glance-api.conf

```
[DEFAULT]
# ...
auth_url = http://controller:35357

[database]
# ...
connection = mysql+pymysql://glance:<GLANCE_DBPASS>@controller/glance

[keystone_authtoken]
# ...
auth_uri = http://controller:5000
memcached_servers = controller:11211
auth_type = password
```

```
auth_url = http://controller:35357
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = <GLANCE_PASS>

[paste_deploy]
# ...
flavor = keystone

[glance_store]
# ...
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Note: <GLANCE_DBPASS> is: glance;pass
<GLANCE_PASS> is: glance;pass

Edit file /etc/glance/glance-registry.conf

```
[database]
# ...
connection = mysql+pymysql://glance:<GLANCE_DBPASS>@controller/glance

[keystone_authtoken]
# ...
auth_uri = http://controller:5000
memcached_servers = controller:11211
auth_type = password

auth_url = http://controller:35357
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = <GLANCE_PASS>

[paste_deploy]
# ...
flavor = keystone
```

Note: *<GLANCE_DBPASS>* is: *glance;pass*
<GLANCE_PASS> is: *glance;pass*

Populate the Image Service database:

```
controller# su -s /bin/sh -c "glance-manage db_sync" glance
```

Restart the Image Service services

```
controller# service glance-api restart
controller# service glance-registry restart
```

6.3. Verify operation

Controller

Create and change into a temporary local directory

```
controller# mkdir -p /var/tmp/images
```

Download the image to the temporary local directory

```
controller# wget -P /var/tmp/images \
  http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img
```

Source the admin credentials to gain access to admin-only CLI commands

```
controller# source admin-rc.sh
```

Upload the image to the Image Service

```
controller# openstack image create "CirrOS" \
    --file /var/tmp/images/cirros-0.3.5-x86_64-disk.img \
    --disk-format qcow2 --container-format bare
```

List of all images

```
controller# openstack image list
```

Delete image

```
controller# openstack image delete <image_id>
```

---

7. Add the Compute service

7.1. install and configure the Compute controller components

Controller

Connect to MySQL database

```
controller# mysql -u root -p
Enter password: <MYSQL_PASSWORD>
```

Note: *<MYSQL_PASSWORD>* is: *openstack;pass*

Create nova database

```
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;
```

Create and grant nova user to nova database

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO \
  'nova'@'localhost' IDENTIFIED BY '<NOVA_DBPASS>';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO \
  'nova'@'%' IDENTIFIED BY '<NOVA_DBPASS>';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO \
  'nova'@'localhost' IDENTIFIED BY '<NOVA_DBPASS>';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO \
  'nova'@'%' IDENTIFIED BY '<NOVA_DBPASS>';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO \
  'nova'@'localhost' IDENTIFIED BY '<NOVA_DBPASS>';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO \
  'nova'@'%' IDENTIFIED BY '<NOVA_DBPASS>';
```

Note: *<NOVA_DBPASS>* is: *nova;pass*

Apply new privileges

```
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit
```

Source the admin credentials

```
controller# source admin-rc.sh
```

Create the nova user:

```
controller# openstack user create --domain default \
    --password-prompt nova
User Password: <NOVA_PASS>
Repeat User Password: <NOVA_PASS>

Note: <NOVA_PASS> is: nova;pass
```

Add the admin role to the nova user:

```
controller# openstack role add --project service --user nova admin
```

Create the nova service entity:

```
controller# openstack service create --name nova \
    --description "OpenStack Compute" compute
```

Create the Compute service API endpoints:

```
controller# openstack endpoint create --region 'RegionOne' \
    compute public http://controller:8774/v2.1

controller# openstack endpoint create --region 'RegionOne' \
    compute internal http://controller:8774/v2.1

controller# openstack endpoint create --region 'RegionOne' \
    compute admin http://controller:8774/v2.1
```

Create a Placement service user:

```
controller# openstack user create --domain default \
    --password-prompt placement
User Password: <PLACEMENT_PASS>
Repeat User Password: <PLACEMENT_PASS>

Note: <PLACEMENT_PASS> is: placement;pass
```

Add the admin role to the placement user:

```
controller# openstack role add --project service --user placement admin
```

Create the Placement API entry in the service catalog:

```
controller# openstack service create --name placement \
    --description "Placement API" placement
```

Create the Placement API service endpoints:

```
controller# openstack endpoint create --region 'RegionOne' \
    placement public http://controller:8778

controller# openstack endpoint create --region 'RegionOne' \
    placement internal http://controller:8778

controller# openstack endpoint create --region 'RegionOne' \
    placement admin http://controller:8778
```

Install the packages

```
controller# apt install nova-api nova-conductor nova-consoleauth \
    nova-novncproxy nova-scheduler nova-placement-api
```

Edit file /etc/nova/nova.conf

```
[DEFAULT]
# ...
my_ip = 10.0.2.100
enabled_apis = osapi_compute,metadata

use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver

transport_url = rabbit://openstack:<RABBIT_PASS>@controller

[api]
# ...
auth_strategy = keystone

[api_database]
# ...
connection = mysql+pymysql://nova:<NOVA_DBPASS>@controller/nova_api

[database]
connection = mysql+pymysql://nova:<NOVA_DBPASS>@controller/nova

[glance]
# ...
api_servers = http://controller:9292

[keystone_authtoken]
# ...
```

```
auth_uri = http://controller:5000
memcached_servers = controller:11211
auth_type = password

auth_url = http://controller:35357
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = <NOVA_PASS>

[placement]
# ...
os_region_name = RegionOne
auth_type = password
auth_url = http://controller:35357
project_name = service
project_domain_name = default
username = placement
user_domain_name = default
password = <PLACEMENT_PASS>

[vnc]
enabled = true
# ...
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip
```

Note: <RABBIT_PASS> is: rabbit;pass
<NOVA_DBPASS> is: nova;pass
<NOVA_PASS> is: nova;pass
<PLACEMENT_PASS> is: placement;pass

Populate the Image Service database

```
controller# su -s /bin/sh -c "nova-manage api_db sync" nova
controller# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
controller# su -s /bin/sh -c "nova-manage cell_v2 create_cell \
    --name=cell1 --verbose" nova
controller# su -s /bin/sh -c "nova-manage db sync" nova
```

```
controller# nova-manage cell_v2 list_cells
+-------+--------------------------------------+
|  Name |                 UUID                 |
+-------+--------------------------------------+
| cell0 | 00000000-0000-0000-0000-000000000000 |
| cell1 | 4477ddd7-af84-4991-9e55-7c6324367589 |
+-------+--------------------------------------+
```

Restart the Compute services

```
controller# service nova-api restart
controller# service nova-consoleauth restart
controller# service nova-scheduler restart
controller# service nova-conductor restart
controller# service nova-novncproxy restart
```

7.2. Create flavors

Controller

Source the admin credentials to gain access to admin-only CLI commands

```
controller# source admin-rc.sh
```

Create m1.nano flavor - 1 CPU, 64MB RAM, 1GB Disk

```
controller# openstack flavor create --id 0 --vcpus 1 \
    --ram 64 --disk 1 m1.nano
```

Create m1.tiny flavor - 1 CPU, 512MB RAM, 1GB Disk

```
controller# openstack flavor create --id 1 --vcpus 1 \
    --ram 512 --disk 1 m1.tiny
```

7.3. Verify operation on Controller

Controller

Source the admin credentials to gain access to admin-only CLI commands:

```
controller# source admin-rc.sh
```

List service components to verify successful launch of each process:

```
controller# openstack compute service list
```

List API endpoints in the Identity service:

```
controller# openstack catalog list
```

List images in the Image service:

```
controller# openstack image list
```

List flavors:

```
controller# openstack flavor list
```

7.4. install and configure the Compute components on compute nodes

Compute Nodes

Install the packages:

```
computeX# apt install nova-compute
```

Edit file /etc/nova/nova.conf

```
[DEFAULT]
# ...
my_ip = 10.0.2.101 (...)
enabled_apis = osapi_compute,metadata

use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver

transport_url = rabbit://openstack:<RABBIT_PASS>@controller

[api]
# ...
auth_strategy = keystone

[glance]
# ...
api_servers = http://controller:9292

[keystone_authtoken]
# ...
auth_uri = http://controller:5000
memcached_servers = controller:11211
auth_type = password

auth_url = http://controller:35357
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
```

```
password = <NOVA_PASS>

[placement]
# ...
os_region_name = RegionOne
auth_type = password
auth_url = http://controller:35357
project_name = service
project_domain_name = default
username = placement
user_domain_name = default
password = <PLACEMENT_PASS>

[vnc]
enabled = true
# ...
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

Note: <RABBIT_PASS> is: rabbit;pass
<NOVA_DBPASS> is: nova;pass
<NOVA_PASS> is: nova;pass
<PLACEMENT_PASS> is: placement;pass

Determine whether your compute node supports hardware acceleration for virtual machines

```
computeX# egrep -c '(vmx|svm)' /proc/cpuinfo
```

If result is 0, vert_type = qemu
If result is not 0, vert_type = kvm

Edit [libvirt] section in file /etc/nova/nova-compute.conf

```
[libvirt]
virt_type = qemu
```

Restart the Compute service:

```
computeX# service nova-compute restart
```

7.5. Add new compute nodes to the cell database

Controller

Source the admin credentials to gain access to admin-only CLI commands:

```
controller# source admin-rc.sh
```

Discover compute hosts:

```
controller# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts \
   --verbose" nova
```

7.6. Verify operation on Compute Node

Controller & Compute Node

Source the admin credentials to gain access to admin-only CLI commands:

```
controller# source admin-rc.sh
```

List service components to verify successful launch of each process:

```
controller# openstack compute service list
```

List hypervisor nodes:

```
controller# openstack hypervisor list
```

List images in the Image Service catalog to verify connectivity with the Identity service and Image Service:

```
controller# openstack image list
```

---

8. Add a network component

8.1. Configure network service

Controller

Connect to MySQL database

```
controller# mysql -u root -p
Enter password: <MYSQL_PASSWORD>
```

Note: *<MYSQL_PASSWORD>* is: *openstack;pass*

Create neutron database

```
MariaDB [(none)]> CREATE DATABASE neutron;
```

Create and grant neutron user to neutron database

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO \
  'neutron'@'localhost' IDENTIFIED BY '<NEUTRON_DBPASS>';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO \
  'neutron'@'%' IDENTIFIED BY '<NEUTRON_DBPASS>';
```

Note: *<NEUTRON_DBPASS>* is: *neutron;pass*

Apply new privileges

```
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit
```

Source the admin credentials

```
controller# source admin-rc.sh
```

Create the neutron user:

```
controller# openstack user create --domain default \
    --password-prompt neutron
User Password: <NEUTRON_PASS>
Repeat User Password: <NEUTRON_PASS>
```

Note: *<NEUTRON_PASS>* is: *neutron;pass*

Add the admin role to the neutron user:

```
controller# openstack role add --project service --user neutron admin
```

Create the neutron service entity:

```
controller# openstack service create --name neutron \
    --description "OpenStack Networking" network
```

Create the Networking service API endpoints:

```
controller# openstack endpoint create --region 'RegionOne' \
    network public http://controller:9696

controller# openstack endpoint create --region 'RegionOne' \
    network internal http://controller:9696
```

```
controller# openstack endpoint create --region 'RegionOne' \
    network admin http://controller:9696
```

8.2. Create Provider Networks on Controller

Controller

Install the packages

```
controller# apt install neutron-server neutron-plugin-ml2 \
    neutron-linuxbridge-agent neutron-dhcp-agent \
    neutron-metadata-agent
```

Edit file /etc/neutron/neutron.conf

```
[DEFAULT]
# ...
auth_strategy = keystone

core_plugin = ml2
service_plugins =
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true

transport_url = rabbit://openstack:<RABBIT_PASS>@controller

[database]
# ...
connection = mysql+pymysql://neutron:<NEUTRON_DBPASS>@controller/neutron

[keystone_authtoken]
# ...
auth_uri = http://controller:5000
memcached_servers = controller:11211
auth_type = password
auth_url = http://controller:35357
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = <NEUTRON_PASS>

[nova]
# ...
region_name = RegionOne
auth_uri = http://controller:5000
```

```
auth_type = password
password = <NOVA_PASS>
project_domain_name = default
project_name = service
user_domain_name = default
username = nova
```

Note: <RABBIT_PASS> is: rabbit;pass
<NEUTRON_DBPASS> is: neutron;pass
<NEUTRON_PASS> is: neutron;pass
<NOVA_PASS> is: nova;pass

Edit file /etc/neutron/plugins/ml2/ml2_conf.ini

```
[ml2]
# ...
type_drivers = flat,vlan
tenant_network_types =
mechanism_drivers = linuxbridge
extension_drivers = port_security

[ml2_type_flat]
# ...
flat_networks = provider

[securitygroup]
# ...
enable_ipset = true
```

Edit file /etc/neutron/plugins/ml2/linuxbridge_agent.ini

```
[linux_bridge]
physical_interface_mappings = provider:enp0s8

[securitygroup]
# ...
firewall_driver =
neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
enable_security_group = true

[vxlan]
enable_vxlan = false
```

Edit file /etc/neutron/dhcp_agent.ini

```
[DEFAULT]
# ...
interface_driver = linuxbridge
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
```

Edit file /etc/neutron/metadata_agent.ini

```
[DEFAULT]
# ...
nova_metadata_ip = controller
metadata_proxy_shared_secret = <METADATA_SECRET>
```

Note: <METADATA_SECRET> is: metadata;secret

Edit file /etc/nova/nova.conf

```
[neutron]
# ...
url = http://controller:9696
region_name = RegionOne
service_metadata_proxy = true
metadata_proxy_shared_secret = <METADATA_SECRET>
auth_type = password
auth_url = http://controller:35357
project_name = service
project_domain_name = default
username = neutron
user_domain_name = default
password = <NEUTRON_PASS>
```

Note: <METADATA_SECRET> is: metadata;secret
<NEUTRON_PASS> is: neutron;pass

Populate the Image Service database

```
controller# su -s /bin/sh -c "neutron-db-manage \
    --config-file /etc/neutron/neutron.conf \
    --config-file /etc/neutron/plugins/ml2/ml2_conf.ini \
    upgrade head" neutron
```

Restart the Compute API services

```
controller# service nova-api restart
```

Restart the Networking services

```
controller# service neutron-server restart
controller# service neutron-linuxbridge-agent restart
controller# service neutron-dhcp-agent restart
controller# service neutron-metadata-agent restart
```

8.3. Add security group rules to the default security group

Controller

Permit ICMP (ping):

```
controller# openstack security group rule create --proto icmp default
```

Permit secure shell (SSH):

```
controller# openstack security group rule create --proto tcp \
    --dst-port 22 default
```

8.4. Create Provider Networks on Compute nodes

Compute Node

Install Networking agent components

```
computeX# apt install neutron-linuxbridge-agent
```

Edit file /etc/neutron/neutron.conf

```
[DEFAULT]
# ...
auth_strategy = keystone

transport_url = rabbit://openstack:<RABBIT_PASS>@controller

[keystone_authtoken]
# ...
auth_uri = http://controller:5000
memcached_servers = controller:11211
auth_type = password

auth_url = http://controller:35357
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = <NEUTRON_PASS>
```

Note: <RABBIT_PASS> is: rabbit;pass
<NEUTRON_PASS> is: neutron;pass
<NOVA_PASS> is: nova;pass

Edit file /etc/neutron/plugins/ml2/linuxbridge_agent.ini

```
[linux_bridge]
physical_interface_mappings = provider:enp0s8

[securitygroup]
# ...
firewall_driver =
neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
enable_security_group = true

[vxlan]
enable_vxlan = false
```

Edit file /etc/nova/nova.conf

```
[neutron]
url = http://controller:9696
region_name = RegionOne
auth_type = password

auth_url = http://controller:35357

project_name = service
project_domain_name = default

username = neutron
user_domain_name = default
password = <NEUTRON_PASS>
```

Note: <METADATA_SECRET> is: metadata;pass
<NEUTRON_PASS> is: neutron;pass

Restart the Compute service

```
computeX# service nova-compute restart
```

Restart the Linux bridge agent

```
computeX# service neutron-linuxbridge-agent restart
```

8.5. Verify operation on Controller

Controller

Source the admin credentials to gain access to admin-only CLI commands:

```
controller# source admin-rc.sh
```

List service components to verify successful launch of each process:

```
controller# openstack extension list --network
```

List service components to verify successful launch of each process:

```
controller# openstack network agent list
```

List security group list:

```
controller# openstack security group list
```

List security group rules:

```
controller# openstack security group rule list
```

8.6. Create Provider Networks

Controller

Source the admin credentials:

```
controller# source admin-rc.sh
```

Create the network:

```
controller# openstack network create  --share \
    --provider-physical-network provider \
    --provider-network-type flat provider
```

Create a subnet on the network:

```
controller# openstack subnet create --network provider \
  --allocation-pool start=START_IP_ADDRESS,end=END_IP_ADDRESS \
  --dns-nameserver DNS_RESOLVER --gateway PROVIDER_NETWORK_GATEWAY \
  --subnet-range PROVIDER_NETWORK_CIDR provider
```

Note: Replace START_IP_ADDRESS, END_IP_ADDRESS, DNS_RESOLVER, PROVIDER_NETWORK_GATEWAY, PROVIDER_NETWORK_CIDR with network that provides to VMs

Example

```
controller# openstack subnet create --network provider \
  --allocation-pool start=192.168.0.101,end=192.168.0.202 \
  --dns-nameserver 8.8.4.4 --gateway 192.168.0.1 \
  --subnet-range 192.168.0.0/24 provider
```

Verify Provider Networks:

```
controller# openstack network list

controller# openstack subnet list
```

---

9. Lunch instance (via command line)

9.1. Determine instance options

Controller

Source the admin credentials:

```
controller# source demo-rc.sh
```

Generate a key pair and add a public key:

```
controller# ssh-keygen -q -N ""
controller# openstack keypair create \
    --public-key ~/.ssh/id_rsa.pub demo_key
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| fingerprint | b3:1f:48:75:c7:39:4b:83:3e:bb:72:84:eb:7d:b4:ae |
| name        | demo_key                                        |
| user_id     | cd5162d22d0d41288ae88fb5bf238d17                |
+-------------+-------------------------------------------------+
```

List available flavors:

```
controller# openstack flavor list
+----+---------+-----+------+-----------+-------+-----------+
| ID | Name    | RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+---------+-----+------+-----------+-------+-----------+
| 0  | m1.nano |  64 |    1 |         0 |     1 | True      |
| 1  | m1.tiny | 512 |    1 |         0 |     1 | True      |
+----+---------+-----+------+-----------+-------+-----------+
```

List available images:

```
controller# openstack image list
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| a4bb1442-8803-4118-8e37-dbcd513bca58 | CirrOS | active |
+--------------------------------------+--------+--------+
```

List available networks:

```
controller# openstack network list
+--------------------------------------+----------+-----------------------+
| ID                                   | Name     | Subnets               |
+--------------------------------------+----------+-----------------------+
| efc84802-c548-47d0-8b33-bce79cebcb62 | provider | 5bbef91...40de3c1d9bc |
+--------------------------------------+----------+-----------------------+
```

List available security groups:

```
controller# openstack security group list
+--------------------------------------+---------+-------------+---------+
| ID                                   | Name    | Description | Project |
+--------------------------------------+---------+-------------+---------+
| 331e6b3a-a7de-49ec-88ef-a063eb96c9f6 | default | Default .... |         |
+--------------------------------------+---------+-------------+---------+
```

9.2. Launch an instance

Controller

Source the admin credentials:

```
controller# source demo-rc.sh
```

Launch the instance:

```
controller# openstack server create --flavor m1.nano --image IMAGE_NAME \
  --nic net-id=PROVIDER_NET_ID --security-group SECURITY_GROUP_NAME \
  --key-name KEY_PAIR_NAME MyFirstInstance
```

Note: Replace IMAGE_NAME, PROVIDER_NET_ID, SECURITY_GROUP_NAME and KEY_PAIR_NAME with id from command above

Check the status of your instance:

```
controller# openstack server list
```

Note: Wait until Status changed from BUILD to ACTIVE

---

10. Add the dashboard

10.1. Install and configure

Controller

Install the packages:

```
controller# apt install apache2 memcached
controller# apt install openstack-dashboard
```

Edit file /etc/openstack-dashboard/local_settings.py

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}

OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'default'
CACHES = {
   'default': {
      'BACKEND' :
'django.core.cache.backends.memcached.MemcachedCache',
      'LOCATION' : '127.0.0.1:11211',
   }
}
OPENSTACK_HOST = "controller"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v2.0" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"

OPENSTACK_NEUTRON_NETWORK = {
    'enable_router': False,
    'enable_quotas': False,
    'enable_ipv6': False,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_lb': False,
    'enable_firewall': False,
```

```
    'enable_vpn': False,
    'enable_fip_topology_check': False,
    #...
}

TIME_ZONE = "Asia/Bangkok"

WEBROOT = '/horizon/'
ALLOWED_HOSTS = ['*',]
```

Change access permission of /var/lib/openstack-dashboard/ for apache2 user

```
controller# chown -R www-data:www-data /var/lib/openstack-dashboard
```

Reload the web server configuration:

```
controller# service apache2 reload
```

10.2. Verify operation

Controller

Access the dashboard using a web browser at http://controller/horizon.

Authenticate using admin or demo user and default domain credentials.

---

11. Add the Block Storage service

11.1. Setup storage cluster (GlusterFS) on compute node

Compute Node

Install the packages on all nodes:

```
computeX# apt install glusterfs-client glusterfs-common \
    glusterfs-server attr
```

Probe and add nodes in to peer list on all nodes:

```
computeX# gluster peer probe block0
computeX# gluster peer probe block1
computeX# gluster peer probe block2
```

Setup glusterfs partitions on all nodes:

```
computeX# mkfs.xfs -f -i size=512 /dev/sdb1
computeX# mkfs.xfs -f -i size=512 /dev/sdc1
```

Create mount points on all nodes;

```
computeX# mkdir -p /gtfs/vms
computeX# mkdir -p /gtfs/img
```

Add new mount partitions to file /etc/fstab on all nodes:

```
#...
/dev/sdb1  /gtfs/vms    xfs      defaults   0  0
/dev/sdc1  /gtfs/img    xfs      defaults   0  0
```

Mount new partition on all nodes:

```
computeX# mount -arw
```

Create sub directory as a brick in new partitions (mount points) on all nodes:

```
computeX# mkdir -p /gtfs/vms/brick
computeX# mkdir -p /gtfs/img/brick
```

Change owner of bricks on all nodes:

```
computeX# chown cinder:cinder /gtfs/vms/brick
computeX# chown cinder:cinder /gtfs/img/brick
```

Create and start gluster volumes -- 2 nodes

```
computeX# gluster volume create gluster_vms_volume replica 2 \
                transport tcp \
                block1:/gtfs/vms/brick \
                block2:/gtfs/vms/brick \
                force
computeX# gluster volume start gluster_vms_volume

computeX# gluster volume create gluster_img_volume replica 2 \
                transport tcp \
                block1:/gtfs/img/brick \
                block2:/gtfs/img/brick \
                force
computeX# gluster volume start gluster_img_volume
```

Set each Gluster volume to use the same UID and GID as the cinder user:

```
computeX# id cinder
uid=121(cinder) gid=127(cinder) groups=127(cinder)
```

```
computeX# gluster volume set gluster_vms_volume storage.owner-uid 121
computeX# gluster volume set gluster_vms_volume storage.owner-gid 127

computeX# gluster volume set gluster_img_volume storage.owner-uid 121
computeX# gluster volume set gluster_img_volume storage.owner-gid 127
```

Configure each Gluster volume to accept libgfapi connections:

```
computeX# gluster volume set gluster_vms_volume server.allow-insecure on
computeX# gluster volume set gluster_img_volume server.allow-insecure on
```

Restart the Glusterfs services:

```
computeX# service glusterfs-server restart
```

11.2. Setup glance on controller to use Glusterfs as image storage

Controller

Install the packages on all nodes:

```
controller# apt install glusterfs-client glusterfs-common \
    glusterfs-server attr
```

Probe and add nodes in to peer list on all nodes:

```
controller# gluster peer probe block0
controller# gluster peer probe block1
controller# gluster peer probe block2
```

Edit /etc/fstab:

```
localhost:/gluster_img_volume /var/lib/glance/images  glusterfs
                          defaults,_netdev,fetch-attempts=10  0  0
```

Create mount point:

```
controller# mkdir -p /var/lib/glance/images
```

Restart the Glusterfs services:

```
controller# service glusterfs-server restart
```

Mount a new mount point:

```
controller# mount -arw
```

Change image storage ownder to be glance:

```
controller# chown -R glance:glance /var/lib/glance/images
```

11.3. Configure Compute to use Glusterfs

11.4. Option 1: Setup compute nodes to use Glusterfs volume without Cinder

Compute Node

Edit /etc/fstab:

```
localhost:/gluster_vms_volume /var/lib/cinder/volumes  glusterfs
                            defaults,_netdev,fetch-attempts=10  0  0
```

Create mount point:

```
computeX# mkdir -p /var/lib/cinder/volumes
```

Mount a new mount point:

```
computeX# mount -arw
```

Change image storage ownder to be glance:

```
computeX# chown -R cinder:cinder /var/lib/cinder/volumes
```

11.5. Option 2: Setup compute nodes to use Glusterfs volume with Cinder

11.5.1. Configure Block storage service

Controller

Connect to MySQL database

```
controller# mysql -u root -p
Enter password: <MYSQL_PASSWORD>
```

Note: *<MYSQL_PASSWORD>* is: *openstack;pass*

Create cinder database

```
MariaDB [(none)]> CREATE DATABASE cinder;
```

Create and grant cinder user to cinder database

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO \
  'cinder'@'localhost' IDENTIFIED BY '<CINDER_DBPASS>';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO \
  'cinder'@'%' IDENTIFIED BY '<CINDER_DBPASS>';
```

```
Note: <CINDER_DBPASS> is: cinder;pass
```

Apply new privileges

```
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit
```

Source the admin credentials

```
controller# source admin-rc.sh
```

Create the cinder user:

```
controller# openstack user create --domain default \
    --password-prompt cinder
User Password: <CINDER_PASS>
Repeat User Password: <CINDER_PASS>
```

```
Note: <CINDER_PASS> is: cinder;pass
```

Add the admin role to the cinder user:

```
controller# openstack role add --project service --user cinder admin
```

Create the cinder service entity:

```
controller# openstack service create --name cinder \
    --description "OpenStack Block Storage" volume

controller# openstack service create --name cinderv2 \
    --description "OpenStack Block Storage" volumev2
```

Create the Block Storage service API endpoints:

```
controller# openstack endpoint create --region 'RegionOne' \
    volume public http://controller:8776/v1/%\(tenant_id\)s

controller# openstack endpoint create --region 'RegionOne' \
    volume internal http://controller:8776/v1/%\(tenant_id\)s

controller# openstack endpoint create --region 'RegionOne' \
    volume admin http://controller:8776/v1/%\(tenant_id\)s
```

```
controller# openstack endpoint create --region 'RegionOne' \
    volumev2 public http://controller:8776/v2/%\(tenant_id\)s

controller# openstack endpoint create --region 'RegionOne' \
    volumev2 internal http://controller:8776/v2/%\(tenant_id\)s

controller# openstack endpoint create --region 'RegionOne' \
    volumev2 admin http://controller:8776/v2/%\(tenant_id\)s
```

Install the packages:

```
controller# apt install cinder-api cinder-scheduler
```

Edit file /etc/cinder/cinder.conf

```
      [DEFAULT]
      # ...
      my_ip = 10.0.0.100
      auth_strategy = keystone

      transport_url = rabbit://openstack:<RABBIT_PASS>@controller

      [database]
      # ...
connection = mysql+pymysql://cinder:<CINDER_DBPASS>@controller/cinder

      [keystone_authtoken]
      # ...
      auth_uri = http://controller:5000
      memcached_servers = controller:11211
      auth_type = password
      auth_url = http://controller:35357
      project_domain_name = default
      project_name = service
      user_domain_name = default
      username = cinder
      password = <CINDER_PASS>

      [oslo_concurrency]
      # ...
      lock_path = /var/lib/cinder/tmp
```

Note: <RABBIT_PASS> is: rabbit;pass
<CINDER_DBPASS> is: cinder;pass

<CINDER_PASS> is: *cinder;pass*

Populate the Block Storage database:

```
controller# su -s /bin/sh -c "cinder-manage db sync" cinder
```

Install the packages on controller:

```
controller# apt install glusterfs-client cinder-volume
```

Configure OpenStack Block Storage to use the /etc/cinder/glusterfs, edit file /etc/cinder/cinder.conf file:

```
[DEFAULT]
#...
volume_driver = cinder.volume.drivers.glusterfs.GlusterfsDriver
glusterfs_shares_config = /etc/cinder/glusterfs
glusterfs_mount_point_base = /var/lib/cinder/volumes
```

Create and add entry to /etc/cinder/glusterfs

```
localhost:/gluster_vms_volume
```

Set /etc/cinder/glusterfs to be owned by the root user and the cinder group:

```
controller# chown root:cinder /etc/cinder/glusterfs
```

Set /etc/cinder/glusterfs to be readable by members of the cinder group:

```
controller# chmod 0640 /etc/cinder/glusterfs
```

Edit /etc/glusterfs/glusterd.vol:

```
volume management
    #...
    option rpc-auth-allow-insecure on
end-volume
```

Set each Gluster volume to use the same UID and GID as the cinder user:

```
controller# id cinder
uid=121(cinder) gid=127(cinder) groups=127(cinder)

controller# gluster volume set gluster_vms_volume storage.owner-uid 121
controller# gluster volume set gluster_vms_volume storage.owner-gid 127
```

Restart the Glusterfs services:

```
controller# service glusterfs-server restart
```

Restart the Block Storage services:

```
controller# service nova-api restart

controller# service cinder-scheduler restart

controller# service apache2 restart
```

11.5.2. Configure Compute to use Block Storage

Compute Node

Edit the /etc/nova/nova.conf:

```
[cinder]
# ...
os_region_name = RegionOne
```

Restart Compute services:

```
compute# service nova-compute restart
```

---

12. Create OS Image -- on other host

12.1. Create Ubuntu image

Create disk image:

```
_# qemu-img create -f qcow2 ./trusty.qcow2 8G
```

Install OS in new image:

```
_# virt-install --virt-type kvm --name trusty --ram 1024 \
  --cdrom=/data/isos/trusty-64-mini.iso \
  --disk ./trusty.qcow2,format=qcow2 \
  --network network=default \
  --graphics vnc,listen=0.0.0.0 --noautoconsole \
  --os-type=linux --os-variant=ubuntutrusty
```

Step through the installation:

- Install os
- Setup hostname
- Select a mirror
- Step through the install

- Partition the disks
- Select automatic updates
- Select software: OpenSSH server
- Install GRUP boot loader
- Detach the CD-ROM and reboot

Start up the machine again as paused, eject the disk and resume:

```
_# virsh start trusty --paused
_# virsh attach-disk --type cdrom --mode readonly trusty "" hdc
_# virsh resume trusty
```

Log in to newly created image and Install cloud-init:

```
_# apt-get install cloud-init
_# dpkg-reconfigure cloud-init
```

Clean up (remove MAC address details):

```
_# virt-sysprep -d trusty
```

Undefine the libvirt domain:

```
_# virsh undefine trusty
```

12.2. Create Ubuntu image

Create disk image:

```
_# qemu-img create -f qcow2 ./debian-8.0.0.qcow2 8G
```

Install OS in new image

```
_# virt-install --virt-type kvm \
     --name Debian-8.0.0 \
     --ram 1024 \
     --cdrom=./debian-8.0.0-amd64-netinst.iso \
     --disk path=./debian-8.0.0.qcow2,size=8,format=qcow2 \
     --os-type=linux \
     --os-variant=generic \
     --network bridge=br100 \
     --graphics vnc,listen=0.0.0.0 \
     --noautoconsole
```

Show VNC remote connection

```
_# virsh vncdisplay Debian-8.0.0
:1
```

Connect VNC viewer to x.x.x.x:5901, start to setup a minimum os, and shutdown.

Clean up (remove MAC address details):

```
_# virt-sysprep -d Debian-8.0.0
```

Undefine the libvirt domain:

```
_# virsh undefine Debian-8.0.0
```

SSH Setup for remote execute

All Nodes

Edit sshd_config

```
_# vi /etc/ssh/sshd_config
:
PermitRootLogin yes
:
```

Restart sshd

```
_# service ssh restart
```

Generate ssh rsa key:

```
_# ssh-keygen -t rsa -b 2048
  Enter file in which to save the key (/root/.ssh/id_rsa): <blank>
  Enter passphrase (empty for no passphrase): <blank>
  Enter same passphrase again: <blank>
```

Merge file /root/.ssh/id_rsa.pub of all hosts to /root/.ssh/authorized_keys:

Controller

```
controller# cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
```

Copy file /root/.ssh/authorized_keys to each Compute node sequential:

```
controller# scp /root/.ssh/authorized_keys compute1:/root/.ssh/authorized_keys
```

Merge file /root/.ssh/id_rsa.pub of all hosts to /root/.ssh/authorized_keys:

compute1

```
compute1# cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
```

Copy file /root/.ssh/authorized_keys to each Compute node sequential:

```
compute1# scp /root/.ssh/authorized_keys compute2:/root/.ssh/authorized_keys
```

Merge file /root/.ssh/id_rsa.pub of current hosts to /root/.ssh/authorized_keys:

compute2

```
compute2# cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
```

Copy file /root/.ssh/authorized_keys to each Compute node sequential:

```
compute2# scp /root/.ssh/authorized_keys compute3:/root/.ssh/authorized_keys
:
:
```

On last compute node, copy completed file /root/.ssh/authorized_keys back to previous nodes:

computeX

```
computeX# scp /root/.ssh/authorized_keys controller:/root/.ssh/authorized_keys
computeX# scp /root/.ssh/authorized_keys compute1:/root/.ssh/authorized_keys
computeX# scp /root/.ssh/authorized_keys compute2:/root/.ssh/authorized_keys
:
```

Network Tuning

All Nodes

Install packages

```
_# apt-get install ethtool -y
```

Edit file /etc/rc.local

```
#...
/etc/openstack.sh

exit 0
```

Create script file /etc/openstack.sh

```
#!/bin/bash
#...
/sbin/ethtool -G <net_dev> tx <N> rx <N>
```

example:

```
/sbin/ethtool -G eth0 tx 2048 rx 2048
/sbin/ethtool -G eth2 tx 511 rx 511
```

Change mode of script file /etc/openstack.sh

```
_# chmod a+x /etc/openstack.sh
```