Install your own OpenStack cloud

OpenStack Compute (Nova) is an open source IaaS cloud computing platform (www.openstack.org) written in the Python programming language.

I will describe in detail the installation and configuration of my own OpenStack platform. You can use it to install your own private or public cloud.

We will install OpenStack on two physical servers:

The **node1** will be:

- the **cloud controller node** (running nova-api, nova-scheduler, nova-objectstore/glance, nova-network, MySQL, RabbitMQ, nova-dashboard)
- a **compute node** (running nova-compute and KVM)
- a **volume node** (running nova-volume and iSCSI)

The **node2** will be:

- a **compute node** (running nova-compute and KVM)
- a **volume node** (running nova-volume and iSCSI)

It means that an instance or a volume can either be created on the node1 or the node2 (the nova-scheduler decides where to create it). It means also that if you de-activate the node2, you can still provision new instances and new volumes: the node1 can run in stand-alone mode.

Software versions:

- Operating System (OS): Linux Ubuntu Server Natty (11.04) 64 bits (for the hosts and for the instances/images)
- Cloud Computing (IaaS): OpenStack Compute (Nova) 2011.3-dev (diablo-dev / trunk) (diablo-1 at this moment)

We will install the latest version of OpenStack Compute (diablo-dev) which is still in development.

- OpenStack latest stable version: 2011.2 (cactus / release)
- OpenStack latest development version: 2011.3-dev (diablo-dev / trunk)

Hardwares:

- node1:

CPU: 1 x Intel i7
RAM: 8 GB
HDD: 2 x 1 GB (sda & sdb)
NIC: 2 (eth0 & eth1)

- node2:

CPU: 1 x Intel Core2Quad
RAM: 4 GB
HDD: 2 x 500 MB (sda & sdb)
NIC: 2 (eth0 & eth1)

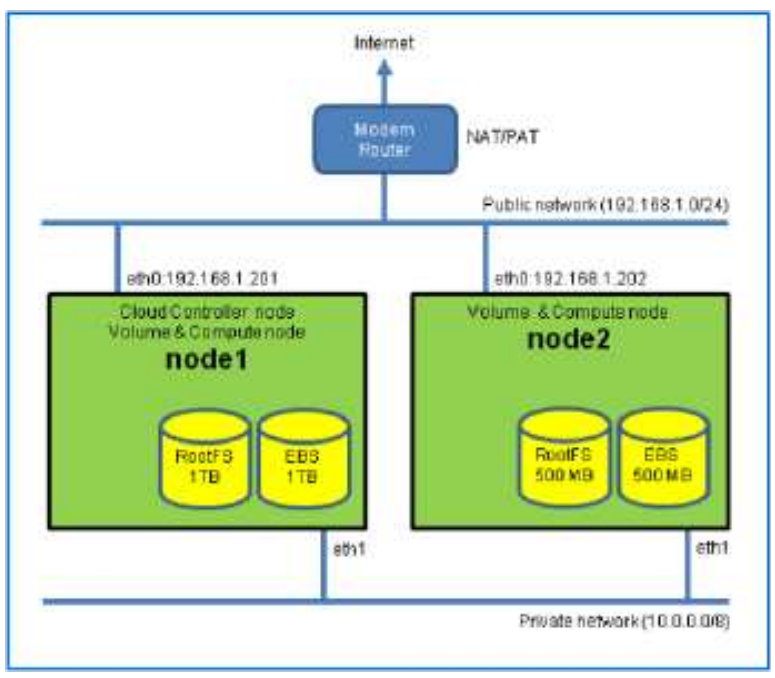
Networks:

Network type: VLAN (VlanNetworkManager)

The node1 (nova-network) is the network gateway: the floating IPs and the private default gateways IPs (10.0.X.7) are configured on it. The node1 acts as a router gateway to access the instances on the node1 or on the node2.

- Network1 (eth0): public / external network – 192.168.1.0/24
- Network2 (eth1): private / internal network (managed by OpenStack) – 10.0.0.0/8

node1: 192.168.1.201
node2: 192.168.1.202



The two nodes and the two networks

Picture of my two OpenStack nodes

1) Install node1

- Configure the PPAs (Ubuntu Personal Package Archives):

```
# aptitude install python-software-properties
# add-apt-repository ppa:nova-core/trunk
# add-apt-repository ppa:glance-core/trunk
# aptitude update
```

Remark: to install the stable (2011.2 / cactus) version of OpenStack: either use ppa:nova-core/release, or the Ubuntu repositories.

- Install RabbitMQ (the messaging system):

```
# aptitude install rabbitmq-server
```

- Install different utilities:

```
# aptitude install python-greenlet python-mysqldb unzip lvm2 vlan
```

- Install all the OpenStack components (six):

```
# aptitude install python-nova nova-common nova-doc
# aptitude install nova-api nova-objectstore nova-scheduler nova-network
# aptitude install nova-compute nova-volume
```

Important remarks:

- Qemu/KVM and libvirt are automatically installed when installing nova-compute.
- No need to configure manually Linux Ethernet bridges for KVM to run: these bridges are automatically configured on the nodes by OpenStack when it is needed (there is one bridge per project).

- Install Glance (the advanced image service):

```
# aptitude install glance python-glance-doc
```

- Install Euca2ools (the Amazon EC2 CLI client):

```
# aptitude install euca2ools
```

- Install MySQL (the DB server) with root password 123456:

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

- Create the nova DB and nova username with password 123456:

```
# mysql -uroot -p123456 -e 'CREATE DATABASE nova;'
# mysql -uroot -p123456 -e "GRANT ALL PRIVILEGES ON *.* TO 'nova'@'%' WITH GRANT OPTION;"
# mysql -uroot -p123456 -e "SET PASSWORD FOR 'nova'@'%' = PASSWORD('123456');"
```

- Create the tables:

```
# nova-manage db sync
```

- Give full access to the nova user:

```
# visudo

nova ALL=(ALL) NOPASSWD:ALL
```

2) Install node2

- Configure the PPAs (Ubuntu Personal Package Archives):

```
# aptitude install python-software-properties
# add-apt-repository ppa:nova-core/trunk
# aptitude update
```

- Install different utilities:

```
# aptitude install python-greenlet python-mysqldb unzip lvm2 vlan
```

- Install the two OpenStack components:

```
# aptitude install python-nova nova-common nova-doc
# aptitude install nova-compute nova-volume
```

- Give full access to the nova user:

```
# visudo

nova ALL=(ALL) NOPASSWD:ALL
```

3) Configure node1 and node2

- On both nodes:

```
# vi /etc/hosts

192.168.1.201 node1
192.168.1.202 node2

# addgroup nova
# chown -R root:nova /etc/nova
# chmod 644 /etc/nova/nova.conf

# vi /etc/nova/nova.conf

# RabbitMQ
--rabbit_host=192.168.1.201
# MySQL
--sql_connection=mysql://nova:123456@192.168.1.201/nova
# Networking
--network_manager=nova.network.manager.VlanManager
--vlan_interface=eth1
--public_interface=eth0
--network_host=192.168.1.201
--routing_source_ip=192.168.1.201
--fixed_range=10.0.0.0/8
--network_size=1024
--dhcpbridge_flagfile=/etc/nova/nova.conf
--dhcpbridge=/usr/bin/nova-dhcpbridge
# Virtualization
--libvirt_type=kvm
# Volumes
--iscsi_ip_prefix=192.168.1.20
--num_targets=100
# APIs
--auth_driver=nova.auth.dbdriver.DbDriver
--cc_host=192.168.1.201
--ec2_url=http://192.168.1.201:8773/services/Cloud
--s3_host=192.168.1.201
--s3_dmz=192.168.1.201
# Image service
--glance_host=192.168.1.201
--image_service=nova.image.glance.GlanceImageService
# Misc
--logdir=/var/log/nova
--state_path=/var/lib/nova
--lock_path=/var/lock/nova
--verbose
# VNC Console
--vnc_enabled=true
--vncproxy_url=http://lcc.louvrex.net:6080
--vnc_console_proxy_url=http://lcc.louvrex.net:6080
```

Some parameters in the *nova.conf* file are the default ones and then do not need to be put in the configuration file. But for clarity, I prefer them to be present.

Examples of some default parameters:

Parameter: **network_manager** (type of networking used on the internal / private network used by the instances)

- nova.network.manager.VlanManager ==> **Default** — The most sophisticated OpenStack network mode (one VLAN, one network subnet, and one DHCP server per project)
- nova.network.manager.FlatManager ==> One flat network for all projects (no DHCP server)
- nova.network.manager.FlatDHCPManager ==> One flat network for all projects (with a DHCP server)

Parameter: **libvirt_type** (type of virtualization used on the compute nodes)

- kvm ==> **Default** — Linux Kernel-based Virtual Machine (hardware virtualization technology like Intel VT-x is required)
- qemu ==> You can use it if you install OpenStack in a VM or on a HW without hardware virtualization technology like Intel VT-x
- lxc ==> LinuX Container (based on the Linux kernel; virtualization of the OS, not of the HW; similar to OpenVZ and Parallels Virtuozzo)

Virtualization which are also supported by OpenStack:

- Xen
- XenServer (Citrix)
- ESX (VMware)
- Hyper-V (Microsoft)

- UML

Configure nova-volume:

- Create one LVM primary partition (sdb1) on the second HDD:

```
# cfdisk /dev/sdb
```

- Create one LVM physical volume:

```
# pvcreate /dev/sdb1
```

- Create one LVM volume group called “nova-volumes”:

```
# vgcreate nova-volumes /dev/sdb1
```

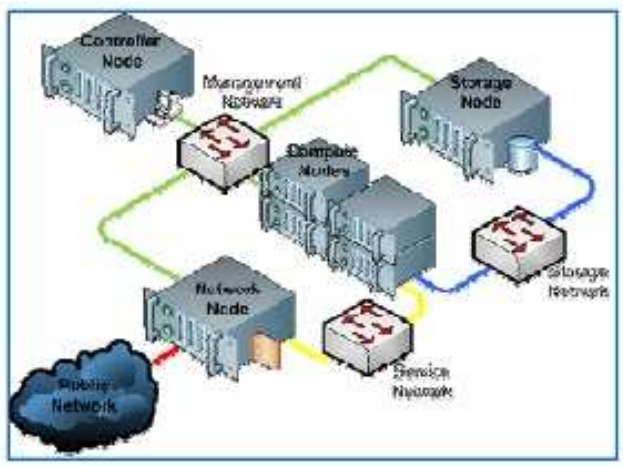
- Start the iSCSI service automatically:

```
# sed -i 's/false/true/g' /etc/default/iscsitarget
```

- Start the nova-volume component and the iSCSI service:

```
# start iscsitarget
# start nova-volume
```

Please note that in our configuration, the iSCSI traffic will pass on the public/external network. This traffic flows between the nova-volume components and the nova-compute components. The nova-compute components then “expose” the storage volumes to the attached instances. In a bigger configuration with more than two nodes, a dedicated “storage network” should be foreseen:



OpenStack multinode architecture (c) StackOps
(www.stackops.org)

- On the node1 only:

- To allow the node1 to act as a router:

```
# vi /etc/sysctl.conf

net.ipv4.ip_forward=1
```

In the VLAN network mode, each project is given a specific VLAN/subnet. We will configure three VLANs/subnets (feel free to create much more):

- VLAN: 1 — Subnet: 10.0.1.0/24 — Bridge name: br_vlan1:

```
# nova-manage network create 10.0.1.0/24 1 256
```

- VLAN: 2 — Subnet: 10.0.2.0/24 — Bridge name: br_vlan2:

```
# nova-manage network create 10.0.2.0/24 1 256
```

- VLAN: 3 — Subnet: 10.0.3.0/24 — Bridge name: br_vlan3:

```
# nova-manage network create 10.0.3.0/24 1 256
```

- Update the DB (this is still not made automatically; you have to use the “network-update.sh” script for that):

```
# mysql nova
mysql> select id,cidr from networks;
```

Result:

```
+----+-------------+
| id | cidr        |
+----+-------------+
| 13 | 10.0.1.0/24 |
| 14 | 10.0.2.0/24 |
| 15 | 10.0.3.0/24 |
+----+-------------+
```

Script: **network-update.sh** ***vlan id***

```
# ./network-update.sh 1 13
# mysql -p123456 nova < vlan.sql
```

```
# ./network-update.sh 2 14
# mysql -p123456 nova < vlan.sql

# ./network-update.sh 3 15
# mysql -p123456 nova < vlan.sql
```

This is the **network-update.sh** bash script:

```
# vi network-update.sh

if [ -z $1 ]; then
 echo "You need to specify the vlan to modify"
fi

if [ -z $2 ]; then
 echo "You need to specify a network id number (check the DB for the network you want to update)"
fi

VLAN=$1
ID=$2

cat > vlan.sql << __EOF_
update networks set vlan = '$VLAN' where id = $ID;
update networks set bridge = 'br_vlan$VLAN' where id = $ID;
update networks set gateway = '10.0.$VLAN.7' where id = $ID;
update networks set dhcp_start = '10.0.$VLAN.8' where id = $ID;
update fixed_ips set reserved = 1 where address in ('10.0.$VLAN.1','10.0.$VLAN.2','10.0.$VLAN.3','10.0.$VLAN.4','10.0.$VLAN.5','10.0.
__EOF_

# chmod +x network-update.sh
```

For each VLAN/subnet on the private network (one per project), the IPs are as follow:

*** VLAN1:**

- Network subnet: 10.0.1.0/24 – Reserved IPs (not used): 10.0.1.1->10.0.1.6
- Default gateway (automatically configured on the node1/nova-network): 10.0.1.7
- Instance IPs (automatically distributed via DHCP to the instances): 10.0.1.8->10.0.1.254
- Bridge name: br_vlan1

*** VLAN2:**

- Network subnet: 10.0.2.0/24
- Reserved IPs (not used): 10.0.2.1->10.0.2.6
- Default gateway (automatically configured on the node1/nova-network): 10.0.2.7
- Instance IPs (automatically distributed via DHCP to the instances): 10.0.2.8->10.0.2.254
- Bridge name: br_vlan2

*** VLAN3:**

- Network subnet: 10.0.3.0/24
- Reserved IPs (not used): 10.0.3.1->10.0.3.6
- Default gateway (automatically configured on the node1/nova-network): 10.0.3.7
- Instance IPs (automatically distributed via DHCP to the instances): 10.0.3.8->10.0.3.254
- Bridge name: br_vlan3

In fact, for each project, a specific VLAN and subnet is attributed, but also:

- a specific Ethernet bridge is configured on the nodes hosting the project's instances;
- a specific DHCP server (dnsmasq) is launched on node1/nova-network to serve IPs to the project's instances.

The **ip addr** (show IP addresses) and **brctl show** (show bridge interfaces) commands on the node1 can give a result like this (I made a lot of cleaning):

```
root@node1:~# ip addr
```

1: lo
inet 127.0.0.1/8
inet 169.254.169.254/32 **–> Amazon EC2 metadata service**

2: eth0 **–> First physical Ethernet port (connected to the external / public network)**
ether 00:24:1d:d3:a1:e6
inet 192.168.1.201/24 **–> node1 public IP**
inet 192.168.1.240/32 **–> elastic IP n° 1 (associated to an instance running on the node1-2)**
inet 192.168.1.241/32 **–> elastic IP n° 2 (associated to an instance running on the node1-2)**

3: eth1 **–> Second physical Ethernet port (connected to the internal / private network)**
ether 00:10:18:34:c0:e5

4: virbr0 **–> bridge configured by the libvirt API**
ether ae:4e:3d:1f:97:3b
inet 192.168.122.1/24

5: vlan1@eth1 **–> VLAN1 tagged eth1 interface**
ether 00:10:18:34:c0:e5

6: br_vlan1 **–> bridge connected on the vlan1@eth1 interface**
ether 00:10:18:34:c0:e5
inet 10.0.1.7/24 **–> Default gateway of the first VLAN network (e.g. for the 1st project)**

7: vlan2@eth1 **–> VLAN2 tagged eth1 interface**
ether 00:10:18:34:c0:e5

8: br_vlan2 **–> bridge connected on the vlan2@eth1 interface**
ether 00:10:18:34:c0:e5
inet 10.0.2.7/24 **–> Default gateway of the second VLAN network (e.g. for the 2nd project)**

9: vlan3@eth1 **–> VLAN3 tagged eth1 interface**
ether 00:10:18:34:c0:e5

10: br_vlan3 **–> bridge connected on the vlan3@eth1 interface**
ether 00:10:18:34:c0:e5
inet 10.0.3.7/24 **–> Default gateway of the third VLAN network (e.g. for the 3rd project)**

11: vnet0 **–> virtual interface for the first instance running on the node1**
ether fe:16:3e:2a:a3:f1

12: vnet1 **–> virtual interface for the second instance running on the node1**
ether fe:16:3e:46:07:6b

13: vnet2 **–> virtual interface for the third instance running on the node1**
ether fe:16:3e:34:53:06

```
root@node1:~# brctl show
```

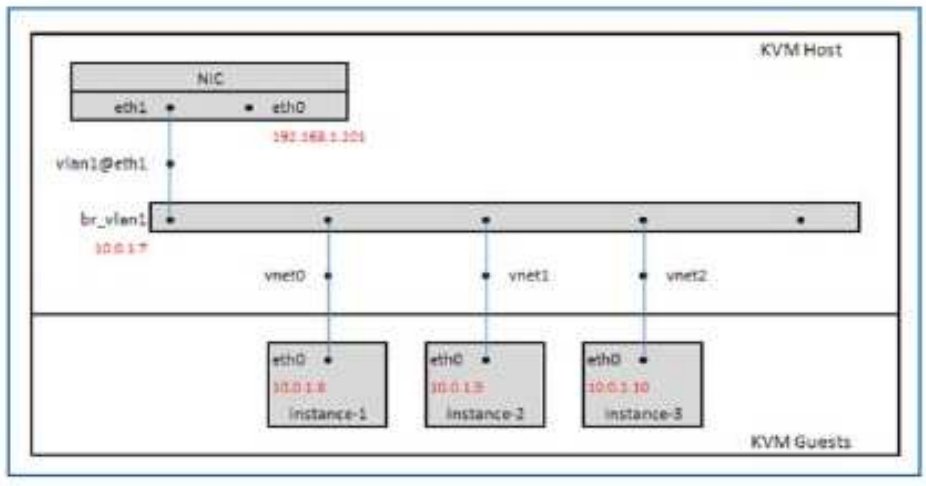
br_vlan1 vlan1

- vnet0 **–> virtual interface for the 1st instance running on the node1 (VLAN1)**
- vnet1 **–> virtual interface for the 2nd instance running on the node1 (VLAN1)**
- vnet2 **–> virtual interface for the 3rd instance running on the node1 (VLAN1)**

br_vlan2 vlan2

br_vlan3 vlan3

virbr0



The OpenStack's VLAN networking in picture

- Configure the public/external floating/elastic IPs (8 IPs: 192.168.1.240->192.168.1.247):

```
# nova-manage floating create node1 192.168.1.240/29
```

iptables is used to configure the floating/elastic IPs on nova-network (node1 in our case). **iptables** is also used to configure the firewall rules (security groups) on nova-compute (node1 and node2 in our case).

For the floating/elastic IPs, the NAT table is used. You can see these NATing rules on the node1 with this command:

```
# iptables -nL -t nat
```

For the firewall rules (security groups), the filter table is used. You can see them on the node1 or the node2 with this command:

```
# iptables -nL -t filter
```

4) Internet access

My personal home network has only one public/internet IP address, and it is a dynamic IP changing every 36 hours.

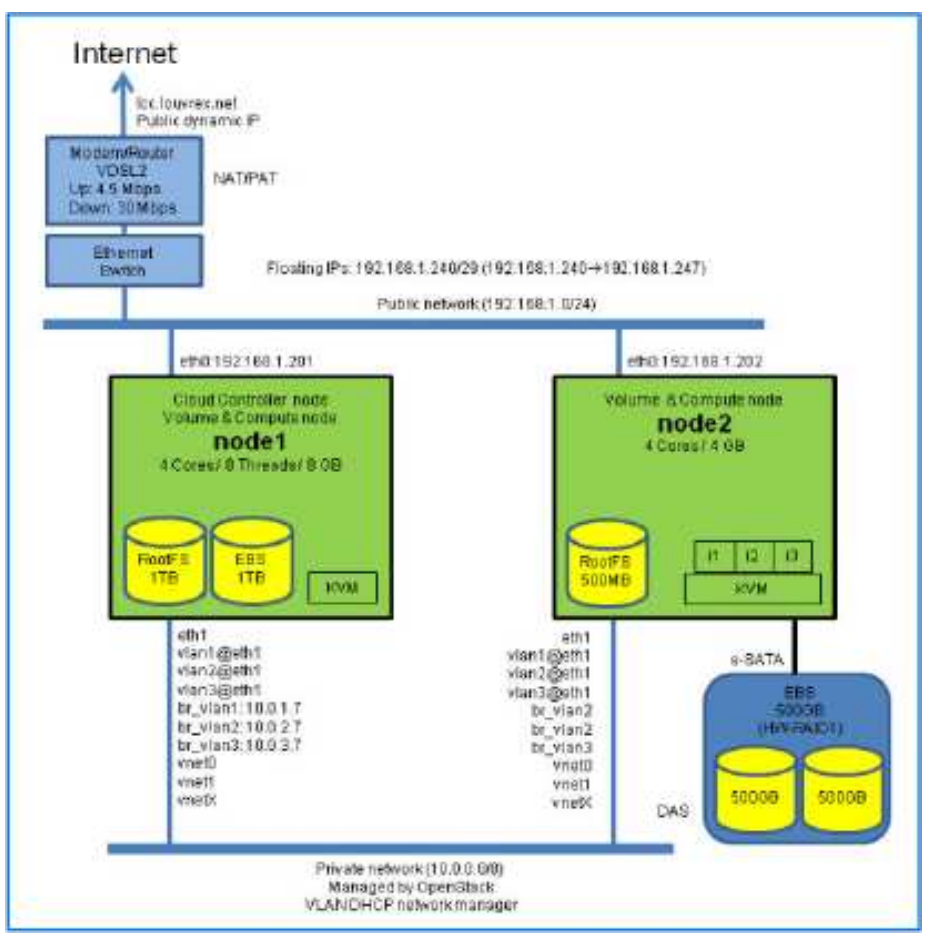
My cloud is available from the Internet with the DNS name **lcc.louvrex.net** (LCC = Louvrex Cloud Computing). The lcc.louvrex.net DNS name is linked to my dynamic IP address. I use the DynDNS service with the **ddclient** running on the node1 to update the DNS name automatically.

Different NAT/PAT rules are configured in the router to access the dashboard web interface, the api, the nodes, and the instances. Here a sample of these rules:

```
Rule name           Internet port     LAN port      LAN IP
-------------------------------------------------------------
node1-ssh           2201              22            192.168.1.201
node2-ssh           2202              22            192.168.1.202
-------------------------------------------------------------
eip1-http           80                80            192.168.1.240
eip2-http           8041              80            192.168.1.241
eip3-http           8042              80            192.168.1.242
eip4-http           8043              80            192.168.1.243
-------------------------------------------------------------
eip1-ssh            22                22            192.168.1.240
```

```
eip2-ssh            2241              22            192.168.1.241
eip3-ssh            2242              22            192.168.1.242
eip4-ssh            2243              22            192.168.1.243
-----------------------------------------------------------------
node1-dashboard     8000              8000          192.168.1.201
node1-api           8773              8773          192.168.1.201
node1-vnc-proxy     6080              6080          192.168.1.201
```

- eip = elastic IP / floating IP (to be associated to instances)
- nova-dashboard = Web interface for the cloud administrator (http://lcc.louvrex.net:8000/admin) and for the cloud end user (http://lcc.louvrex.net:8000)
- nova-api = Amazon EC2 API (used by the euca2tools CLI and by the dashboard web interface)
- vnc-proxy = to access the console GUI of every instance via VNC from the dashboard web interface



A more complete drawing of the configuration

FS = File System
EBS = Elastic Block Storage volume (equivalent to iSCSI SAN)
KVM = Kernel-based Virtual Machine
br = Ethernet BRidge
vnet = Virtual Network (virtual Ethernet interface)
i = Instance (virtual machine)
DAS = Directly Attached Storage
SAN = Storage Attached Network
NAT/PAT = Network Address Translation / Port Address Translation

5) Install OpenStack Dashboard

The OpenStack Dashboard in a web interface for the cloud administrator (http://lcc.louvrex.net:8000/admin) and for the cloud end user (http://lcc.louvrex.net:8000). It is based on the django framework.

The **cloud administrator interface** is used to create users, projects, and manage the user rights.

The **cloud end user interface** is used to create SSH key pairs, to start instances and to create/attach EBS volumes. You cannot (yet) manage the security groups (firewall rules), associate elastic IPs, etc.

Installation procedure:

```
# cd /root
# mkdir src
# cd src
# mkdir openstack-dashboard
# cd openstack-dashboard
# bzr init-repo .
# bzr branch lp:openstack-dashboard trunk
```

You should now have a directory called trunk, which contains the OpenStack Dashboard application.

```
# cd trunk/openstack-dashboard/local
# cp local_settings.py.example local_settings.py
# vi local_settings.py

NOVA_DEFAULT_ENDPOINT = 'http://lcc.louvrex.net:8773/services/Cloud'
NOVA_DEFAULT_REGION = 'nova'
NOVA_ACCESS_KEY = 'xxxxxxxx-7339-4df8-xxxx-30e75f25dd32:project-one'
NOVA_SECRET_KEY = 'xxxxxxxx-4252-4b0d-xxxx-34ca0973285c'
NOVA_ADMIN_USER = 'dodeeric'
```

```
NOVA_PROJECT = 'project-one'

EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
```

Remarks:

- I have installed Postfix (SMTP email server) on the node1. That is why I have configured the Dashboard EMAIL_HOST parameter with localhost.
- The parameters to configure in *local_settings.py* can be found in your novarc credential file:

```
# nova-manage project zipfile project-one dodeeric
# unzip nova.zip
# cat novarc
```

- The NOVA_ADMIN_USER parameter has to be configured with a username who has full cloud admin rights.

```
# apt-get install -y python-setuptools
# easy_install virtualenv
# cd ..
# python tools/install_venv.py ../../../django-nova/trunk
# tools/with_venv.sh dashboard/manage.py syncdb
```

Enter an admin username, e.g.:

```
Username (Leave blank to use 'root'): admin
E-mail address: admin@louvrex.net
Password: 123456
Password (again): 123456
```

Start the server on port tcp/8000 (or another one if you want):

```
# tools/with_venv.sh dashboard/manage.py runserver 0.0.0.0:8000
```

If you want to start the dashboard automatically at boot time:

```
# vi /etc/rc.local

/root/src/openstack-dashboard/trunk/openstack-dashboard/tools/with_venv.sh   /root/src/openstack-dashboard/trunk/openstack-dashboard/da
```

You can now connect on:

- the end user dashboard: http://192.168.1.201:8000 or http://lcc.louvrex.net:8000
- the admin interface (not yet customize; still django look and feel): http://192.168.1.201:8000:admin or http://lcc.louvrex.net:8000:admin

To be able to access the instance's VNC console from the dashboard:

```
# aptitude install nova-vncproxy
# vi /etc/nova/nova.conf
--vnc_enabled=true
--vncproxy_url=http://lcc.louvrex.net:6080
--vnc_console_proxy_url=http://lcc.louvrex.net:6080
```

Install a slightly modified version of noVNC to work with the nova-vnc-proxy:

```
# cd /var/lib/nova
# git clone https://github.com/openstack/noVNC.git
```

At this moment, there is a bug which needs to be corrected:

```
# vi /var/lib/nova/noVNC/vnc_auto.html
```

At line 92:

Before:

```
var host, port, password;
```

After:

```
var host, port, password, token;
```

The browser has to be a websocket enabled browser like **Google Chrome**.

You can now connect to the VNC consoles: from the dashboard, go to the **Instances** tab, then click on the **ID** of the instance you want to connect to, then click on **VNC**.

6) Use the cloud

From node1 or another host on the public/external network:

- Create one user (in this case with full cloud admin rights):

```
# nova-manage user admin dodeeric
```

- Create one project:

```
# nova-manage project create project-one dodeeric "Test project"
```

- Retrieve the credentials:

```
# nova-manage project zipfile project-one dodeeric
# unzip nova.zip
# . novarc
```

- Allow by default ssh (tcp/22) and icmp (ping) for project-one (adapt the default "security group"):

```
# euca-authorize -P icmp -t -1:-1 default
# euca-authorize -P tcp -p 22 default
```

- Create a key pair to be used to access your instances:

```
# euca-add-keypair key-dodeeric > key-dodeeric.priv
# chmod 600 key-dodeeric.priv
```

- Add at least one image (Ubuntu Server 11.04 64 bits):

```
# wget http://uec-images.ubuntu.com/natty/current/natty-server-uec-amd64.tar.gz
# uec-publish-tarball natty-server-uec-amd64.tar.gz bucket-ubuntu
```

Result: ami-00000002

Or see part 7 if you want to **customize your own image**.

- Let's launch our first instance (VM):

```
# euca-run-instances -k key-dodeeric -t m1.medium ami-00000002
```

Result: i-00000001

The **-t** parameter is the **type** of the instance. It is defined like this (but can be changed):

```
# nova-manage instance_type list

m1.tiny:   Memory:   512MB, VCPUS: 1, Storage:   0GB
m1.small:  Memory:  2048MB, VCPUS: 1, Storage:  20GB
m1.medium: Memory:  4096MB, VCPUS: 2, Storage:  40GB
m1.large:  Memory:  8192MB, VCPUS: 4, Storage:  80GB
m1.xlarge: Memory: 16384MB, VCPUS: 8, Storage: 160GB
```

The storage value is the non-persistent local storage (/dev/vdb).

- Let's associate one public/external (floating / elastic) IP to the instance:

```
# euca-allocate-address
```

Result: 192.168.1.240

```
# euca-associate-address -i i-00000001 192.168.1.240
```

- Let's connect to the instance:

```
# ssh -i key-dodeeric.priv ubuntu@192.168.1.240
```

- Let's create an EBS volume of 5 GB and attach it to the instance:

```
# euca-create-volume -s 5 -z nova
```

Result: vol-00000001

```
# euca-attach-volume -i i-00000001 -d /dev/vdc vol-00000001
```

- Let's use that EBS volume from inside the instance:

Check if the volume is seen by the instance:

```
# fdisk -l
```

You should see **/dev/vdc**.

- Let's format and mount that EBS volume:

```
# cfdisk /dev/vdc (create one partition: vdc1)
# mkfs.ext4 /dev/vdc1 (format the partition with the ext4 filesystem)
# mkdir /ebs (create a directory to mount the volume)
# mount /dev/vdc1 /ebs (mount the volume)
```

- There is also a non-persistent local volume (vdb). Let's use it also to add swap and storage:

```
# cfdisk /dev/vdb
# mkfs.ext4 /dev/vdb2
# mount /dev/vdb2 /mnt
# mkswap /dev/vdb1
# swapon /dev/vdb1
```

Important remark: the **EBS volume** (created by nova-volume and attached with the iSCSI protocol) is a **permanent/persistent storage**: when the instance is stopped (killed), that EBS volume will survive. This is not the case of the local volume (vdb).

- You can edit the *fstab* file to make the mount automatic at boot time:

```
# vi /etc/fstab

/dev/vdb1  none  swap  sw,comment=cloudconfig  0  0
```

```
/dev/vdb2  /mnt  auto  defaults,nobootwait,comment=cloudconfig  0  2
/dev/vdc1  /ebs  ext4  nobootwait
```

- To check if you now have swap memory:

```
# free -m
```

7) Customize your own image

Let's customize the latest Linux Ubuntu server image: Natty (11.04) 64 bits.

We will add the following packages:

- Apache/PHP (web server + PHP programing language)
- MySQL (DB server)
- PhpMyAdmin (web interface for MySQL)
- Postfix (SMTP email server)

We will also make some specific configurations.

- Let's download the UEC (Ubuntu Enterprise Cloud / Eucalyptus) image which is compatible with OpenStack:

```
# mkdir custom-image
# cd custom-image
#  wget  http://uec-images.ubuntu.com/natty/current/natty-server-uec-amd64.tar.gz
```

- Untar the file, and delete the non-needed files/images (we only need natty-server-uec-amd64.img):

```
# tar -xzvf natty-server-uec-amd64.tar.gz
# rm natty-server-uec-amd64-vmlinuz-virtual natty-server-uec-amd64-loader natty-server-uec-amd64-floppy README.files natty-server-uec
```

- Rename the image:

```
# mv natty-server-uec-amd64.img dodeeric-lamp-v1-natty-server-uec-amd64.img
```

- We will mount the image and chroot into it to install/configure it:

```
# mkdir mnt
# mount -o loop dodeeric-lamp-v1-natty-server-uec-amd64.img mnt
# mount -o bind /proc mnt/proc
# mount -o bind /sys mnt/sys
# mount -o bind /dev mnt/dev
# chroot mnt
```

- Configure a working DNS server (remove first all the lines in the *resolv.conf* file):

```
# vi /etc/resolv.conf

nameserver 192.168.1.201
```

- Install this mandatory packages:

```
# aptitude install language-pack-en-base
```

- Configure your time zone:

```
# dpkg-reconfigure tzdata
```

- Install the Apache / MySQL / PHP packages (select LAMP):

```
# tasksel
```

- Install and configure Postfix. You can choose the "satellite" configuration and enter a working SMTP relay server (my relay is relay.skynet.be):

```
# aptitude install postfix
```

- Allow to connect with the root username, in place of the ubuntu username:

```
# vi /etc/cloud/cloud.cfg
```

Before:

```
disable_root: 1
```

After:

```
disable_root: 0
```

- Exit the "chrooted" environment:

```
# exit
```

- Let's retrieve the kernel and the ramdisk (rd) images:

```
# cd mnt/boot
# cp vmlinuz-2.6.38-8-virtual ../../
# cp initrd.img-2.6.38-8-virtual ../../
```

- Now we can umount the image:

```
# umount -l mnt
```

- Let's rename the kernel and the ramdisk images into something more clear:

```
# mv vmlinuz-2.6.38-8-virtual natty-server-uec-amd64-kernel-2.6.38-8-virtual
# mv initrd.img-2.6.38-8-virtual natty-server-uec-amd64-ramdisk-2.6.38-8-virtual
```

- Let's bundle, upload, and register the three images. For that you need your credentials:

```
# cp nova.zip /root/custom-image
# unzip nova.zip
# . novarc

# euca-bundle-image -i natty-server-uec-amd64-kernel-2.6.38-8-virtual --kernel true
# euca-upload-bundle -b bucket-ubuntu -m /tmp/natty-server-uec-amd64-kernel-2.6.38-8-virtual.manifest.xml
# euca-register bucket-ubuntu/natty-server-uec-amd64-kernel-2.6.38-8-virtual.manifest.xml
```

Result: aki-00000001

```
# euca-bundle-image -i natty-server-uec-amd64-ramdisk-2.6.38-8-virtual --ramdisk true
# euca-upload-bundle -b bucket-ubuntu -m /tmp/natty-server-uec-amd64-ramdisk-2.6.38-8-virtual.manifest.xml
# euca-register bucket-ubuntu/natty-server-uec-amd64-ramdisk-2.6.38-8-virtual.manifest.xml
```

Result: ari-00000002

```
# euca-bundle-image -i natty-server-uec-amd64.img --kernel aki-00000001 --ramdisk ari-00000002
# euca-upload-bundle -b bucket-ubuntu -m /tmp/natty-server-uec-amd64.img.manifest.xml
# euca-register bucket-ubuntu/natty-server-uec-amd64.img.manifest.xml
```

Result: ami-00000003

You will have to wait some minutes before the ami image is available.

- To check if the new image is available:

```
# euca-describe-images
```

cloud, iaas, nova, openstack