University of Bologna

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

Engineering Bologna Campus

Class of

Computer Networks M

OpenStack Hands-on Lab

Antonio Corradi

Luca Foschini

Michele Solimando

Academic year 2016/2017

OpenStack history in a nutshell

OpenStack

- Founded by **NASA** and **Rackspace** in 2010
- Currently supported by more than **600 companies** (https://www.openstack.org/foundation/companies/) and **74006 people** distributed over the world.
- Latest release: **Ocata**, February 2017

- **Six-month** time-based **release cycle** (aligned with Ubuntu release cycle)
- **Open-source** vs Amazon, Microsoft, Vmware…
- **Constantly growing** project

OpenStack stable branches

The **stable branches** are a safe source of fixes for high impact bugs and security issues of a given release.

Stability is always a trade-off between “bug-free” and “slow-moving”. In order to reach that stability, OpenStack developers community defines several support phases, for which only a limited class of changes are appropriate :

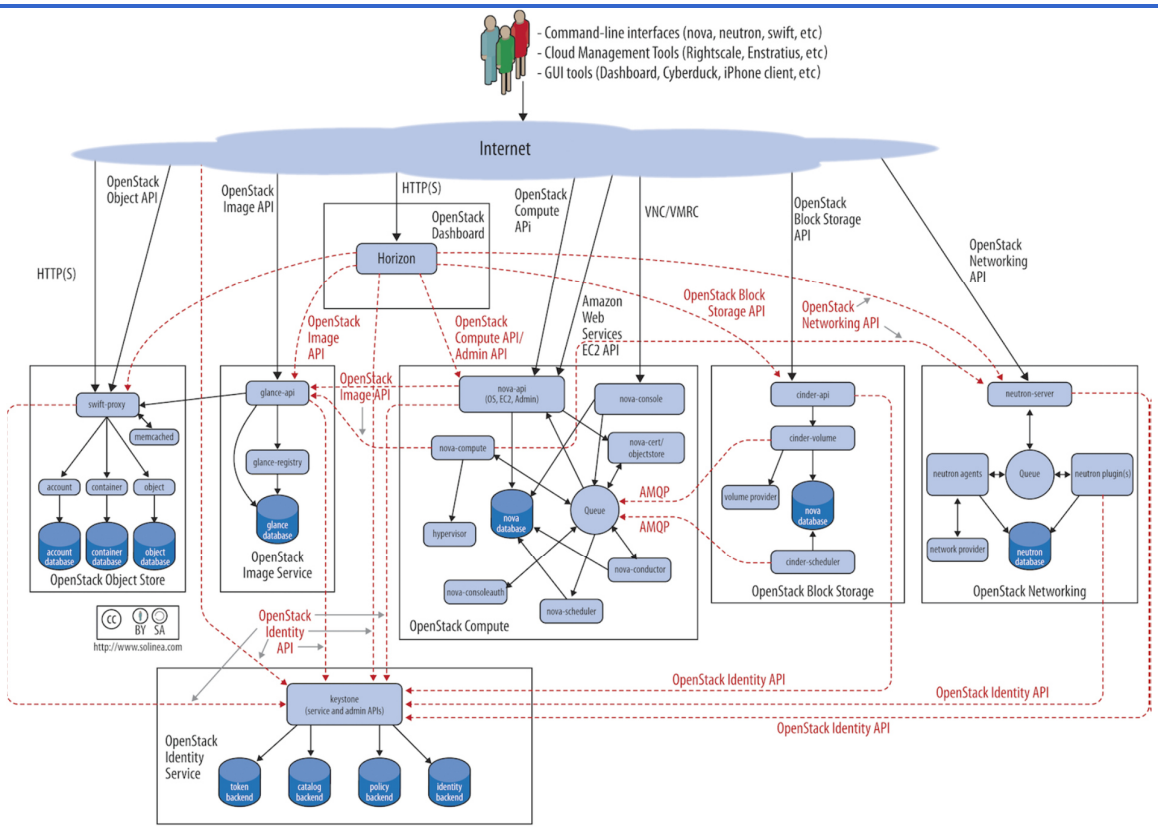
- **Phase I**, *Latest release*: (first 6 months), all bug fixes;
- **Phase II**, *Maintained release*: (6-12 months after release), critical bugfixes and security patches;
- **Phase III**, *Legacy release*: (more than 12 months after release), only security patches.

Only one branch is in Phase I or Phase II support. Depending on how long each branch is supported, there may be one or more releases in Phase III support.

OpenStack overview

- OpenStack is a **cloud operating system** that controls large pools of *compute, storage, and networking* resources throughout a datacenter.
- OpenStack is a collaborative project that involves developers and cloud computing technologists producing the **open standard** cloud computing platform for both public and private clouds. All of the code for OpenStack is freely available under the *Apache 2.0 license*.
- OpenStack has a very large **Community** that provides open discussion spaces for Ask & Question, Mailing List, Blogs, User Groups and many other forms of participation to help the process of development.

OpenStack overall architecture



OpenStack getting started

- **Developer environment**: the goal is “Getting it Done”. Is the environment in which changes to software are developed.
- **Production environment**: the goal is “Keeping it Running”. Software and other products are actually put into operation for their intended uses by end users.

There are many official projects that help us to deploy OpenStack in different ways:

- **OpenStack for Developer**: the main project is **Devstack** (https://docs.openstack.org/developer/devstack/). It includes a series of extensible scripts used to bring up a OpenStack environment. It is used as a development environment and as the basis for much of the OpenStack project’s functional testing.
- **OpenStack for Production**: (*out of the scope of this lesson…*) the main project is **Ansible-OpenStack** (https://github.com/openstack/openstack-ansible). This project aims to deploy production environments from source in a way that makes it scalable while also being simple to operate, upgrade, and grow.
- **OpenStack Sandbox**: (*out of the scope of this lesson…*) the main project is **TryStack** (http://trystack.org/), a totally free OpenStack RDO Liberty installation, on x86 hardware (provided by some of the biggest IT companies). It represents the easiest way for developers to test code against a real OpenStack environment, without having to stand up hardware themselves.
  - **WARNING!!!**: your account on TryStack will be periodically wiped and the lifetime of the virtual servers created by user is limited.

Deployment developer environment

We are going to present the deploy process of the master branch (on May 2017) of the OpenStack main environment.

Versions:

- System version: Ubuntu **16.04.2** LTS.
- OpenStack version: **Ocata**, 2017.
  (https://releases.openstack.org/ocata/index.html)
- Tool for deploy: **DevStack** 0.0.1.dev8456.

DevStack download

To quickly build dev OpenStack environments in a clean Ubuntu environment (https://docs.openstack.org/developer/devstack/).

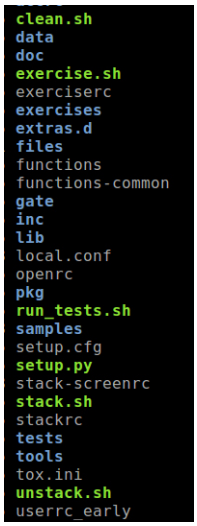
```
$ git clone https://git.openstack.org/openstack-dev/devstack
```

The DevStack **master branch** generally points to trunk versions of OpenStack components. For older, stable versions, look for branches named stable/[release] in the DevStack repo. For example, you can do the following to create a Newton OpenStack cloud:

```
$ cd devstack/
$ git branch -a                 #show the available branches and
                                      underline the current one.
$ git checkout stable/newton
```

DevStack file system

The main folder of DevStack contains all bash scripts and configuration files needed for the installation.



- **stack.sh**: script to run (NOT AS ROOT!) to install a new cloud deployment. This script reads the directives contained in the *local.conf* file.
- **unstack.sh**: stops all cloud services and virtual machines. To run before rebooting the system.
- **clean.sh**: executes *unstack.sh* and also deletes all the configurations. Useful to completely remove the cloud services.
- Folder **samples/**: contains a minimal sample of the configuration file, *local.conf*.
- **local.conf**: has a main role in installation process because give all the installation directives for all the OpenStack's components.
- **stack-screenrc**: automatically created after a successful installation. It contains a list of installed services and related processes. Useful to restart the cloud modules.
- **openrc**: configures a set of credentials to use OpenStack command line interface.

stack.sh

```
$ /stack.sh
```

The script executes the following steps based on information contained in *local.conf*:

- Downloads and sets up the **OpenStack components** from git;
- Downloads and sets up the **tools and the dependencies** of the OpenStack environment, such as MySQL, RabbitMQ, Open vSwitch, etc…;
- Creates base configuration within **OpenStack environment**: creates two example projects, an administrator user, a basic network and related subnet, a virtualized router; downloads the cloud base image of Cirros OS.

Test case architecture 1/2

We are testing a multi-node installation. On every node there is a local.conf file that specifies the desired configuration for the host, and every node has multiple physical interfaces.

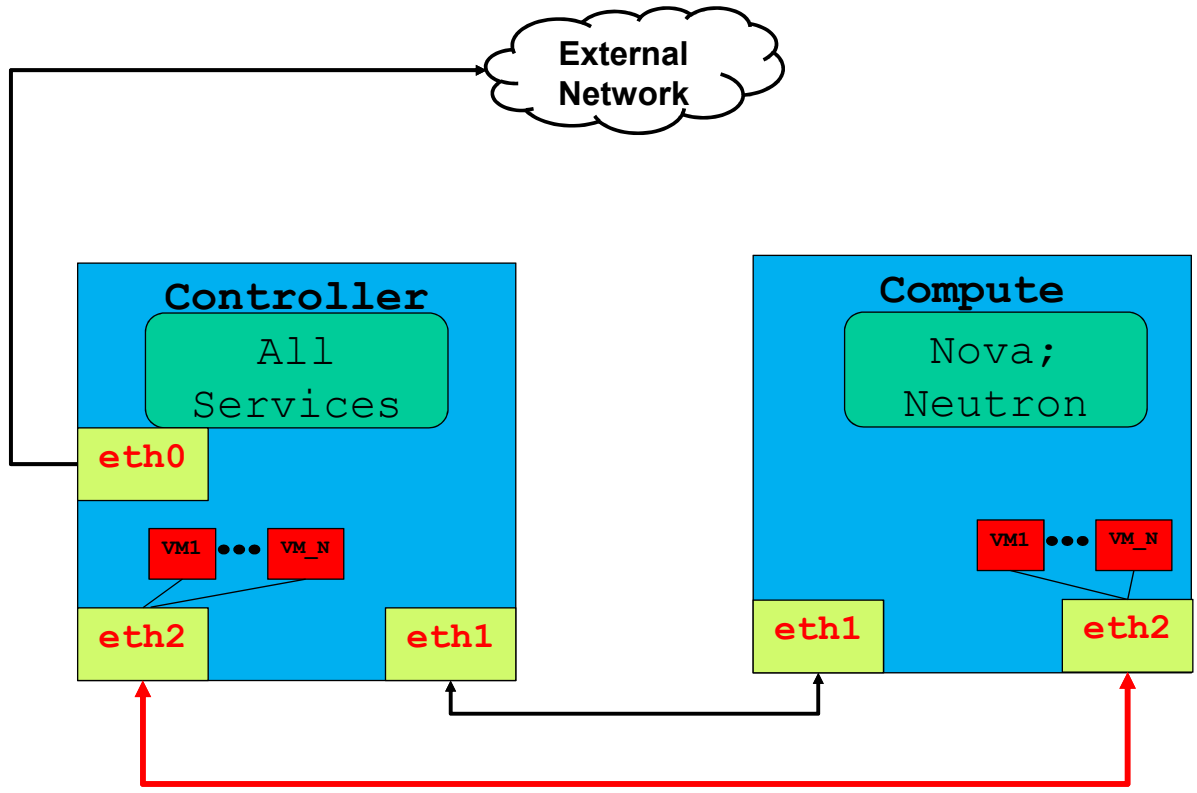
Our test case:

- One ***Controller*** Node: runs all the services needed to your cluster: compute service, networking service, storage services, etc… In our case, it is also a compute node.
- One ***Compute*** Node: runs the nova-compute service, this is where virtual instances actually run, and part of the network service.

Our network:

- The *Controller node* has three physical interfaces: the first (**eth0**) is the interface that is connected to the external network; the second (**eth1**) connects the cluster nodes; the third (**eth2**) is that forwards the VM traffic to the external network (it is added to a bridge with the first interface).
- The *Compute node* has only two physical interfaces: the first (**eth1**) to connect to the other nodes and the second (**eth2**) for the VM traffic.

Test case architecture 2/2



local.conf Controller

```
[[local|localrc]]
ADMIN_PASSWORD=nomoresecret
DATABASE_PASSWORD=stackdb
RABBIT_PASSWORD=stackqueue
SERVICE_PASSWORD=$ADMIN_PASSWORD

HOST_IP=172.18.161.6
SERVICE_HOST=172.18.161.6
MYSQL_HOST=172.18.161.6
RABBIT_HOST=172.18.161.6
GLANCE_HOSTPORT=172.18.161.6:9292

# Select services to be run
DISABLE_SERVICES tempest n-obj n-net n-vol
ENABLED_SERVICES+=,q-svc,q-dhcp,q-meta,q-agt,q-l3

# Neutron options
Q_USE_SECGROUP=True
FLOATING_RANGE="172.18.161.0/24"
IPV4_ADDRS_SAFE_TO_USE=10.0.0.0/24
Q_FLOATING_ALLOCATION_POOL=start=172.18.161.250,end=172.18.161.254
PUBLIC_NETWORK_GATEWAY="172.18.161.1"
PUBLIC_INTERFACE=eth1
```

This is the minimum required configuration to get started with DevStack, in case of single node installation. The **pre-set passwords** prevent interactive prompts during *stack.sh*.

- **HOST_IP** = Sets the API endpoint.
- ***_HOST** = Indicate the endpoints address of the services.

- **Q_USE_SECGROUP** = Enable security groups.
- **FLOATING_RANGE** = is a range not used on the local network and represents the public network.
- **IPV4_ADDRS_SAFE_TO_USE** = configures the internal address space used by the instances. Virtual machines are always given an internal IP address from the *IPV4_ADDRS_SAFE_TO_USE.*
- **Q_FLOATING_ALLOCATION_POOL** = explicitly set the pool of IPs used for instances.

local.conf Compute

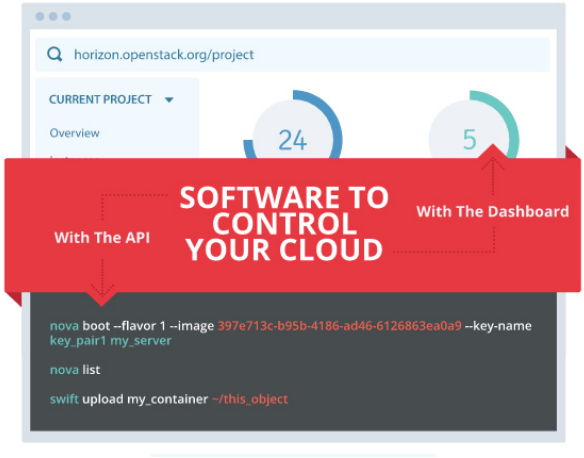
```
[[local|localrc]]
HOST_IP=172.18.161.7
SERVICE_HOST=172.18.161.6
MYSQL_HOST=172.18.161.6
RABBIT_HOST=172.18.161.6
GLANCE_HOSTPORT=172.18.161.6:9292
ADMIN_PASSWORD=nomoresecret
DATABASE_PASSWORD=stackdb
RABBIT_PASSWORD=stackqueue
SERVICE_PASSWORD=$ADMIN_PASSWORD

## Neutron options
PUBLIC_INTERFACE=eth0
ENABLED_SERVICES=n-cpu,rabbit,q-agt
```

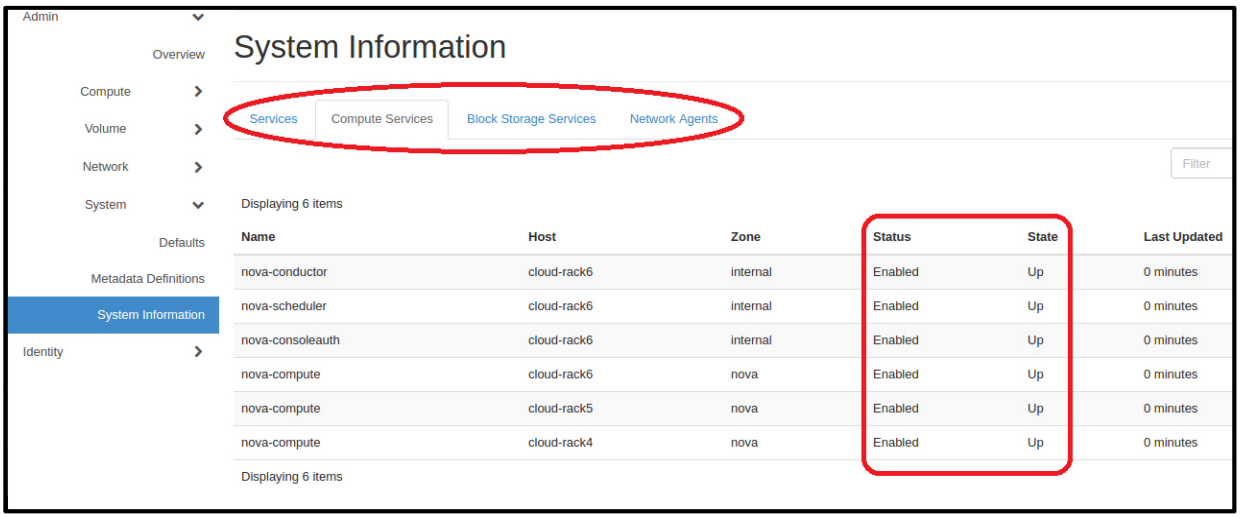
On a compute node only few services are running and for this it has a minimal local.conf. Network traffic from the compute nodes is then NAT'd by the controller node that runs Neutron's neutron-l3-agent and provides L3 connectivity.

Administration of the cluster 1/4

For almost all OpenStack operations, we have two main way to act: **dashboard or command line clients**. Also if we do not have a DevStack installation.



Administration of the cluster 2/4



To monitor our installation, included the status of all services, we can act alternatively from dashboard or from CLI.
If we use the Dashboard we can see the status under the tab Admin → System → System Information.

Administration of the cluster 3/4

Alternatively we can check status of the system using the CLI. For example to check the health of nova services we can type:

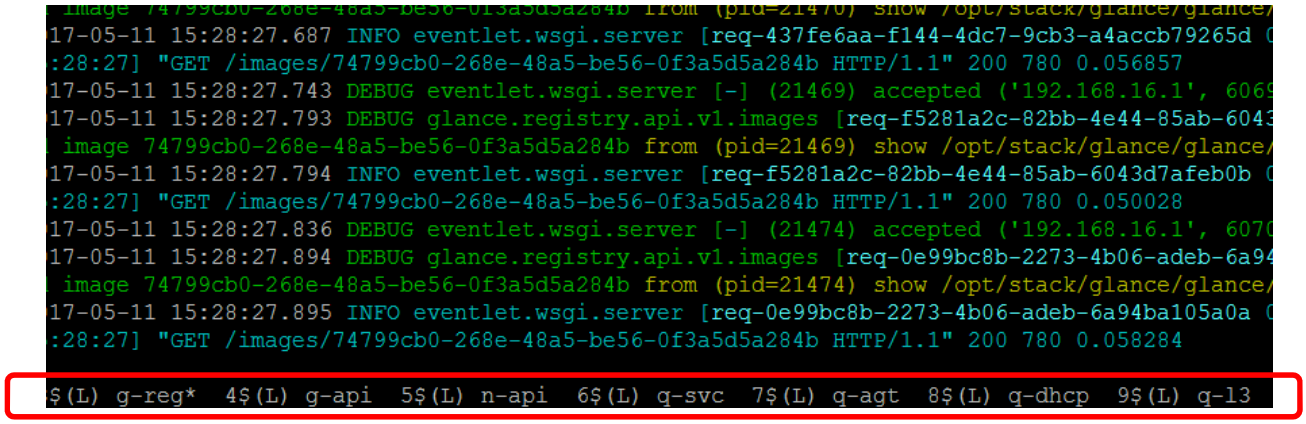
```
$ nova service-list
```

```
+----+------------------+-------------+----------+---------+-------+
| Id | Binary           | Host        | Zone     | Status  | State |
+----+------------------+-------------+----------+---------+-------+
| 3  | nova-conductor   | cloud-rack6 | internal | enabled | up    |
| 6  | nova-scheduler   | cloud-rack6 | internal | enabled | up    |
| 7  | nova-consoleauth | cloud-rack6 | internal | enabled | up    |
| 8  | nova-compute     | cloud-rack6 | nova     | enabled | up    |
| 9  | nova-compute     | cloud-rack5 | nova     | enabled | up    |
| 10 | nova-compute     | cloud-rack4 | nova     | enabled | up    |
```

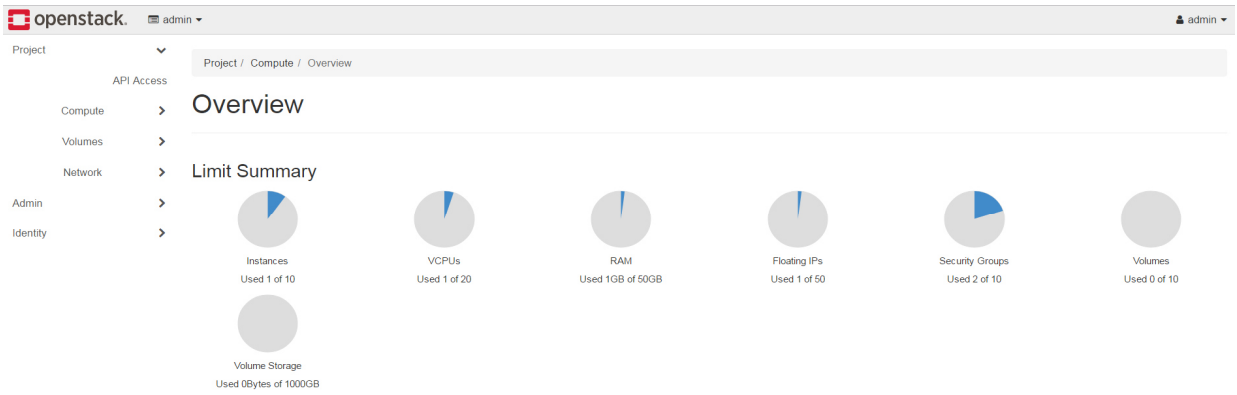
To use this approach we have to authenticate ourselves via Keystone, the next slides show how it is possible using the command line.

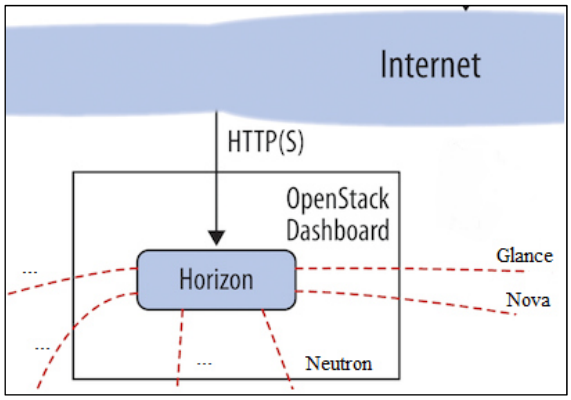
Administration of the cluster 4/4

Every OpenStack component has its own **Log file**. The log file contains the output messages produced by the system events about that component. DevStack opens in continuous **tail** all the Log files, each of them in a separate **screen**. See `man tail` and `man screen` for information.



Dashboard (*HORIZON*)

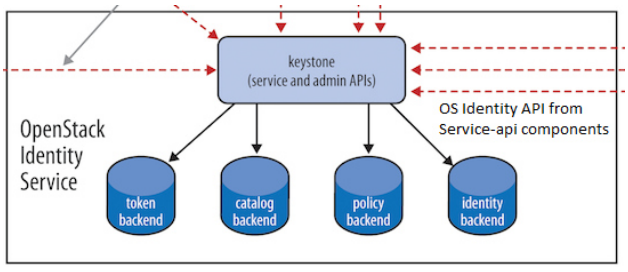


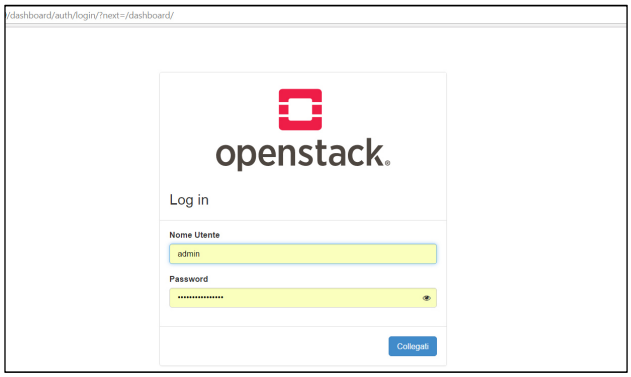


After the successful execution of *stack.sh* script, we can go to the home page of OpenStack.

From the dashboard it is possible to control all the services of our cloud.

Authentication (*KEYSTONE*)



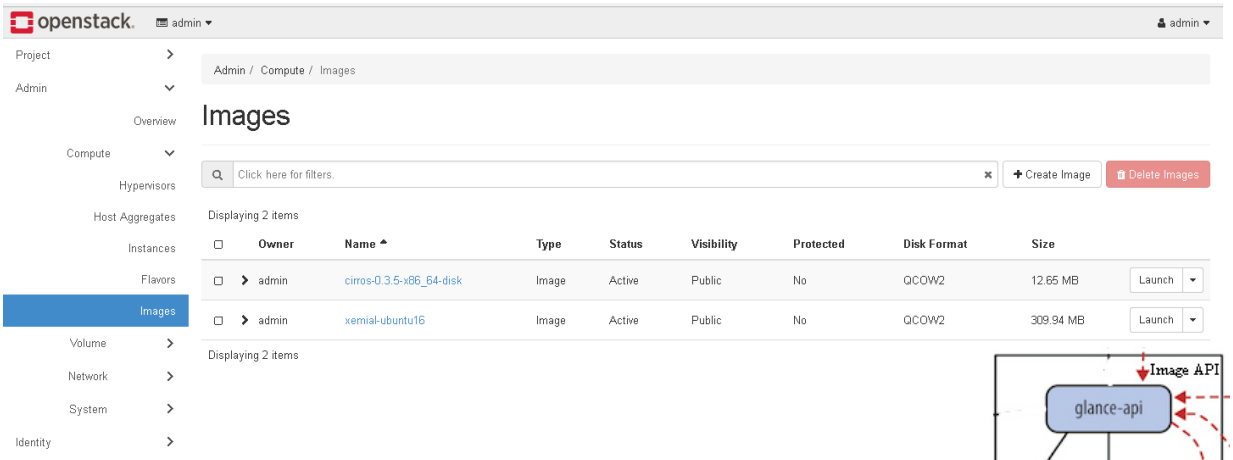


Authentication is possible in both way: dashboard and from CLI:

- **Dashboard Auth**: we can use the *ADMIN_PASSWORD* of the *local.conf* file;
- **CLI Auth**: we can use the *openrc* to source the preconfigured environment variables.

```
~/devstack$ source openrc admin admin
...
$ printenv | grep OS_
OS_REGION_NAME=RegionOne
OS_PROJECT_NAME=demo
OS_IDENTITY_API_VERSION=2.0
OS_PASSWORD=nomoresecret
OS_AUTH_URL=http://x.x.x.x:5000/v2.0
OS_USERNAME=admin
OS_TENANT_NAME=admin
OS_VOLUME_API_VERSION=2
OS_NO_CACHE=1
```

Image service (*GLANCE*) 1/2



Before creation of a virtual machine, we have to create a base image of an operative system.
There are many cloud image of the main operating systems: e.g. https://cloud-images.ubuntu.com/.

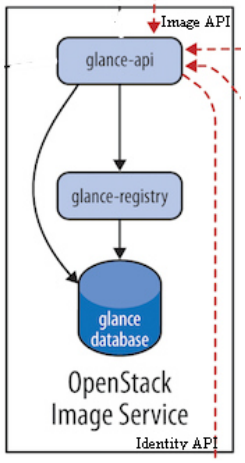


Image service (*GLANCE*) 2/2

```
$ glance image-create --name "NAME" \
        --is-public IS_PUBLIC \
        --disk-format DISK_FORMAT \
        --container-format CONTAINER_FORMAT \
        --file IMAGE
```

We also can create a Glance Image from a downloaded file representing the base image of an operating system. We have to provide:

- **NAME** = to refer to the disk image by.
- **IS_PUBLIC** = *true* means that all users will be able to view and use the image.
- **DISK_FORMAT** = format of the virtual machine disk image. Valid values include raw, vhd, vmdk, vdi, iso, *qcow2*, aki,, and ami.
- **CONTAINER_FORMAT** = container format of the image.
- **IMAGE** = local path to the image file to upload.

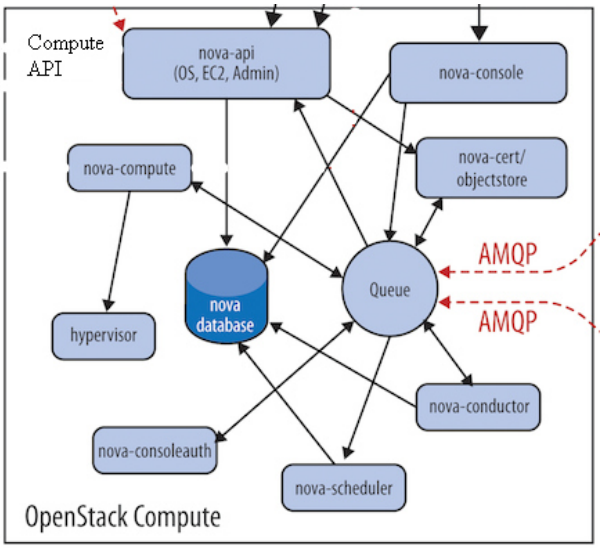
It is possible to check the images with the following command:

```
$ glance image-list
```

To show many informations about an image use:

```
$ glance image-show <image_id>
```

Compute service (*NOVA*) 1/3



From an image we can start a new Virtual Server. We have to specify some mandatory parameters, such as the *Name* of the new instance, the *Network* and the amount of virtualized resources (the *Flavor*).

Compute service (*NOVA*) 2/3

Launch Instance

Details
Source
Flavor *
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name *
CorbaInstance

Availability Zone
nova

Count *
1

Total Instances (10 Max)

20%

1 Current Usage
1 Added
8 Remaining

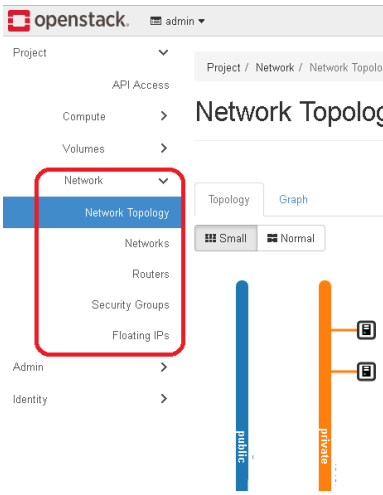
Compute service (*NOVA*) 3/3

To correctly spawn a virtual server, we can use the CLI:

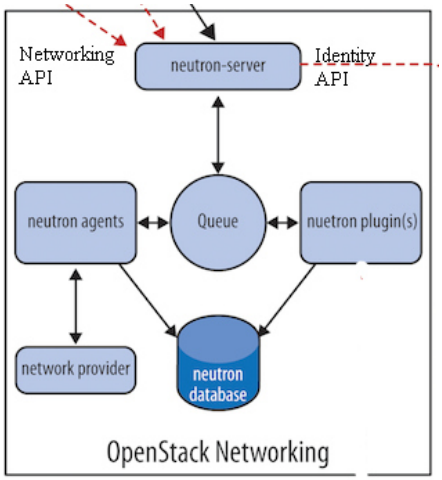
```
$ openstack server create --flavor <flavorName> \
      --image <imageId> \
      --nic net-id=<netId> \
      --security-group <secName> \
      --availability-zone <zone>:<host> \
      --key-name <keyname> \
      <VM_NAME>
```

We can retrieve the elements required by the command, listing the resources of the cluster and choosing the proper one.

Advanced Networking service (*NEUTRON*) 1/3



Neutron module gives us the possibility to virtualize the main network elements: the network itself, the routers, the security groups, the dhcp, etc…



Advanced Networking service (*NEUTRON*) 2/3

In order to connect to an instance, we have to create a network and a security group and add a rule that open the port for the connection (e.g. SSH on port 22). We have also to create a keypair that we'll ask that the public key be put in the VM so we can SSH into it.
By default, DevStack creates networks called private and public. Run the following command to see the existing networks:

```
$ openstack network list
```

- To create a new network we can use:

```
$ openstack network create
```

- To create a new keypair:

```
$ openstack keypair create demo > id_rsa_demo
$ chmod 600 id_rsa_demo
```


Advanced Networking service (*NEUTRON*) 3/3

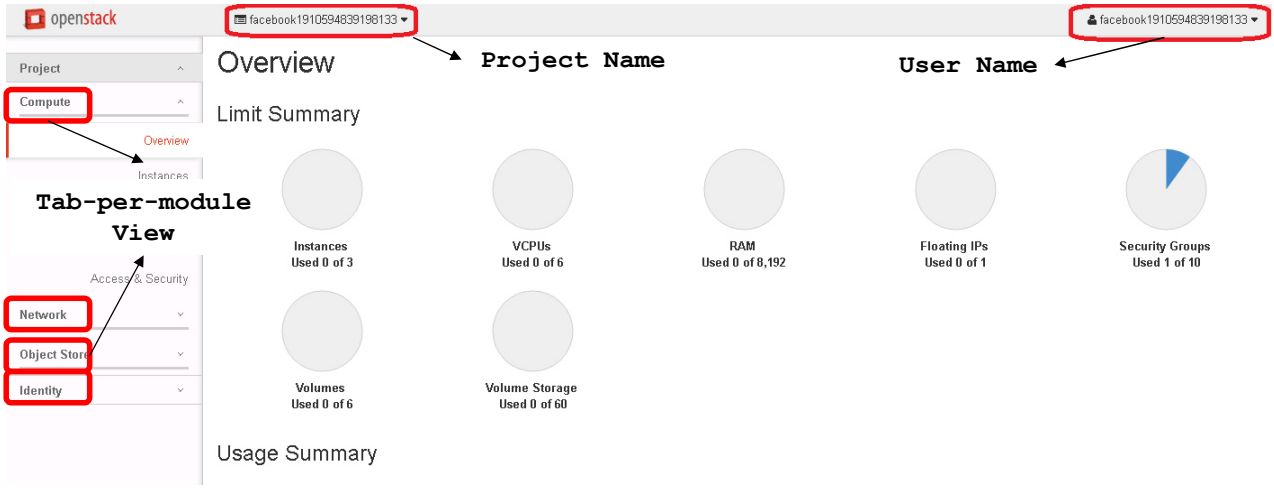
To enable ICMP and SSH communication with the VMs we have to create two different rule in Default security group, created by DevStack during the installation process:

```
$ openstack security group rule create --ingress \
      --ethertype IPv4 --dst-port 22 \
      --protocol tcp default
$ openstack security group rule create --ingress \
      --ethertype IPv4 --protocol ICMP default
```

The VMs inside this security group will have opened port ICMP and SSH.

TryStack

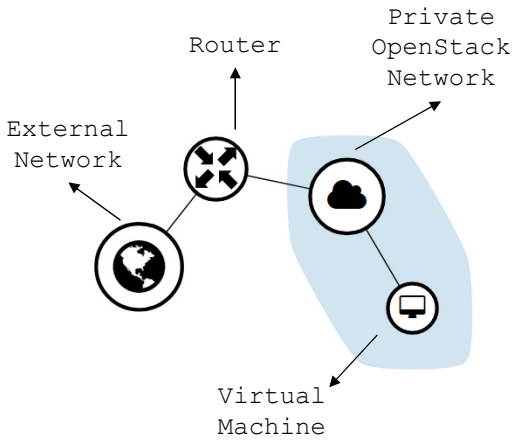
The Easiest Way To Try Out OpenStack



We want test basic OpenStack functionalities in a sandbox environment. Once signed up to the TryStack site via Facebook, we will have a private user and project.

Set up the network 1/2

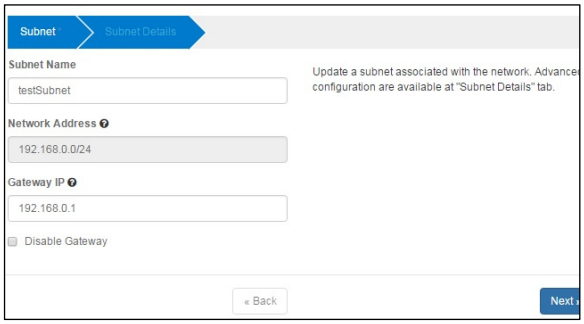
We want to obtain this topology of network:

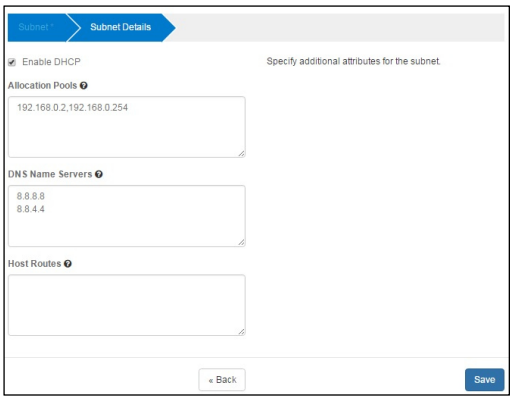


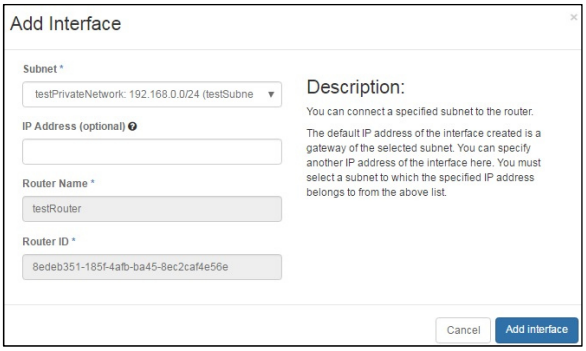
Set up the network 2/2

From the Dashboard we can:

1. Create a private virtual network;
2. Create a virtual subnet inside the network;
3. Create a router connected to the public network;
4. Connect our network to the router, adding a new interface.



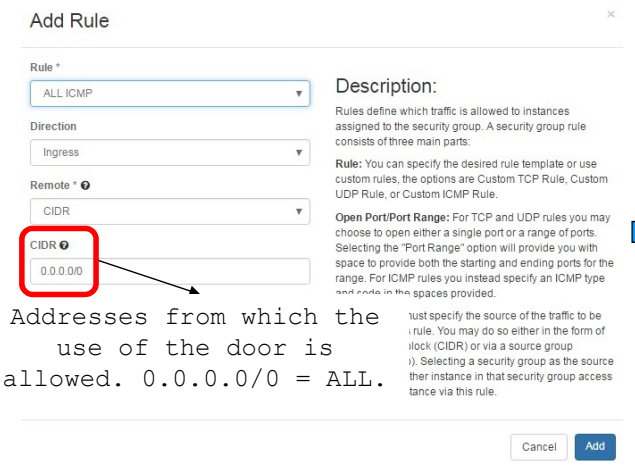




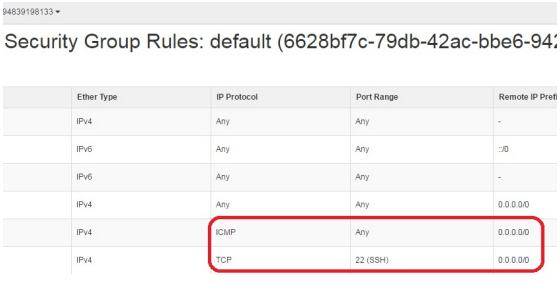
Security setup

In order to connect to a new instance:

1. Create a keypair and download the private key;
   WARNING: due to the security policy the server does not store the private keys of the end user!!! This is the only time you can download the private key.
2. Add **ICMP** and **SSH** rules to the *default* security group.



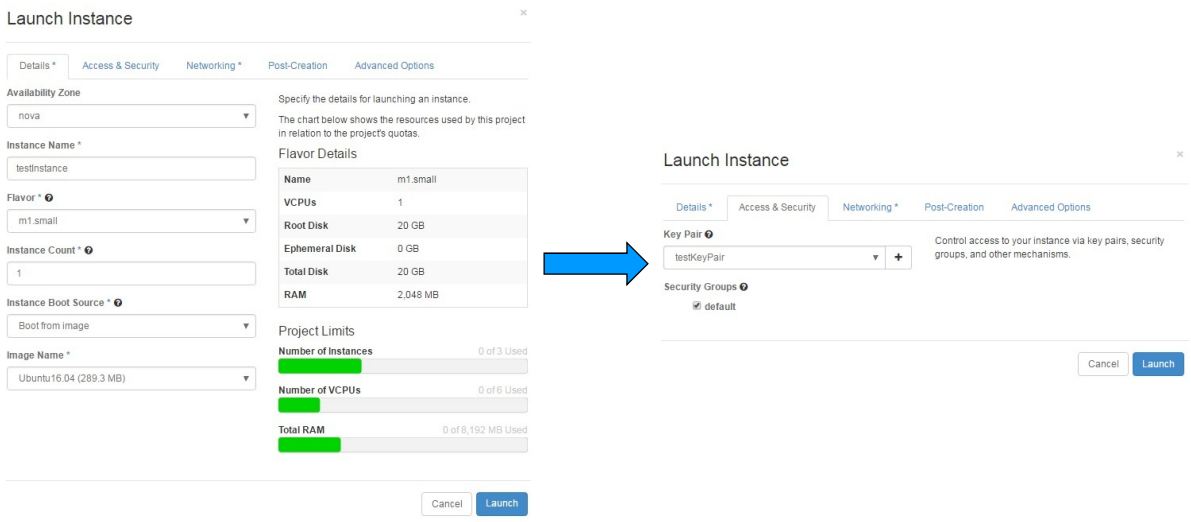
Addresses from which the use of the door is allowed. 0.0.0.0/0 = ALL.



Starting a virtual Server

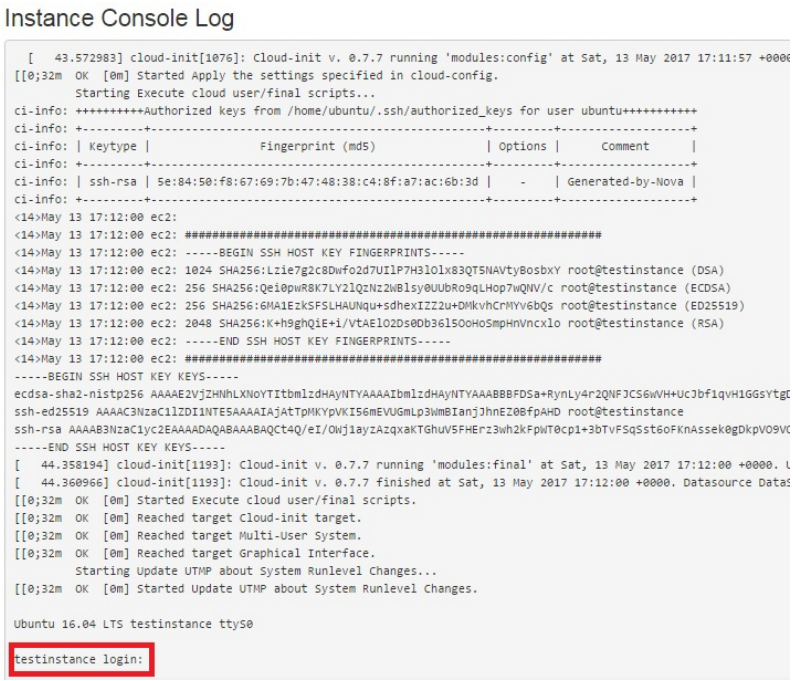
It is the time to launch our virtual server:

1. Choose an *Image*, a *Flavor*, an *Availability Zone*;
2. Choose a *Key Pair*, a *Security Group*;
3. Choose a *Network* and click on Launch button.



Starting a virtual Server

Inspect the instance **Log** and interact via **Console**.



TryStack

The Easiest Way To Try Out OpenStack

TEST1 (*Nova*): Launch a virtual machine.

TEST2 (*Network*): Create a virtual router.

TEST3 (*Network*): Create a network and add it to the router.

TEST4 (*Cinder*): Create a Volume and a virtual server with this volume attached.

TEST5 (…): Try at home!!! ☺

References

OpenStack Docs: https://docs.openstack.org/

OpenStack Slides: http://lia.deis.unibo.it/Courses/CompNetworksM/1617/slides/Openstackx2.pdf

DevStack Docs: https://docs.openstack.org/developer/devstack/

DevStack Git: https://github.com/openstack-dev/devstack

DevStack+Neutron tutorial: https://docs.openstack.org/developer/devstack/guides/neutron.html

TryStack: http://trystack.org/

Thanks to All ☺