



University of Bologna

**Dipartimento di Informatica –
Scienza e Ingegneria (DISI)**

Engineering Bologna Campus

Class of

Computer Networks M

OpenStack and Amazon Web Services

Antonio Corradi

Luca Foschini

Michele Solimando

Academic year 2017/2018

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: **Queens**, February 2018
- **Six-month** time-based **release cycle** (aligned with Ubuntu release cycle)
- **Open-source** vs Amazon, Microsoft, Vmware...
- **Constantly growing** project



openstack™
CLOUD SOFTWARE

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 :

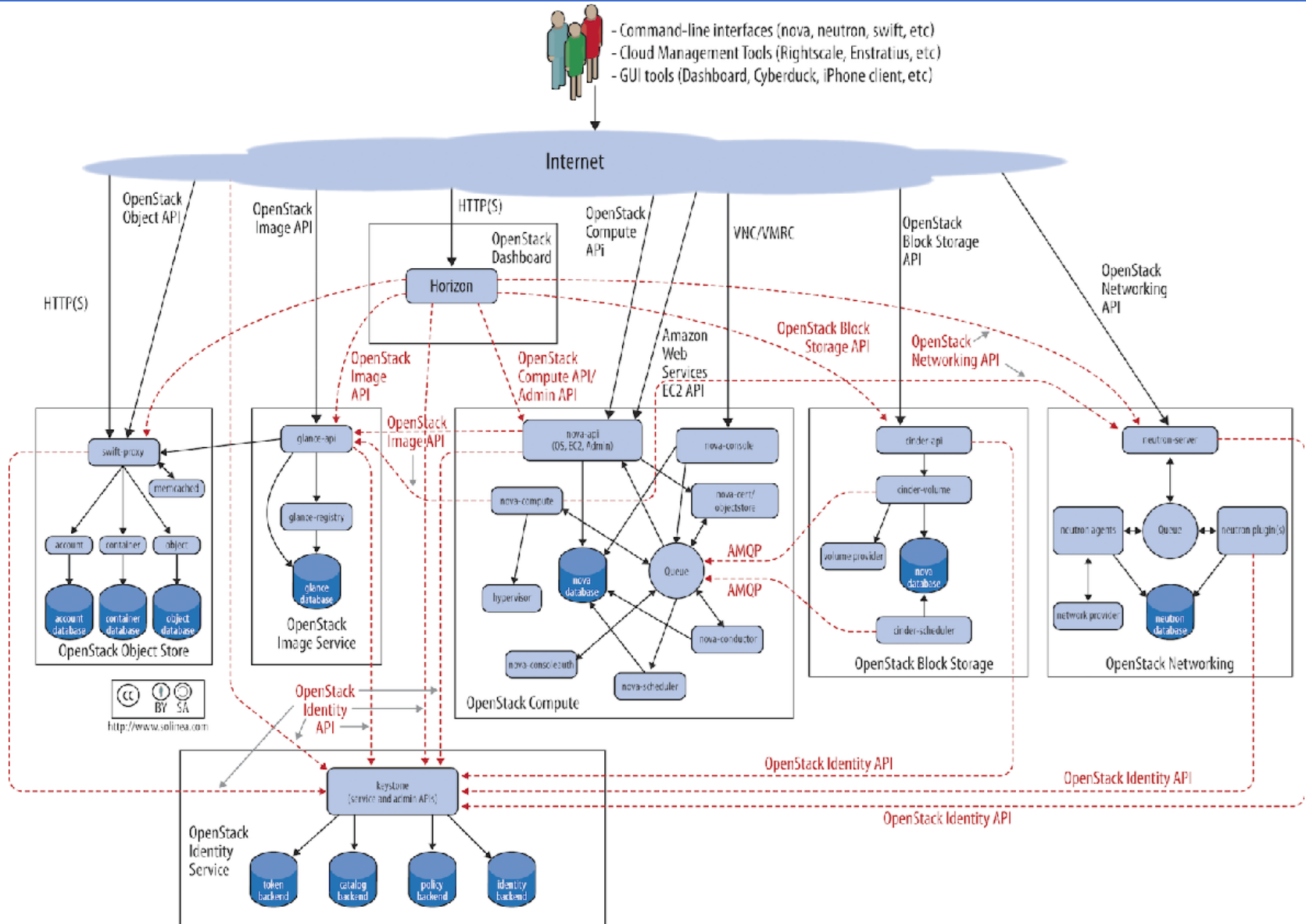
- **Phase I**, *Latest release*: (first 6 months), all bug fixes;
- **Phase II**, *Maintained release*: (6-12 months after release), critical bugfixes and security patches;
- **Phase III**, *Legacy release*: (more than 12 months after release), only security patches.

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 start

- **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...*) **RDO** (RPM Distribution of OpenStack) (<https://www.rdoproject.org/>) is a community focused on packaging and integrating code from the upstream OpenStack project on CentOS, Red Hat Enterprise Linux and Fedora-based platforms.

DevStack deploy

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 the bash scripts and configuration files useful for the installation.

```
clean.sh
data
doc
exercise.sh
exerciserc
exercises
extras.d
files
functions
functions-common
gate
inc
lib
local.conf
openrc
pkg
run_tests.sh
samples
setup.cfg
setup.py
stack-screenrc
stack.sh
stackrc
tests
tools
tox.ini
unstack.sh
userrc_early
```

- **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 informations 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.

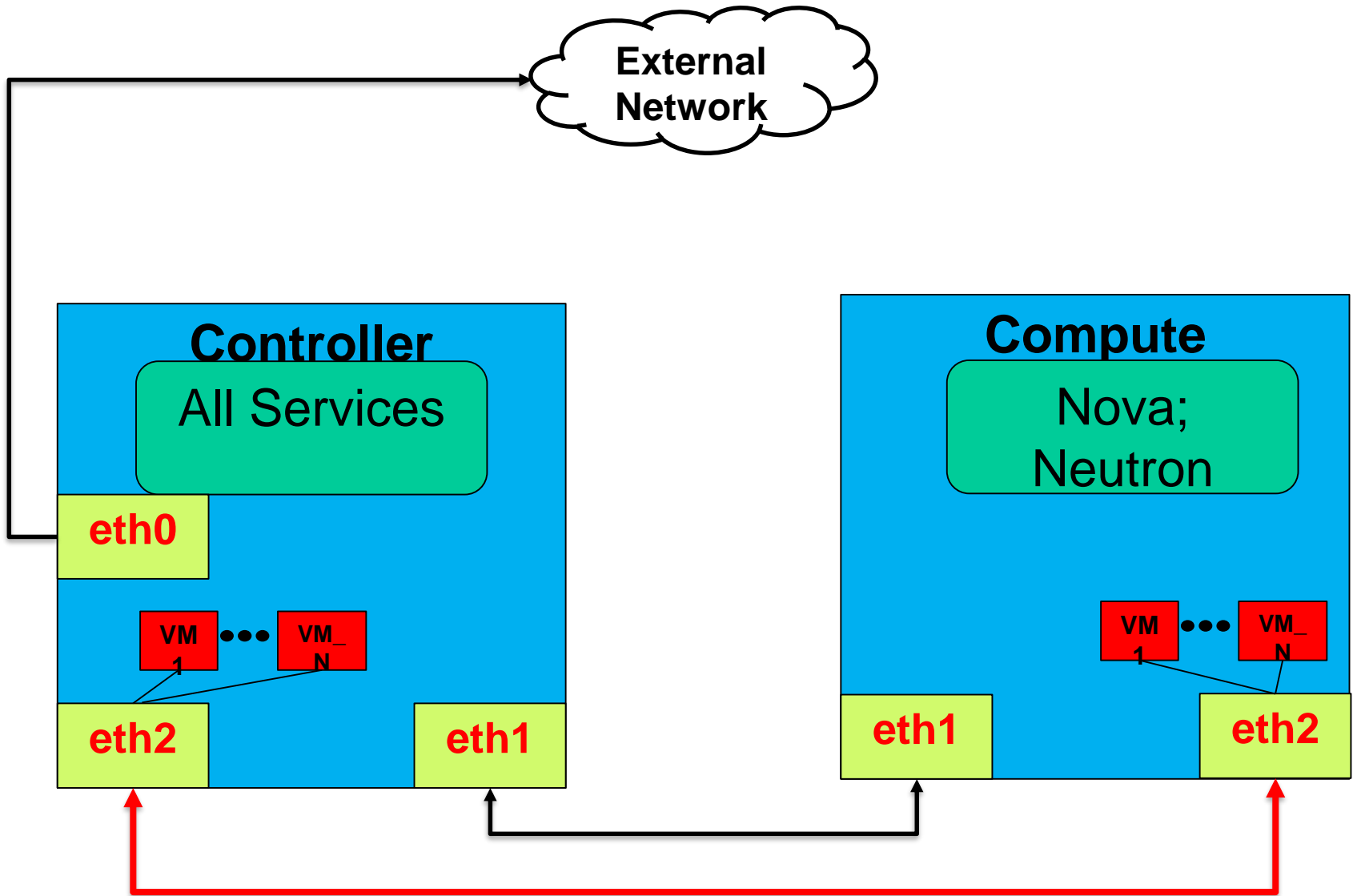
Our test case:

- 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=172.18.1
61.250,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 (in our scenario, is a different physical host!) only few services are running and for this it has a very 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.

The image shows a screenshot of the OpenStack Horizon dashboard. The browser address bar displays 'horizon.openstack.org/project'. Below the search bar, there is a 'CURRENT PROJECT' dropdown menu and an 'Overview' link. Two circular progress indicators are visible, showing values 24 and 5. A large red banner is overlaid on the dashboard, containing the text 'SOFTWARE TO CONTROL YOUR CLOUD' in white. To the left of the banner, it says 'With The API' with a dashed arrow pointing down to a terminal window. To the right, it says 'With The Dashboard' with a dashed arrow pointing up to the dashboard area. The terminal window shows the following commands:

```
nova boot --flavor 1 --image 397e713c-b95b-4186-ad46-6126863ea0a9 --key-name key_pair1 my_server  
  
nova list  
  
swift upload my_container ~/this_object
```

Administration of the cluster 2/4

The screenshot shows the OpenStack Admin dashboard. The left sidebar contains navigation options: Admin, Overview, Compute, Volume, Network, System, Defaults, Metadata Definitions, System Information (highlighted), and Identity. The main content area is titled 'System Information' and has four tabs: Services (highlighted in red), Compute Services, Block Storage Services, and Network Agents. Below the tabs, it says 'Displaying 6 items'. A table lists the services with columns: Name, Host, Zone, Status, State, and Last Updated. The 'Status' and 'State' columns are highlighted in red. A 'Filter' button is visible on the right.

Name	Host	Zone	Status	State	Last Updated
nova-conductor	cloud-rack6	internal	Enabled	Up	0 minutes
nova-scheduler	cloud-rack6	internal	Enabled	Up	0 minutes
nova-consoleauth	cloud-rack6	internal	Enabled	Up	0 minutes
nova-compute	cloud-rack6	nova	Enabled	Up	0 minutes
nova-compute	cloud-rack5	nova	Enabled	Up	0 minutes
nova-compute	cloud-rack4	nova	Enabled	Up	0 minutes

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.

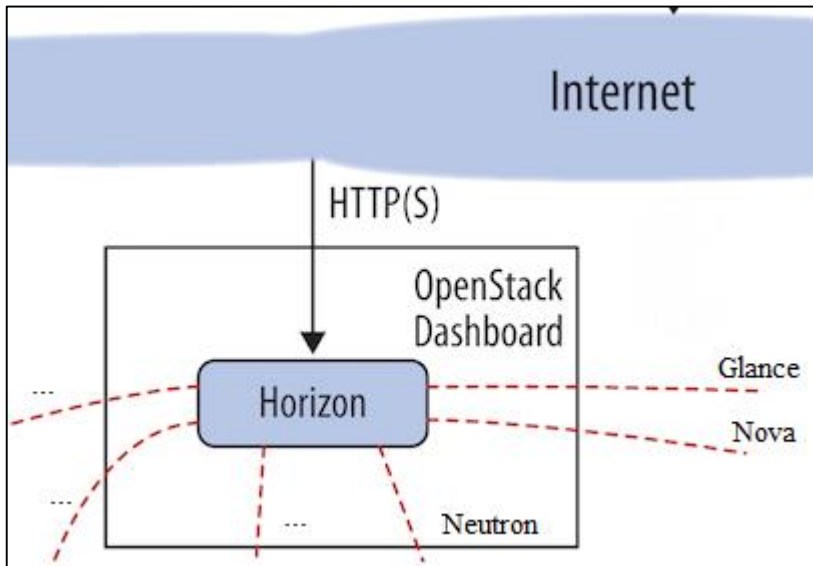
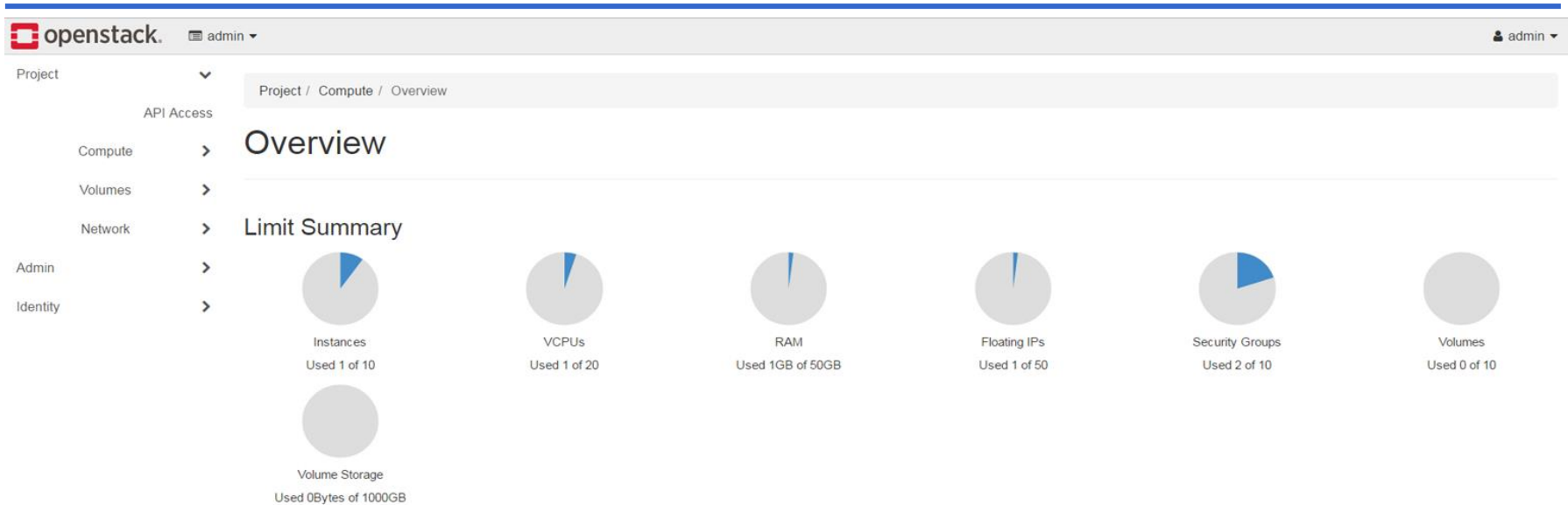
Administration of the cluster 4/4

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.

```
image 74799cb0-268e-48a5-be56-0f3a5d5a284b from (pid=21470) show /opt/stack/glance/glance/
17-05-11 15:28:27.687 INFO eventlet.wsgi.server [req-437fe6aa-f144-4dc7-9cb3-a4accb79265d 0
:28:27] "GET /images/74799cb0-268e-48a5-be56-0f3a5d5a284b HTTP/1.1" 200 780 0.056857
17-05-11 15:28:27.743 DEBUG eventlet.wsgi.server [-] (21469) accepted ('192.168.16.1', 6069
17-05-11 15:28:27.793 DEBUG glance.registry.api.v1.images [req-f5281a2c-82bb-4e44-85ab-6043
image 74799cb0-268e-48a5-be56-0f3a5d5a284b from (pid=21469) show /opt/stack/glance/glance/
17-05-11 15:28:27.794 INFO eventlet.wsgi.server [req-f5281a2c-82bb-4e44-85ab-6043d7afeb0b 0
:28:27] "GET /images/74799cb0-268e-48a5-be56-0f3a5d5a284b HTTP/1.1" 200 780 0.050028
17-05-11 15:28:27.836 DEBUG eventlet.wsgi.server [-] (21474) accepted ('192.168.16.1', 6070
17-05-11 15:28:27.894 DEBUG glance.registry.api.v1.images [req-0e99bc8b-2273-4b06-adeb-6a94
image 74799cb0-268e-48a5-be56-0f3a5d5a284b from (pid=21474) show /opt/stack/glance/glance/
17-05-11 15:28:27.895 INFO eventlet.wsgi.server [req-0e99bc8b-2273-4b06-adeb-6a94ba105a0a 0
:28:27] "GET /images/74799cb0-268e-48a5-be56-0f3a5d5a284b HTTP/1.1" 200 780 0.058284

$(L) g-reg* 4$(L) g-api 5$(L) n-api 6$(L) q-svc 7$(L) q-agt 8$(L) q-dhcp 9$(L) q-13
```

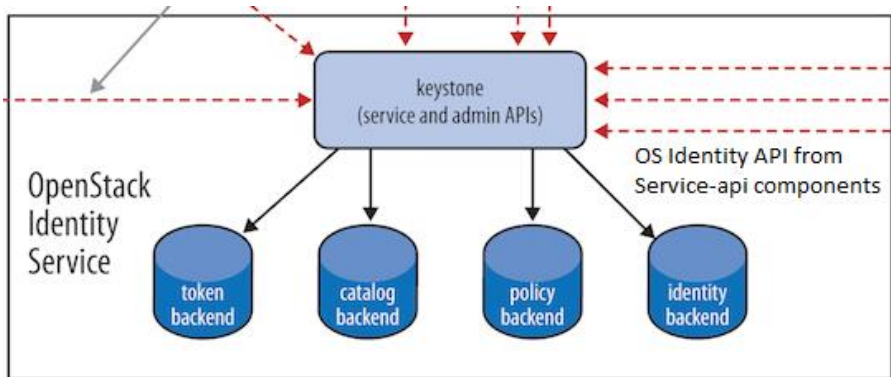
Dashboard (*HORIZON*)



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
```

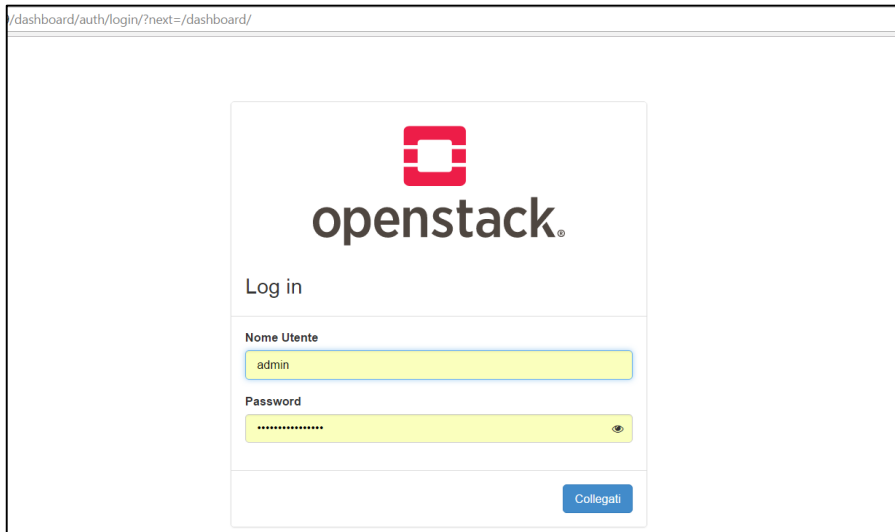


Image service (*GLANCE*) 1/2

openstack. admin

Admin / Compute / Images

Images

Click here for filters. + Create Image Delete Images

Displaying 2 items

<input type="checkbox"/>	Owner	Name	Type	Status	Visibility	Protected	Disk Format	Size	
<input type="checkbox"/>	admin	cirros-0.3.5-x86_64-disk	Image	Active	Public	No	QCOW2	12.65 MB	Launch
<input type="checkbox"/>	admin	xemial-ubuntu16	Image	Active	Public	No	QCOW2	309.94 MB	Launch

Displaying 2 items

Volume

Network

System

Identity

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/>.

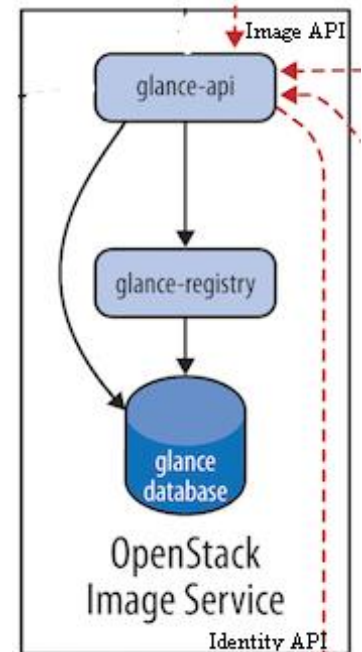


Image service (*GLANCE*) 2/2

```
$ 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.

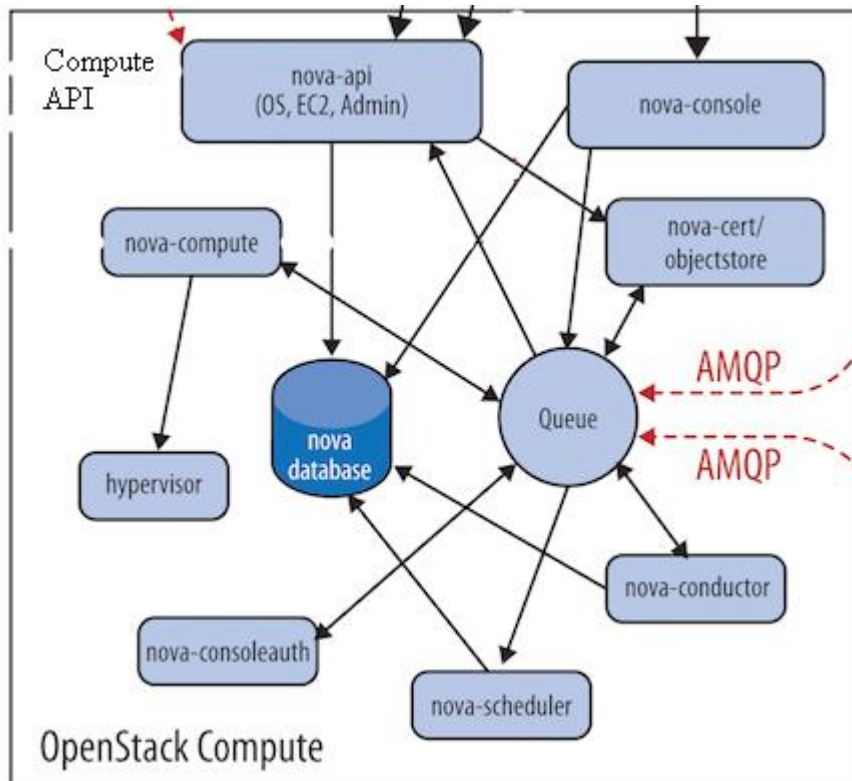
It is possible to check the images with the following command:

```
$ 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 *

CorbalInstance

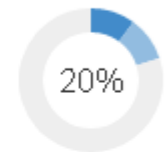
Availability Zone

nova

Count *

1

Total Instances (10 Max)



■ 1 Current Usage
■ 1 Added
■ 8 Remaining

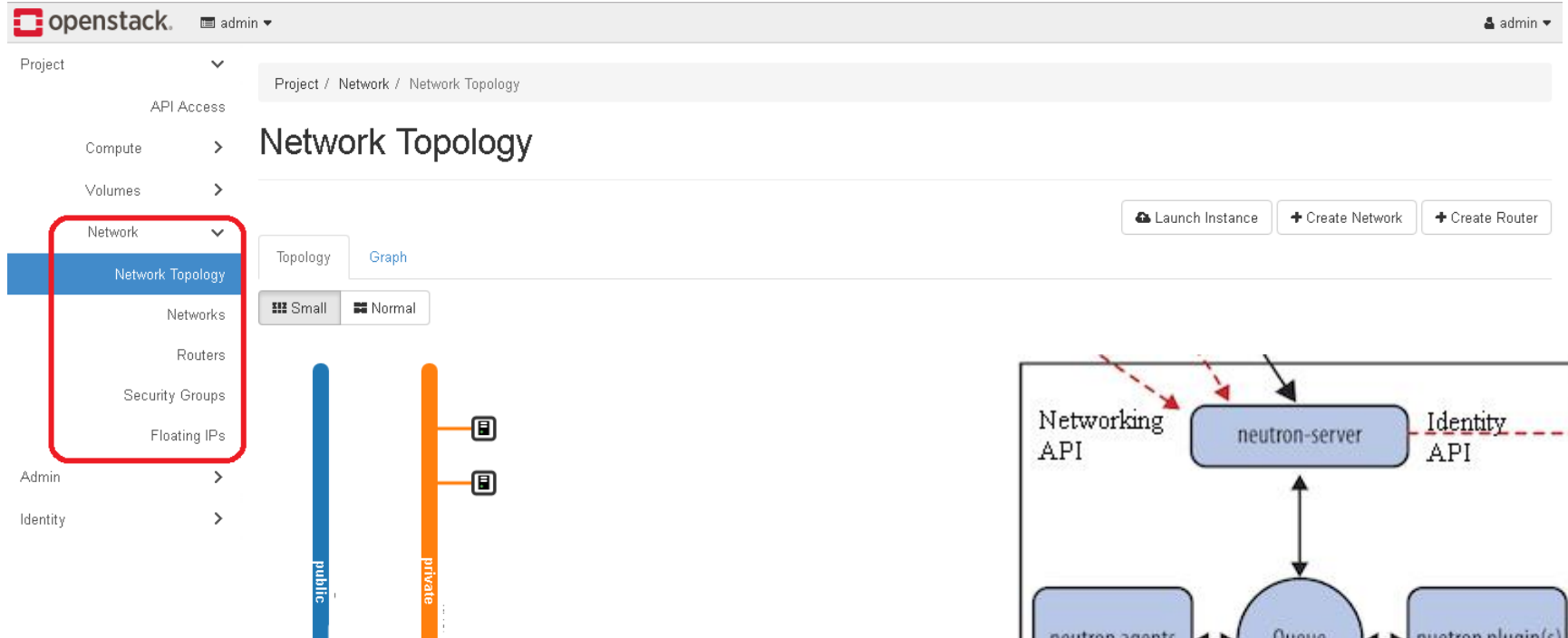
Compute service (NOVA) 3/3

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

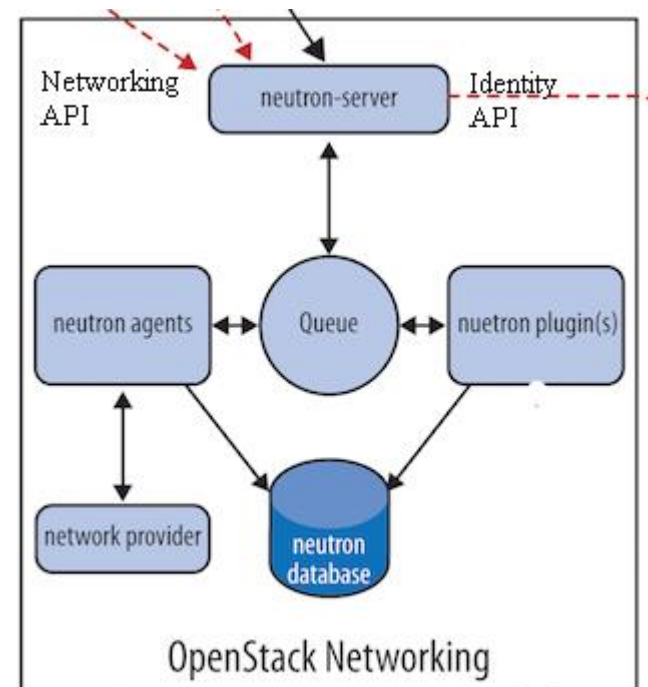
```
$ openstack server create --flavor <flavorName> \  
  --image <imageId> \  
  --nic net-id=<netId> \  
  --security-group <secName> \  
  --availability-zone <zone>:<host> \  
  --key-name <keyname> \  
  <VM_NAME>
```

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.

Amazon Web Services 1/2

Compute

Amazon Elastic Compute Cloud (Amazon EC2)



Amazon Elastic MapReduce



Storage

Amazon Simple Storage Service (Amazon S3)



Amazon Elastic Block Storage (Amazon EBS)



AWS Import/Export



AWS Storage Gateway Service

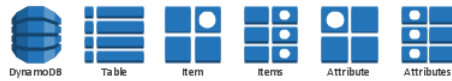


AWS Glacier



Database

Amazon DynamoDB



Amazon Relational Database Service (Amazon RDS)



Amazon ElastiCache



Networking

Amazon Route 53



Amazon Elastic Load Balancing



AWS Direct Connect



Amazon Virtual Private Cloud (VPC)



Content Delivery

Amazon CloudFront



Elastic Network Instance



Application Services

Amazon Simple Queue Service (SQS)



Amazon CloudSearch



Amazon Simple Email Service (SES)



Amazon Simple Workflow Service (SWF)



Amazon Simple Notification Service (SNS)



Deployment and Management

Amazon Elastic Beanstalk



AWS Identity and Access Management (IAM)



AWS CloudFormation



Monitoring

Amazon CloudWatch



Non-Service Specific



Groups



Amazon Web Services 2/2

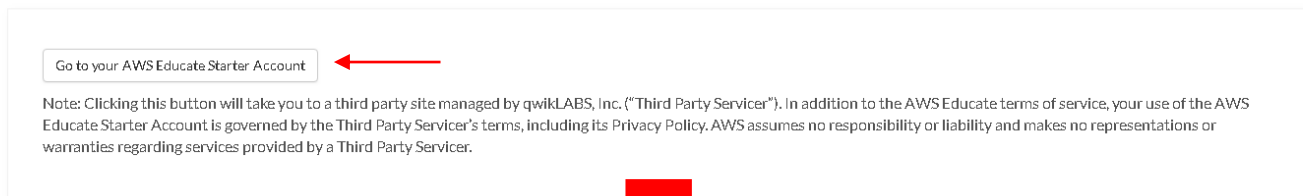
Amazon Web Services is a collection of cloud computing services offered by the Amazon company.

Some important services that the platform offers are:

- **Elastic Compute Cloud (EC2):** provides secure, resizable compute capacity in the cloud.
- **Simple Storage Service (S3):** provides object storage through web services interfaces . S3's design aims to provide scalability, high availability, and low latency at commodity costs.
- **Lambda:** is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you.
- ...many other services

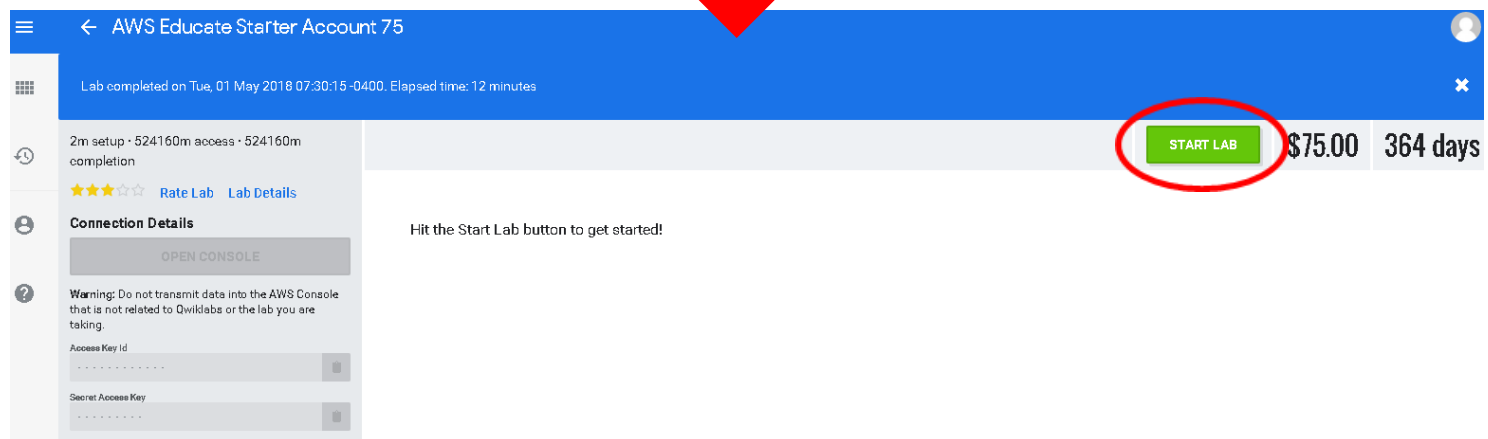
AWS Educate 1/2

- The AWS Educate program is Amazon's global initiative to provide students and educators with the resources needed to greatly accelerate Cloud-related learning.



Go to your AWS Educate Starter Account

Note: Clicking this button will take you to a third party site managed by qwikLABS, Inc. ("Third Party Servicer"). In addition to the AWS Educate terms of service, your use of the AWS Educate Starter Account is governed by the Third Party Servicer's terms, including its Privacy Policy. AWS assumes no responsibility or liability and makes no representations or warranties regarding services provided by a Third Party Servicer.



← AWS Educate Starter Account 75

Lab completed on Tue, 01 May 2018 07:30:16 -0400. Elapsed time: 12 minutes

2m setup · 524160m access · 524160m completion

★★★★☆ Rate Lab Lab Details

Connection Details

OPEN CONSOLE

Warning: Do not transmit data into the AWS Console that is not related to Qwiklabs or the lab you are taking.

Access Key Id
.....

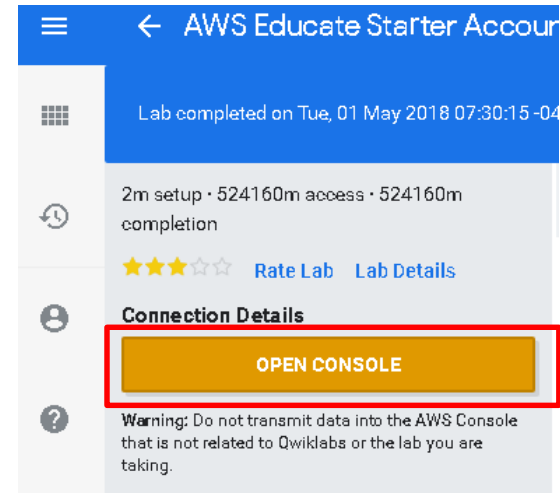
Secret Access Key
.....

Hit the Start Lab button to get started!

START LAB \$75.00 364 days

AWS Educate 2/2

- After clicking on Open Console, we will see the management console of all the Amazon Web Services for our account.



- **WARNING!!!** NEVER click END LAB button on previous page if you want to continue to use the AWS Educate program until its natural expiration



AWS Free Tier

- Amazon provides a free subscription plan. You must use a debit/credit card during the subscription.
- For the first year, you can freely use
 - an instance for 750 hours/month;
 - 5 GB of object storage;
 - ...many other thresholds per service.
- **WARNING!!!** You may incur in charge if you exceed the thresholds

AWS Console

- The Console facilitates cloud management for all aspects of your AWS account, including billing information, security credentials and so on

The screenshot shows the AWS Management Console interface for the US West (Oregon) region. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information. The left sidebar lists navigation options like 'EC2 Dashboard', 'Events', 'Tags', 'Reports', 'Limits', 'INSTANCES', 'IMAGES', 'ELASTIC BLOCK STORE', and 'NETWORK & SECURITY'. The main content area is divided into several sections:

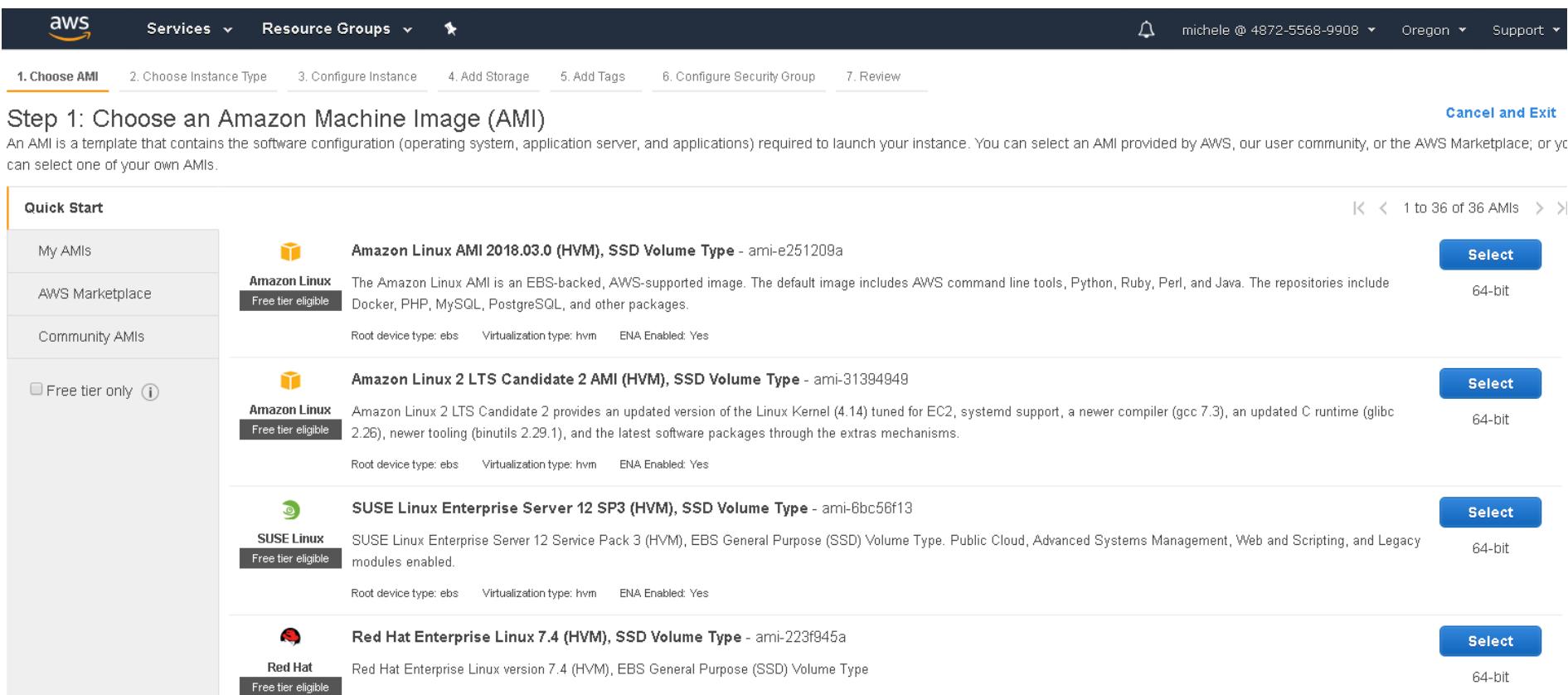
- Resources:** A summary of EC2 resources in the US West (Oregon) region, including 0 Running Instances, 0 Elastic IPs, 0 Dedicated Hosts, 0 Snapshots, 0 Volumes, 0 Load Balancers, 1 Key Pairs, and 1 Security Groups.
- Create Instance:** A section with a 'Launch Instance' button and a note that instances will launch in the US West (Oregon) region.
- Service Health:** A section showing the status of the US West (Oregon) service, which is operating normally, and the status of availability zones (us-west-2a and us-west-2b), both of which are also operating normally.
- Scheduled Events:** A section showing no events for the US West (Oregon) region.
- Account Attributes:** A section on the right showing account details like 'Supported Platforms', 'Default VPC', and 'Resource ID length management'.
- Additional Information:** A section on the right providing links to 'Getting Started Guide', 'Documentation', 'All EC2 Resources', 'Forums', 'Pricing', and 'Contact Us'.
- AWS Marketplace:** A section on the right featuring 'Barracuda CloudGen Firewall for AWS - PAYG' with a 5-star rating and pricing information.

AWS Elastic Compute Cloud (EC2)

- Amazon Elastic Compute Cloud provides scalable computing capacity in the Amazon Web Services (AWS) cloud.
- You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.
- Amazon EC2 provides the following features:
 - a) Virtual computing environments, known as **instances**;
 - b) Templates for your instances, known as **Amazon Machine Images (AMIs)**;
 - c) Configurations of CPU, memory, storage, and networking capacity for your instances, known as **instance types**;
 - d) Secure login information for your instances using **key pairs**;
 - e) Temporary/Persistent storage volumes, known as **Elastic Block Store volumes**;
 - f) A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using **security groups**;
 - g) Metadata, known as **tags**, that you can create and assign to your EC2 resources;
 - h) Virtual networks known as **virtual private clouds (VPCs)**, similarly to what we have seen for OpenStack Neutron.

AWS Amazon Machine Image 1/2

- To create a new Instance click on the  button.







1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI) Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start 1 to 36 of 36 AMIs

- My AMIs**
- AWS Marketplace**
- Community AMIs**
- Free tier only ⓘ

 Amazon Linux Free tier eligible	Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-e251209a The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages. Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	Select 64-bit
 Amazon Linux Free tier eligible	Amazon Linux 2 LTS Candidate 2 AMI (HVM), SSD Volume Type - ami-31394949 Amazon Linux 2 LTS Candidate 2 provides an updated version of the Linux Kernel (4.14) tuned for EC2, systemd support, a newer compiler (gcc 7.3), an updated C runtime (glibc 2.26), newer tooling (binutils 2.29.1), and the latest software packages through the extras mechanisms. Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	Select 64-bit
 SUSE Linux Free tier eligible	SUSE Linux Enterprise Server 12 SP3 (HVM), SSD Volume Type - ami-6bc56f13 SUSE Linux Enterprise Server 12 Service Pack 3 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled. Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	Select 64-bit
 Red Hat Free tier eligible	Red Hat Enterprise Linux 7.4 (HVM), SSD Volume Type - ami-223f945a Red Hat Enterprise Linux version 7.4 (HVM), EBS General Purpose (SSD) Volume Type	Select 64-bit

AWS Amazon Machine Image 2/2

- Amazon Machine Image (AMI) provides the information required to launch an instance.
- An AMI includes the following:
 - A template for the root volume for the instance (an operating system, an application server, applications, etc...).
 - Launch permissions that control which AWS accounts can use the AMI to launch instances.
 - A block device mapping that specifies the volumes to attach to the instance when it's launched.
- AMI Sources:
 - AMI provided by AWS,
 - AMI from the AWS Marketplace,
 - your own AMIs.
- We can choose the **Ubuntu Server 16.04 LTS** image for free

AWS Instance Configuration

After choosing the **t2.micro** instance type, we have to configure some parameters:

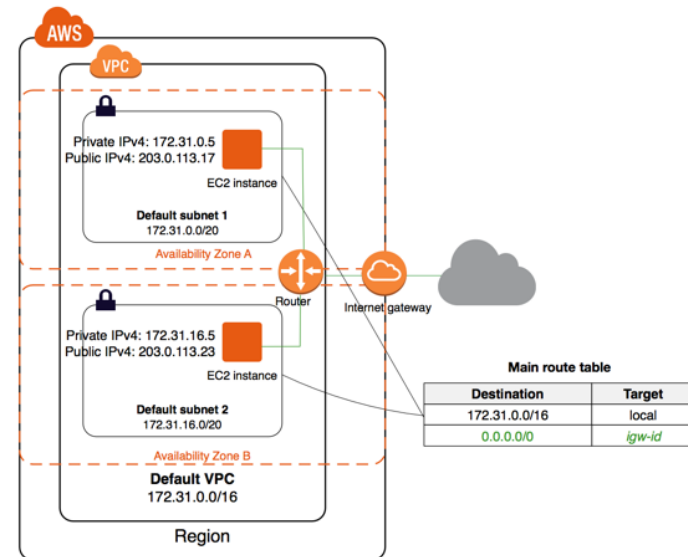
- a) **Number of instance [1]** = if you need multiple instances with the same configuration,
- b) **Network and Subnet [default]** = to launch your instance into the Amazon Virtual Private Cloud (VPC) network,
- c) **Storage [8 GiB SSD]** = you can choose to attach other volumes to you instance,
- d) **Security Group [NEW]** = a set of firewall rules that control the traffic for your instance. Create rule for **SSH** and **All ICMP** access,
- e) **Key Pair [NEW]** = create a new key pair. AWS stores the *public key* and (ONLY!!!) you store the *private key* file. You can download the key and use it to securely SSH into your instance.
- f) **Launch the Instance!**

AWS Identity and Access Management (IAM)

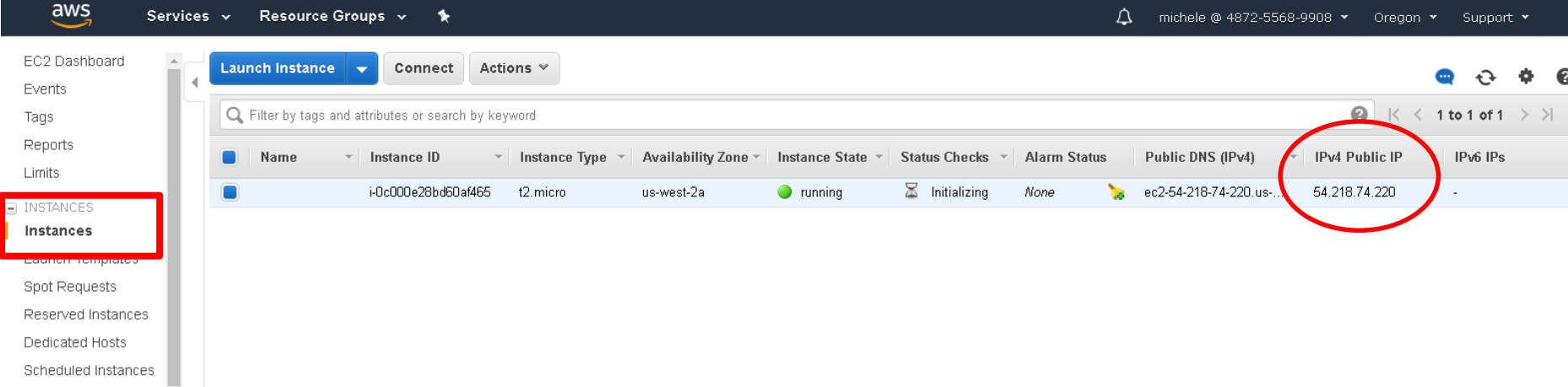
- AWS IAM is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- AWS account **root-user** = When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. The root-user is accessed by signing in with the email address and password that you used to create the account.
- IAM Roles for Amazon EC2 = a strategy for managing credentials for your applications that run on EC2 instances. For example you can choose what an instance can use of your AWS account.

AWS Virtual Private Cloud (VPC)

- The Amazon Virtual Private Cloud is a virtual networking environment, it is the networking layer for Amazon EC2. The VPC is logically isolated from other virtual networks in the AWS Cloud.
- You can configure your VPC by modifying its IP address range, create subnets, and configure route tables, network gateways, and security settings. A **subnet** is a range of IP addresses in your VPC. Your account comes with a default VPC that has a default subnet in which you can launch your instances.
- You control how the instances that you launch into a VPC access resources outside the VPC. Your default VPC includes an internet gateway, and each default subnet is a public subnet. Each instance that you launch into a default subnet has a private IPv4 address and a **public IPv4 address**. These instances can communicate with the internet through the internet gateway. An internet gateway enables your instances to connect to the internet through the Amazon EC2 network edge.



Amazon EC2 Running Instance



The screenshot displays the AWS Management Console interface for EC2 instances. The left-hand navigation menu is visible, with the 'INSTANCES' section expanded and 'Instances' selected, both highlighted with a red box. The main content area shows a table of instances. The table has columns for Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS (IPv4), IPv4 Public IP, and IPv6 IPs. A single instance is listed with the following details:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
	i-0c000e28bd60af465	t2.micro	us-west-2a	running	Initializing	None	ec2-54-218-74-220.us-...	54.218.74.220	-

The 'IPv4 Public IP' column value, '54.218.74.220', is circled in red. The top navigation bar shows the user is logged in as 'michele @ 4872-5568-9908' in the 'Oregon' region.

- We can see our instances by selecting the corresponding item from the menu.
- For this case, we have a **public IP address** dedicated to the instance.

AWS Exercises

ONE → Access to your Virtual Server.

- Use the priv. key (*I hope you downloaded it...*) with the right permission ... (600?).
- And the user?? ...*ubuntu* obviously! 😊

TWO → Deploy your own service on your Virtual Server.

- This seems a complex task...but there is python to help us!
- Simple http server on port 80 → `python3 -m http.server 80` .
- Permission denied...? WHAT?? Ah ok, the user *ubuntu* is in *sudo* group.

THREE → Reach your service from outside.

- Why can't I reach my resource?
- Do you remember the security group and its rules...?

References

- **OpenStack Docs:** <https://docs.openstack.org/>
- **OpenStack Slides:**
<http://lia.deis.unibo.it/Courses/CompNetworksM/1718/slides/Openstackx2.pdf>
- **DevStack Docs:** <https://docs.openstack.org/developer/devstack/>
- **AWS Educate:**
<https://www.awseducate.com/microsite/CommunitiesEngageHome>
- **AWS Free:** <https://aws.amazon.com/free/>
- **AWS Docs:** <https://aws.amazon.com/documentation/>

Thanks to All