Instalación y administración de OpenStack

Resumen: Este documento es un manual de introducción a la instalación, configuración y explotación de OpenStack Havana en sistemas GNU/Linux.

Autor: Alejandro Roca Alhama

Versión: v.1.9 (21/mayo/2014)

A falta de:

Índice de contenido

Requerimientos y enlaces interesantes....2
Instalación de OpenStack Havana en sistemas Ubuntu....2
Introducción....2
Prerrequisitos....3
Servicios y configuración básica....4
Nombres de los equipos y configuración de la red....4
Configuración del bonding y de las VLAN....7
¿Qué es el bonding?....7
Configuración de bonding y VLAN....7
Configuración para OpenStack....9
Configuración de red con dos interfaces....13
Instalación de Ubuntu 12.04....15
NTP....17
MySQL....18
Instalación de los repositorios Havana....19
Instalación del servicio de cola de mensajes....19
Instalación de Keystone....19
Instalación y configuración....19
Instalación y configuración de Glance....26
Probando el servicio glance....29
Instalación y configuración de Nova....30
Instalación en el nodo controlador....31
Instalación en los nodos de computación....35
Configuración de la red en los nodos de computación: nova-network....39
¿Cómo hacer que el rango de IP de las VMs coincida con el externo?....40
Administración de las redes....40
Arranque de instancias....41
Instalación de Horizon....43
Explotación de OpenStack....44
Creación de un nuevo proyecto....44
Script para la creación de máquinas virtuales....45

Requerimientos y enlaces interesantes

Distribución de Linux a utilizar:

- Ubuntu 12.04.4 LTS. (Precise Pangolin)

Instalación de OpenStack Havana en sistemas Ubuntu

Introducción

Esta sección muestra el proceso detallado de instalación y configuración de OpenStack basado en la plataforma Ubuntu 12.04.4 (Precise Pangolin) usando 5 servidores (nodos). Se usará uno de los servidores como nodo controlador, ejecutando los siguientes componentes de OpenStack:

- API Services (glance-api, nova-api).
- Nova (nova-consoleauth, nova-scheduler, nova-cert, nova-conductor).
- Servicios VNC (nova-novncproxy).
- Keystone.
- Neutron (o nova-network).
- Glance.
- Swift.
- Cinder.
- Ceilometer.
- Horizon.
- Servicios de estado:
  - MySQL.
  - QPid/RabbitMQ.

Los nodos de computación solo ejecutarán los siguientes servicios de Nova:

- Hipervisor (KVM).
- nova-compute.
- nova-network.

En nuestra instalación vamos a tener un nodo controlador y varios nodos de computación. En general, OpenStack permite los siguientes tipos de nodos:

- Endpoint node.
  - Servicios de balanceo de carga y alta disponibilidad.
- Controller node.
  - Ejecuta servicios de estado y los servicios propios de OpenStack.
- Compute node.
  - Son los nodos que ejecutan las instancias o máquinas virtuales (VM).
- Volume node.
  - Ejecutan servicios de almacenamiento: Cinder (antes nova-volume) e iSCSI target.

Podemos resumir el proceso global de instalación en los siguientes pasos:

1. Configuración de la red del Cloud.
2. Instalación de servicios básicos (NTP, MySQL, ...).
3. Instalación y configuración de Keystone (servicio de autenticación).
4. Instalación y configuración de Glance (servicio de gestión de imágenes).
5. Instalación y configuración de Nova (servicios de computación).
   Configuración del tipo de red: FlatNetwork, FlatDHCPNetwork o VLANNetwork.
6. Añadir imágenes para la creación de máquinas virtuales.
7. Iniciar una máquina virtual de prueba.
8. Instalación y configuración del Dashboard (Horizon).

El proceso de instalación y configuración de OpenStack es un proceso complejo y propenso a errores por lo que es muy importante conocer todos los elementos y comprender cómo interactúan unos con otros. OpenStack es un proyecto muy joven y bajo un desarrollo muy activo, para obtener información más allá de este documento se recomienda visitar las páginas oficiales:

- http://www.openstack.org
- http://docs.openstack.org
- https://launchpad.net/openstack/

Prerrequisitos

Para el seguimiento del proceso de instalación y configuración de OpenStack que se detalla posteriormente, es necesario partir de algunas suposiciones y de cumplir ciertos requisitos, son los siguientes:

- Se necesitan, al menos dos nodos con Ubuntu 12.04 LTS instalado.
- Uno de los nodos será el nodo controlador, que ejecutará todos los servicios excepto nova-compute y nova-network.
- Se recomienda el uso de LVM (el gestor de volúmenes lógicos de Linux), la configuración de LVM y el listado del esquema de particionamiento a seguir, se detalla en las siguientes secciones.
- La resolución de nombres DNS para todas las máquinas debe ser perfecta.
- Nuestra red cuenta con un nombre de dominio, concretamente *iescierva.net*, este será el dominio utilizado durante todo el documento.
- Si no se cuenta con un servidor DNS en la red, todas las máquinas deberán contar con el fichero /etc/hosts con todas las máquinas correctamente registradas.
- Todos los nodos que participen en el Cloud deben tener la fecha y hora sincronizada a través del servicio NTP (Network Time Protocol). Se recomienda que la red cuente con uno varios servidores NTP.
- Entre todos los hipervisores disponibles en la comunidad FLOSS, hemos optado por KVM.
- La contraseña para todos los usuarios y para todos los servicios será la misma: `OpenStack_PASSWORD`
  - Este documento presupone el uso de la misma contraseña PARA TODOS LOS SERVICIOS, en una instalación en un entorno de producción habría que tener una contraseña única para:

    - MySQL (usuario root).
    - MySQL (usuarios keystone, glance y nova).
    - RabbitMQ.
    - Keystone.
    - Servicio Keystone.
    - Servicio Glance.
    - Servicio Nova.
- El sistema deberá estar actualizado antes de empezar con el proceso de instalación/configuración:

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

- También se recomienda que todos los nodos del cloud ejecuten la misma versión del kernel. Actualmente se ha optado por un kernel 3.11, el proporcionado a través del paquete: `linux-generic-lts-saucy`.

Servicios y configuración básica

A continuación se detallan los aspectos clave en la instalación de Ubuntu y lo servicios básicos a instalar.

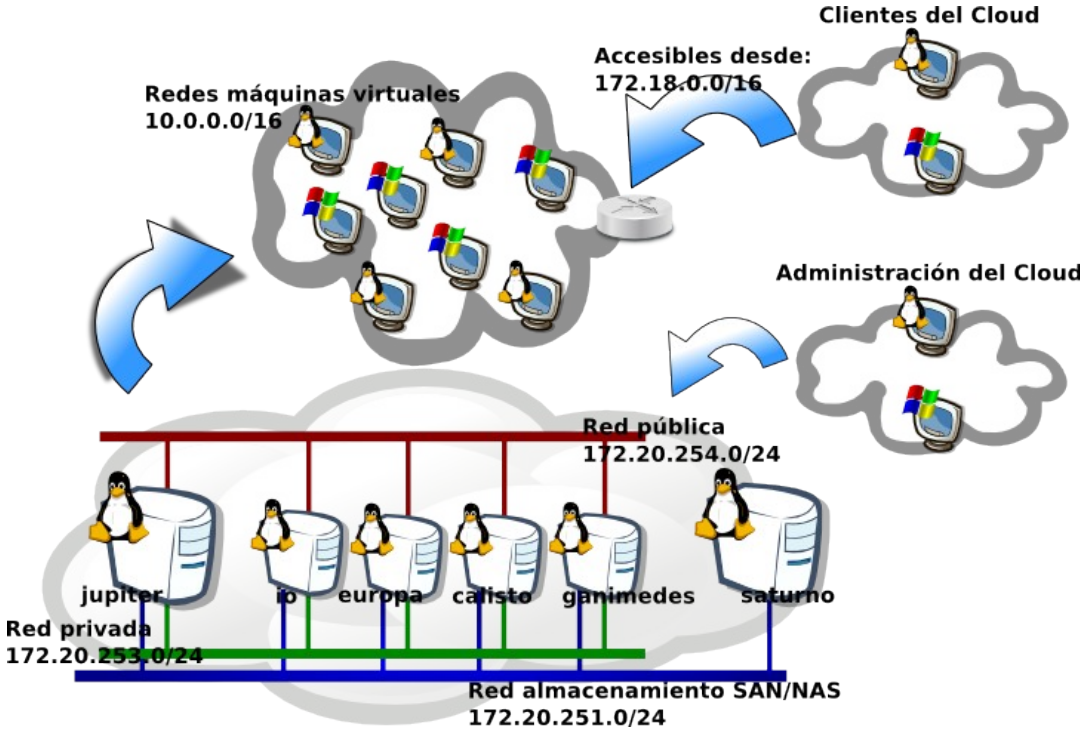
Nombres de los equipos y configuración de la red

Se han elegido los siguientes nombres para los equipos en los que se va a realizar la instalación de la infraestructura de OpenStack:

- **jupiter**. Nodo controlador.
    - Será el encargado de gestionar todos los recursos del cloud, interaccionar con los clientes y ordenar a los nodos de virtualización que ejecuten las instancias.
    - En jupiter no se ejecutaran máquinas virtuales.
    - La mayor parte de componentes del Cloud y configuración se realizará en este equipo, pero comparado con los nodos de computación la carga de trabajo será pequeña, por lo que no es necesario un equipo con mucha memoria RAM o gran capacidad de procesamiento.
- **io**, **europe**, **ganimedes** y **callisto**.
    - Las 4 lunas principales de júpiter.
    - Serán 4 los nodos de virtualización o nodos de computación, como se les denomina habitualmente en la jerga propia de OpenStack.
    - En estos equipos se instalarán sólo los componentes necesarios para que se ejecuten las instancias (máquinas virtuales) en ellos y estarán esperando las órdenes de jupiter.
- **venus**.
    - Será un equipo convencional en el que se instalarán los paquetes necesarios para usar el cliente nova con el que podemos gestionar el cloud desde línea de comandos sin la necesidad de realizar las operaciones desde jupiter.
    - La administración del Cloud a través de la interfaz web del Dashboard (Horizon) se podrá realizar desde cualquier máquina que tenga acceso a la red pública del

Cloud (incluyendo máquinas Windows).

Para la configuración de la red se ha optado por la siguiente configuración, tal como se muestra en la figura.



En nuestra infraestructura el controlador del Cloud (jupiter) cuenta con dos tarjetas de red Ethernet, mientras que los nodos de computación cuentan con seis, en lugar de asignar de forma estática las tarjetas a las redes se ha optado por una configuración en bonding y con VLANs.

De esta forma, todos los nodos tienen una sola interfaz de red, **bond0**, unión de todas las interfaces ethX del sistema, y en la que se han configurado las siguientes redes locales virtuales (VLAN):

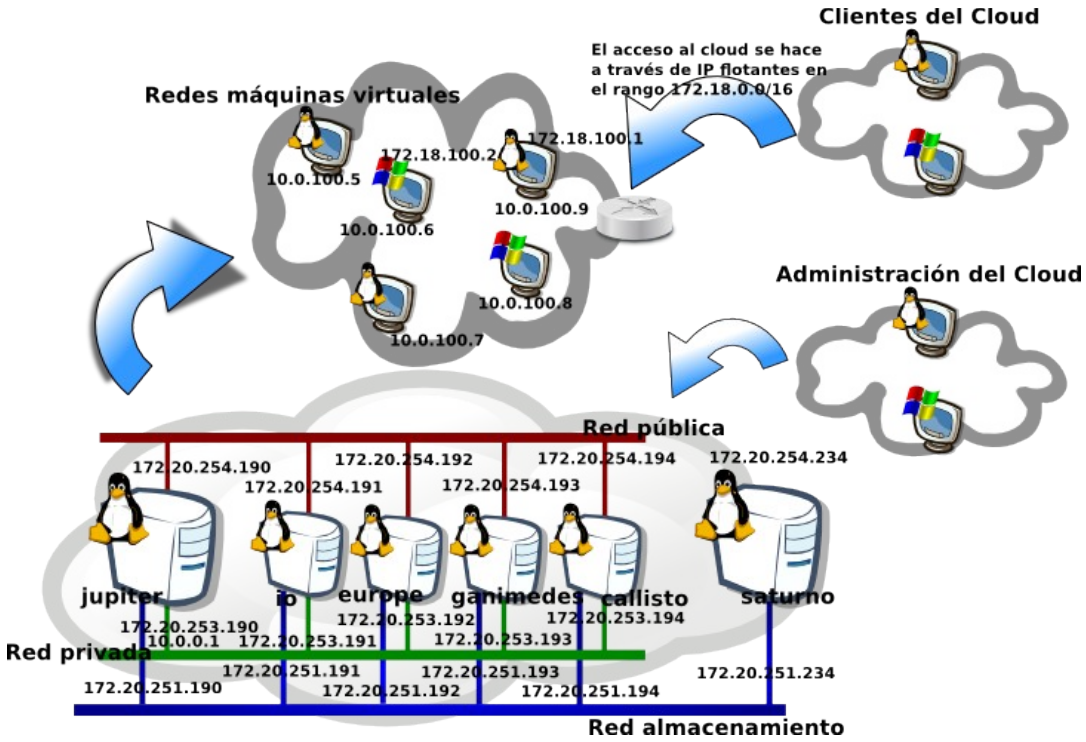
Red virtual	Dirección IP de la red	Interfaz virtual	Interfaz física (id VLAN)
Red pública	172.20.254.0/24	-	bond0 (untagged)
Red privada admin. del cloud	172.20.253.0/24	bond0.60	bond0 (VLAN 60)
Red interna VM A usar si se opta por la una configuración de red basada en VLAN y en direcciones IP flotantes.	10.0.0.0/16	bond0.XX	bond0 (VLAN XX)
Red almacenamiento	172.20.251.0/24	bond0.63	bond0 (VLAN 63)

Red acceso público VM (IP flotantes)	172.18.0.0/16	bond0.61	bond0 (VLAN 61)

De esta forma cada nodo tiene:

- Todas las interfaces desde eth0 hasta eth5 unidas por bonding a través de la interfaz bond0.
- La interfaz bond0, untagged (sin VLAN), asociada a la red pública.
  - En el caso del controlador la 172.20.254.190, en el caso del primer nodo, la 172.20.254.191, ...
- La interfaz bond0.60, unida a la red privada y utilizada para los servicios de gestión del Cloud.
  - En el caso del controlador la 172.20.253.190, en el caso del primer nodo, la 172.20.253.191, ...
- La interfaz bond0 también se usa para la comunicación entre las máquinas virtuales a través de la red 10.0.0.0/24. Por cada proyecto creado se crea una nueva VLAN con id 100, 101, 102, … Estas VLAN se construyen sobre la interfaz bond0. Internamente los nodos de computación utilizan un bridge, br100, br101, br102, para comunicar las máquinas virtuales entre ellas.
  - En las configuraciones Flat de nova-network esto no es necesario, sencillamente se usa la interfaz bond0.61 para el tráfico de máquinas virtuales.
- La interfaz bond0.62, asociada a la red de almacenamiento, utilizada para el acceso a los servidores que ofrecen SAN y NAS.
  - En el caso del controlador la 172.20.252.190, en el caso del primer nodo, la 172.20.252.191., ...
- La interfaz bond0.61, se utiliza para asignar direcciones IP flotantes. Una dirección IP flotante es la que asocia a una máquina virtual de forma temporal para que sea accesible desde fuera del Cloud. Esta dirección es la que se proporcionará a los clientes del Cloud.
  - La red utilizada será la 172.18.0.0/16.

A continuación se muestra la figura anterior con los nombres de las interfaces y las direcciones IP:



Configuración del bonding y de las VLAN

¿Qué es el bonding?

Bonding, es un concepto utilizado en redes de ordenadores, que describe varias formas de combinar (agregar) en paralelo varias interfaces de red creando una sola interfaz lógica. Esta nueva interfaz proporciona redundancia en el enlace y un mayor rendimiento que el proporcionado por las interfaces físicas trabajando de forma individual.

También se conoce como port trunking, link bundling, Ethernet/network/NIC bonding, o NIC teaming. Los estándares que hay bajo esta tecnología son el IEEE 802.1ax (LACP: Link Aggregation Control Protocol para redes Ethernet) o el protocolo previo IEEE 802.3ad, además de varias soluciones propietarias.

Configuración de bonding y VLAN

Para ofrecer un alto rendimiento, es casi obligatorio el uso de bonding, y altamente recomendable el uso de LACP (802.3ad), por lo que se recomienda un switch con soporte LACP.
Si no se cuenta con él, se puede optar por otros modos de bonding tal como se detalla en http://www.kernel.org/doc/Documentation/networking/bonding.txt.

Además de bonding, vamos a configurar varias VLAN sobre el bonding, para tener soporte en Ubuntu tenemos que hacer dos cosas:

1.- Instalar el paquete VLAN:

```
root@jupiter:~# apt-get install vlan
```

2. Asegurarnos de tener el módulo del kernel 8021q cargado, para ello modificamos el fichero /etc/modules para que quede así:

```
# /etc/modules: kernel modules to load at boot time.
```

```
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
loop
lp
rtc
8021q
```

Para la configuración de bonding en Ubuntu seguimos los siguientes pasos:
1.- Instalamos los siguientes paquetes:

```
root@jupiter:~# apt-get install ifenslave ethtool
```

2.- Bajamos las interfaces de red:

```
root@jupiter:~# ifdown eth0
root@jupiter:~# ifdown eth1
```

3.- (OPTATIVO) Creamos el fichero /etc/modprobe.d/bonding.conf con el siguiente contenido:

```
alias bond0 bonding
# options bonding mode=0 miimon=100
```

4.- Configuramos las nuevas interfaces a través del fichero /etc/network/interfaces:

```
#/etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual
  bond-master bond0
  pre-up ethtool -s eth0 wol g
  post-down ethtool -s eth0 wol g

auto eth1
iface eth1 inet manual
  bond-master bond0
  pre-up ethtool -s eth1 wol g
  post-down ethtool -s eth1 wol g

# The primary network interface
auto bond0
iface bond0 inet static
        address 172.20.254.191
        netmask 255.255.255.0
        broadcast 172.20.254.255
        network 172.20.254.0
        gateway 172.20.254.254
        bond-mode 802.3ad
        bond-miimon 100
        bond-lacp-rate 1
        bond-slaves none
        dns-nameservers 172.20.254.104 172.20.254.235
```

```
        dns-search iescierva.net
```

Las líneas:

```
  pre-up ethtool -s eth1 wol g
  post-down ethtool -s eth1 wol g
```

… no son obligatorias para el bonding, lo que hacen es activar el arranque de las máquinas por la red (WOL, Wake On LAN).

5. Reconfiguramos las interfaces de red, en principio no hace falta reiniciar, pero podemos hacerlo para comprobar que todo funciona correctamente tras iniciarse el sistema.

6.- Probamos que tenemos conectividad y que el bonding está configurado:

```
root@jupiter:~# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

802.3ad info
LACP rate: fast
Min links: 0
Aggregator selection policy (ad_select): stable
Active Aggregator Info:
      Aggregator ID: 1
      Number of ports: 2
      Actor Key: 17
      Partner Key: 58
      Partner Mac Address: 00:9c:02:b7:f9:40

Slave Interface: eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:25:90:72:2c:47
Aggregator ID: 1
Slave queue ID: 0

Slave Interface: eth0
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:25:90:72:2c:46
Aggregator ID: 1
Slave queue ID: 0
root@jupiter:~#
```

Configuración para OpenStack

En nuestra configuración, hemos optado por el uso de bonding más el uso de VLAN, tras

seguir los pasos anteriores habría que configurar el fichero /etc/network/interfaces tal como se muestra a continuación.

La configuración del bonding más las VLAN en el nodo jupiter sería la siguiente:

```
root@jupiter:~# cat /etc/network/interfaces
#/etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual
  bond-master bond0
  pre-up ethtool -s eth0 wol g
  post-down ethtool -s eth0 wol g

auto eth1
iface eth1 inet manual
  bond-master bond0
  pre-up ethtool -s eth1 wol g
  post-down ethtool -s eth1 wol g

# The primary network interface
auto bond0
iface bond0 inet static
      address 172.20.254.190
      netmask 255.255.255.0
      broadcast 172.20.254.255
      network 172.20.254.0
      gateway 172.20.254.254
      bond-mode 802.3ad
      bond-miimon 100
      bond-lacp-rate 1
      bond-slaves none
      dns-nameservers 172.20.254.104 172.20.254.235
      dns-search iescierva.net

# Private network for cloud administration
auto bond0.60
iface bond0.60 inet static
      address 172.20.253.190
      netmask 255.255.255.0
      broadcast 172.20.253.255
      network 172.20.253.0
      vlan-raw-device bond0

# Network for the Virtual Machines of the cloud (floating IP)
auto bond0.61
iface bond0.61 inet static
      address 172.18.254.190
      netmask 255.255.0.0
      broadcast 172.18.255.255
      network 172.18.0.0
      vlan-raw-device bond0

# Reserved use
```

```
auto bond0.62
iface bond0.62 inet manual
      #address 172.20.252.190
      #netmask 255.255.255.0
      #broadcast 172.20.252.255
      #network 172.20.252.0
      vlan-raw-device bond0

# SAN and NAS storage
auto bond0.63
iface bond0.63 inet static
      address 172.20.251.190
      netmask 255.255.255.0
      broadcast 172.20.251.255
      network 172.20.251.0
      vlan-raw-device bond0

root@jupiter:~# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

802.3ad info
LACP rate: fast
Min links: 0
Aggregator selection policy (ad_select): stable
Active Aggregator Info:
      Aggregator ID: 1
      Number of ports: 2
      Actor Key: 17
      Partner Key: 58
      Partner Mac Address: 00:9c:02:b7:f9:40

Slave Interface: eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:25:90:72:2c:47
Aggregator ID: 1
Slave queue ID: 0

Slave Interface: eth0
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:25:90:72:2c:46
Aggregator ID: 1
Slave queue ID: 0
root@jupiter:~# cat /proc/net/vlan/config
VLAN Dev name      | VLAN ID
Name-Type: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
bond0.60       | 60  | bond0
bond0.61       | 61  | bond0
bond0.62       | 62  | bond0
```

```
bond0.63       | 63   | bond0
root@jupiter:~#
```

Configuración de red en los nodos de computación

Para el proyecto OpenVDI se optó inicialmente por una configuración plana basada en nova-netowrk a través de una red FlatDHCP, en ese caso la configuración de red de los nodos de computación/red debería ser la siguiente:

```
#/etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual
  bond-master bond0
  pre-up ethtool -s eth0 wol g
  post-down ethtool -s eth0 wol g

auto eth1
iface eth1 inet manual
  bond-master bond0
  pre-up ethtool -s eth1 wol g
  post-down ethtool -s eth1 wol g

auto eth2
iface eth2 inet manual
  bond-master bond0
  pre-up ethtool -s eth2 wol g
  post-down ethtool -s eth2 wol g

auto eth3
iface eth3 inet manual
  bond-master bond0
  pre-up ethtool -s eth3 wol g
  post-down ethtool -s eth3 wol g

auto eth4
iface eth4 inet manual
  bond-master bond0
  pre-up ethtool -s eth4 wol g
  post-down ethtool -s eth4 wol g

auto eth5
iface eth5 inet manual
  bond-master bond0
  pre-up ethtool -s eth5 wol g
  post-down ethtool -s eth5 wol g

# The primary network interface
auto bond0
iface bond0 inet static
      address 172.20.254.191
      netmask 255.255.255.0
      broadcast 172.20.254.255
      network 172.20.254.0
```

```
      gateway 172.20.254.254
      bond-mode 802.3ad
      bond-miimon 100
      bond-lacp-rate 1
      bond-slaves none
      dns-nameservers 172.20.254.181
      dns-search iescierva.net

# Private network for cloud administration
auto bond0.60
iface bond0.60 inet static
      address 172.20.253.191
      netmask 255.255.255.0
      broadcast 172.20.253.255
      network 172.20.253.0
      vlan-raw-device bond0

# Network for the Virtual Machines of the cloud (floating IP)
auto bond0.61
iface bond0.61 inet manual
      vlan-raw-device bond0

# Reserved use
auto bond0.62
iface bond0.62 inet static
      address 172.20.252.191
      netmask 255.255.255.0
      broadcast 172.20.252.255
      network 172.20.252.0
      vlan-raw-device bond0

# SAN and NAS storage
auto bond0.63
iface bond0.63 inet static
      address 172.20.251.191
      netmask 255.255.255.0
      broadcast 172.20.251.255
      network 172.20.251.0
      vlan-raw-device bond0
```

Configuración de red con dos interfaces

Para una instalación y configuración básica tenemos la siguiente configuración de red en el nodo controlador:

Red	Red IP	Direcció IP	Interfaz
Red externa con acceso a Internet	- red externa- 192.168.166.0/24	192.168.166.190	eth0
Red de gestión de los nodos del cloud	172.20.253.0/24	172.20.253.190	eth1

El fichero de configuración de red para el nodo controlador sería el siguiente:

```
root@controller:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
```

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
      address 192.168.166.190
      netmask 255.255.255.0
      gateway 192.168.166.2
      dns-domain alex.local
      dns-nameservers 192.168.166.2
auto eth1
iface eth1 inet static
      address 172.20.253.190
      netmask 255.255.255.0
```

Para los nodos de computación la configuración de interfaces sería:

Red	Red IP	Direcció IP	Interfaz
Red externa con acceso a Internet	- red externa- 192.168.166.0/24	192.168.166.191	eth0
Red de gestión de los nodos del cloud	172.20.253.0/24	172.20.253.191	eth1
Red para el tráfico de las VMs	--no aplicable--	Direcciones VMs	eth2 (br100)

La configuración de eth2 sería:

```
root@node1:~# ifconfig eth2
eth2      Link encap:Ethernet  direcciónHW 00:0c:29:1a:46:1c
          Dirección inet6: fe80::20c:29ff:fe1a:461c/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:171 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:180 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:53376 (53.3 KB)  TX bytes:53578 (53.5 KB)

root@node1:~# brctl show
bridge name bridge id         STP enabled interfaces
br100       8000.000c291a461c no          eth2
                                          vnet0
```

El fichero de configuración de red para el nodo de computación sería el siguiente:

```
root@node1:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
      address 192.168.166.191
```

```
      netmask 255.255.255.0
      gateway 192.168.166.2
      dns-domain alex.local
      dns-nameservers 192.168.166.2
auto eth1
iface eth1 inet static
      address 172.20.253.191
      netmask 255.255.255.0

auto eth2
iface eth2 inet manual
```

Instalación de Ubuntu 12.04

La máquina *jupiter* incluye dos discos duros SATAII de 500 GiB y una controladora de disco 3ware 9650SE. Se ha optado por configurar esta controladora de disco en modo RAID 1, para incluir un nivel elemental de seguridad y consistencia de datos, aunque obviamente esto no descarta la utilización adicional de otros mecanismos de copias de seguridad que se configurarán posteriormente. Al tratarse de una controladora RAID hardware y estar configurada previamente, el sistema operativo que arranque en el equipo sólo verá un disco duro en /dev/sda de aproximadamente 500 GiB.

Para prevenir corrupción de datos es muy importante que la controladora RAID se configure con una política de escritura Write Through. En esta configuración, el rendimiento del RAID se resiente un poco, pero lo hace más inmune a una posible corrupción de datos en caso de cortes en el suministro eléctrico.

El sistema operativo elegido para los equipos del cloud es la distribución de GNU/Linux Ubuntu 12.04 LTS 64 bits, que actualmente se conoce con el nombre en código *Precise Pangolin*. Pensamos que es la mejor opción entre todas las distribuciones de GNU/Linux ya que es la utilizada en toda la documentación de OpenStack y la mejor soportada.

Durante la instalación realizamos el siguiente esquema de particionamiento:

```
root@jupiter:~# fdisk -l /dev/sda

Disk /dev/sda: 500.0 GB, 499989348352 bytes
255 heads, 63 sectors/track, 60786 cylinders, total 976541696 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000e7fd7

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048   124999679    62498816   83  Linux
/dev/sda2       124999680   132812799     3906560   82  Linux swap / Solaris
/dev/sda3       132812800   976541695   421864448   8e  Linux LVM
```

Crearemos tres particiones, una partición de unos 32 GiB para el sistema (directorio /), una partición de 4 GiB como área de intercambio, y el resto configurado para su uso a través de volúmenes lógicos con LVM.

Tras la instalación del sistema, configuraremos el subsistema LVM pero sin llegar a crear

ningún volumen. Podemos crear un volumen en cualquier momento a través de los comandos `pvcreate`, `vgcreate` y `lvcreate`. Los directorios susceptibles de residir en un volumen son:

- **`/var/lib/glance`**, incluye los ficheros imágenes del servicio Glance. Conviene formatearlo con el Sistema de Ficheros XFS.
  - jupiter sí contendrá este volumen.
- **`/var/lib/nova`**, incluye ciertos ficheros para el funcionamiento del servicio Nova, así como los discos virtuales de las instancias en ejecución. Conviene formatearlo con el Sistema de Ficheros XFS.
  - jupiter no contendrá este volumen a no ser que ejecute algún servicio de nova.
  - Los nodos de computación sí contendrán este volumen.

Aunque la controladora RAID es hardware y el sistema operativo no la gestiona, es importante que se pueda controlar su estado a través de algún módulo del kernel. En este caso el módulo `3w-9xxx` que se carga automáticamente y nos envía estos mensajes a los logs del sistema:

```
3ware 9000 Storage Controller device driver for Linux v2.26.02.014.
3w-9xxx 0000:01:00.0: PCI INT A -> GSI 19 (level, low) -> IRQ 19
3w-9xxx 0000:01:00.0: setting latency timer to 64
3w-9xxx: scsi6: Found a 3ware 9000 Storage Controller at 0xfe8df000, IRQ: 19.
3w-9xxx: scsi6: Firmware FE9X 4.08.00.006, BIOS BE9X 4.08.00.001, Ports: 2.
3w-9xxx: scsi6: AEN: INFO (0x04:0x0029): Verify started:unit=0.
3w-9xxx: scsi6: AEN: INFO (0x04:0x002B): Verify completed:unit=0.
```

Para Debian existe un repositorio bajo http://jonas.genannt.name con aplicaciones para manejar el RAID 3ware 9650SE. Para Ubuntu podemos optar por utilizar el software oficial de 3ware. Para instalar este software seguimos los siguientes pasos:
1.- Nos descargamos el software desde el link "SUPPORT" de la página www.3ware.com (ahora LSI). Se trata de la controladora RAID 3ware 9650SE-2LP.
2.- Nos descargamos el fichero 3DM2_CLI-Linux_10.2.1_9.5.4.zip

3.- Ejecutamos los siguientes comandos:

```
root@jupiter:~# unzip 3DM2_CLI-Linux-10.1.zip -d 3dm2
root@jupiter:~# cd 3dm2
root@jupiter:~# bash install.sh -i
```

4.- Seguimos el asistente en modo texto.

Al finalizar tendremos un demonio iniciado, una web administrativa (opcional) y el comando tw_cli para la gestión de la controladora.

Como software a instalar solo instalaremos el servidor SSH, a través del comando:

```
root@jupiter:~# apt-get install openssh-server
```

Tras finalizar la instalación actualizaremos la máquina a través de los comandos:

```
root@jupiter:~# apt-get update
root@jupiter:~# apt-get upgrade
root@jupiter:~# apt-get dist-upgrade
```

Tras la instalación, lo más probable es que tengamos que reiniciar.

NTP

Para mantener todos los servicios sincronizados (a nivel de fecha y hora) es necesario instalar un cliente NTP (Network Time Protocol). En el caso de instalaciones multinodo hay que configurar uno de los nodos como servidor NTP, o confiar en otro servidor de nuestra red o de Internet.

La red `iescierva.net` ya cuenta con un servidor NTP, por lo que únicamente hay que configurar correctamente el cliente, para ello basta con que sigamos los siguientes pasos:

1.- Nos aseguramos que el paquete `ntpdate` esté instalado en todos los nodos, incluyendo el nodo controlador.

```
root@jupiter:~# dpkg --list | grep ntpdate
```

Si no lo estuviera lo instalamos:

```
root@jupiter:~# apt-get install ntpdate
```

2.- Configuramos crontab para que se ejecute el comando ntpdate de forma periódica. Para ello ejecutamos (como root) el comando `crontab -e` y editamos el fichero que nos sugiere añadiendo la siguiente línea al final:

```
# [...]
0 4 * * * /usr/sbin/ntpdate ntp.iescierva.net ; /sbin/hwclock --systohc
```

Podemos sustituir el servidor `ntp.iescierva.net` por uno en Internet como `ntp.ubuntu.com` o como `hora.rediris.es`.

Podemos comprobar que la configuración es correcta a través del comando crontab -l:

```
root@jupiter:~# crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 4 * * * /usr/sbin/ntpdate ntp.iescierva.net ; /sbin/hwclock --systohc
```

MySQL

Vamos a configurar todos los servicios del Cloud para que utilicen como base de datos MySQL, en vez de la base de datos SQLite que se usa en la configuración por defecto. Para ello será necesario instalar MySQL y fijar una contraseña para el usuario root. Para ello seguimos los siguientes pasos:
1.- Instalamos los paquetes necesarios, básicamente el servidor MySQL, y la interfaz (DB-API) de Python para MySQL:

```
root@jupiter:~# apt-get install mysql-server python-mysqldb
```

2.- Durante la instalación del servidor se nos pedirá que introduzcamos la contraseña para el usuario root de MySQL, en nuestro caso *OpenStack_PASSWORD*

3.- Modificamos la interfaz de escucha de MySQL, aunque inicialmente la fijamos a 0.0.0.0 para que el servidor escuche en todas las interfaces, posteriormente fijaremos la interfaz con la IP de la red privada del Cloud.
Configuramos el fichero /etc/mysql/my.cnf modificando la siguiente línea:

```
bind-address = 127.0.0.1
```

Por estas otras:

```
#bind-address           = 127.0.0.1
bind-address            = 172.20.253.190
```

4.- Reiniciamos el demonio de MySQL:

```
root@jupiter:~# service mysql restart
```

5.- Securizamos el acceso a MySQL a través de los comandos:

```
root@jupiter:~# mysql_install_db
...
root@jupiter:~# mysql_secure_installation
...
```

Estos comandos nos aseguran borrar el usuarios anónimo y las bases de datos de pruebas.

6.- Probamos el cliente MySQL con la contraseña fijada:

```
root@jupiter:~# mysql -u root -p
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1181
Server version: 5.5.31-0ubuntu0.12.04.2 (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql>
```

En el resto de nodos basta con instalar únicamente el cliente MySQL y la librería de Python asociada:

```
root@nodo01:~# apt-get install python-mysqldb
```

Instalación de los repositorios Havana

Para instalar OpenStack Havana es necesario configurar el repositorio especial "Ubuntu Cloud Archive", para ello instalamos el siguiente paquete, añadimos el repositorio, actualizamos el sistema y reiniciamos:

```
root@jupiter:~# apt-get install python-software-properties
root@jupiter:~# add-apt-repository cloud-archive:havana
root@jupiter:~# apt-get update
root@jupiter:~# apt-get upgrade
root@jupiter:~# apt-get dist-upgrade
root@jupiter:~# reboot
```

Instalación del servicio de cola de mensajes

Para el funcionamiento de OpenStack se necesita de un servicio de cola de mensajes ("Advanced Message Queuing Protocol"). OpenStack opta por la instalación de RabbitMQ:

```
root@jupiter:~# apt-get install rabbitmq-server
```

Por defecto se usa el usuario guest con password guest, para cambiarla ejecutamos:

```
root@jupiter:~# rabbitmqctl change_password guest OpenStack_PASSWORD
```

Instalación de Keystone

En este apartado describiremos la instalación y configuración de unos de los servicios básicos de OpenStack, el servicio de autenticación y gestión de la identidad (Identity Service): **Keystone** .

Instalación y configuración

Una infraestructura OpenStack solo necesita un servidor que ejecute el servicio KeyStone, en nuestra instalación el servicio se ejecutará en el controlador, es decir, en la máquina *jupiter*.

El servicio de identidad Keystone realiza dos funciones fundamentales:

- Gestión de usuarios.
  - Encargado del mantenimiento de usuarios y sus permisos asociados.
- Catálogo de servicios.
  - Proporciona un catálogo de los servicios disponibles así como los endpoints de las API asociadas.

Para la instalación de Keystone seguimos los siguientes pasos:

1.- Instalamos todo el software necesario:

```
root@jupiter:~# apt-get install keystone
```

2.- Keystone utiliza por defecto una base de datos SQLite, por lo que tendremos que borrar la base de datos que se configura en la instalación por defecto:

```
root@jupiter:~# rm /var/lib/keystone/keystone.db
```

3.- Editamos el fichero de configuración principal de Keystone para realizar los siguientes cambios:

- La interfaz de escucha del servicio.
- El token administrativo (admin_token).
  - Podemos generar un token aleatorio a través del comando:

```
openssl rand -hex 10
```

- La cadena de conexión a la base de datos keystone de MySQL.

Para ello editamos el fichero /etc/keystone/keystone.conf haciendo las siguientes modificaciones:

```
[DEFAULT]
# A "shared secret" between keystone and other openstack services
# admin_token = ADMIN
admin_token = 343c7e259ad32fd07b1b

# The IP address of the network interface to listen on
# bind_host = 0.0.0.0
bind_host = 172.20.253.190

# The port number which the public service listens on
# public_port = 5000

# The port number which the public admin listens on
# admin_port = 35357

# The base endpoint URLs for keystone that are advertised to clients
# (NOTE: this does NOT affect how keystone listens for connections)
# public_endpoint = http://localhost:%(public_port)s/
# admin_endpoint = http://localhost:%(admin_port)s/

...

[sql]
# The SQLAlchemy connection string used to connect to the database
#connection = sqlite:////var/lib/keystone/keystone.db
connection = mysql://keystone:OpenStack_PASSWORD@172.20.253.190/keystone

# the timeout before idle sql connections are reaped
# idle_timeout = 200

...
```

Ver el subapartado posterior sobre admin_token.

4.- Creamos la base de datos que el servicio necesita en MySQL. Para ello iniciamos el cliente MySQL:

```
root@jupiter:~# mysql -u root -p
```

Desde el prompt de MySQL ejecutamos las siguientes sentencias:

```
mysql> CREATE DATABASE keystone;
mysql> GRANT ALL PRIVILEGES ON keystone.* to 'keystone'@'localhost'
IDENTIFIED BY 'OpenStack_PASSWORD';
mysql> GRANT ALL PRIVILEGES ON keystone.* to 'keystone'@'%' IDENTIFIED BY
'OpenStack_PASSWORD';
mysql> FLUSH PRIVILEGES;
```

5. Creamos la base de datos inicial:

```
root@jupiter:~# keystone-manage db_sync
```

6.- Reiniciamos el servicio:

```
root@jupiter:~# service keystone restart
```

7. Creamos los usuarios, proyectos (tenants) y roles.
Crearemos ciertos usuarios a los que daremos ciertos roles en ciertos proyectos, tal como se muestra en la siguiente tabla:

Usuarios	Tenants (proyectos)	Roles
admin	admin	admin
nova glance swift	service	admin
admin	admin	_member_

De la tabla se puede deducir, por ejemplo, que al usuario *admin* tiene el rol *admin* en el proyecto *admin*, o que a los usuarios *nova*, *glance* y *swift* tienen el rol *admin* en el proyecto *service*.

Antes de crear usuarios debemos definir dos variables de entorno para la autenticación, aún no hemos creado usuarios, por lo que solo nos queda autenticarnos a través del token de administración antes definido. Ejecutamos los siguientes comandos:

```
root@jupiter:~# export OS_SERVICE_TOKEN=343c7e259ad32fd07b1b
root@jupiter:~# export OS_SERVICE_ENDPOINT=http://172.20.253.190:35357/v2.0
```

La creación de usuarios, proyectos y roles es un proceso tedioso, muy repetitivo y propenso a errores, por lo que ejecutaremos el siguiente script que nos simplificará mucho este proceso `/root/bin/KeyStoneRoles.sh`:

```
#!/bin/bash

# Alejandro Roca Alhama
# Script para la creación inicial de usuarios, proyectos y roles.
# Versión 2.0.
# Última modificación: 3/enero/2014.
```

```
# Solo hay que modificar estos parámetros
PASSWORD='OpenStack_PASSWORD'
EMAIL=alex@iescierva.net
PUBLIC_IP=172.20.253.190
PRIVATE_IP=172.20.253.190
ADMIN_IP=172.20.253.190

# Creación de tenants, usuarios y roles

keystone tenant-create --name admin --description "Admin Tenant"
keystone tenant-create --name service --description "Service Tenant"

keystone user-create --name admin --pass $PASSWORD --email $EMAIL
keystone user-create --name nova --pass $PASSWORD --email $EMAIL
keystone user-create --name glance --pass $PASSWORD --email $EMAIL
#keystone user-create --name swift --pass $PASSWORD --email $EMAIL

keystone role-create --name admin
#keystone role-create --name Member

#ADMIN_TENANT=`keystone tenant-list | grep admin | tr -d " " | awk -F \| '
{ print $2 }'`
#SERVICE_TENANT=`keystone tenant-list | grep service | tr -d " " | awk -F \| '
{ print $2 }'`

#ADMIN_ROLE=`keystone role-list | grep admin | tr -d " " | awk -F \| ' { print
$2 }'`
#MEMBER_ROLE=`keystone role-list | grep Member | tr -d " " | awk -F \| '
{ print $2 }'`

#ADMIN_USER=`keystone user-list | grep admin | tr -d " " | awk -F \| ' { print
$2 }'`

# Añadimos el rol admin al usuario admin en el tenant admin
keystone user-role-add --user=admin --tenant=admin --role=admin
keystone user-role-add --user=glance --tenant=service --role=admin
keystone user-role-add --user=nova --tenant=service --role=admin
#keystone user-role-add --user=admin --tenant=admin –role=Member

# Creamos los servicios
keystone service-create --name keystone --type identity --description "Keystone
Identity Service"
keystone service-create --name nova --type compute --description "Nova Compute
Service"
#keystone service-create --name volume --type volume --description "OpenStack
Volume Service"
keystone service-create --name glance --type image --description "Glance Image
Service"
#keystone service-create --name swift --type object-store --description
"OpenStack Storage Service"

#keystone service-create --name ec2 --type ec2 --description "OpenStack EC2
Service"

# Creamos los endpoints
#for service in nova volume glance swift keystone ec2
for service in nova keystone glance
do
    ID=`keystone service-list | grep $service | tr -d " " | awk -F \| ' { print
$2 } '`
    case $service in
```

```
    "nova"     ) keystone endpoint-create --service-id $ID \
                 --publicurl   "http://$PUBLIC_IP"':8774/v2/%(tenant_id)s' \
                 --adminurl    "http://$ADMIN_IP"':8774/v2/%(tenant_id)s' \
                 --internalurl "http://$PRIVATE_IP"':8774/v2/%(tenant_id)s'
    ;;
    "volume"   ) keystone endpoint-create --service_id $ID \
                 --publicurl   "http://$PUBLIC_IP"':8776/v1/$(tenant_id)s' \
                 --adminurl    "http://$ADMIN_IP"':8776/v1/$(tenant_id)s' \
                 --internalurl "http://$PRIVATE_IP"':8776/v1/$(tenant_id)s'
    ;;
    "glance"   ) keystone endpoint-create --service-id $ID \
                 --publicurl   "http://$PUBLIC_IP"':9292' \
                 --adminurl    "http://$ADMIN_IP"':9292' \
                 --internalurl "http://$PRIVATE_IP"':9292'
    ;;
    "swift"    ) keystone endpoint-create --service_id $ID \
                 --publicurl   "http://$PUBLIC_IP"':8080/v1/AUTH_$(tenant_id)s'
\
                 --adminurl    "http://$ADMIN_IP"':8080/v1' \
                 --internalurl "http://$PRIVATE_IP"':8080/v1/AUTH_$
(tenant_id)s'
    ;;
    "keystone" ) keystone endpoint-create --service_id $ID \
                 --publicurl   "http://$PUBLIC_IP"':5000/v2.0' \
                 --adminurl    "http://$ADMIN_IP"':35357/v2.0' \
                 --internalurl "http://$PRIVATE_IP"':5000/v2.0'
    ;;
    "ec2"      ) keystone endpoint-create --service_id $ID \
                 --publicurl   "http://$PUBLIC_IP"':8773/services/Cloud' \
                 --adminurl    "http://$ADMIN_IP"':8773/services/Admin' \
                 --internalurl "http://$PRIVATE_IP"':8773/services/Cloud'
    ;;
    esac
done
```

Para la correcta ejecución de script hay que configurar las siguientes variables de entorno en el inicio del script:

- PASSWORD: contraseña de acceso de los usuarios que se van a crear. El script utiliza para el acceso a Keystone el token administrativo definido anteriormente. Este token se puede eliminar una vez que los usuarios estén creados.
- EMAIL: al crear los usuarios hay que asignar una dirección de correo, al tratarse de usuarios administrativos, podemos utilizar la misma cuenta para todos.
- PUBLIC_IP (publicurl): dirección pública de acceso a los servicios del cloud.
- PRIVATE_IP (internalurl): dirección privada o interna de acceso a los servicios del cloud.
- ADMIN_IP (adminurl): dirección de administración para los servicios del cloud. Los servicios de OpenStack o EC2 utilizan diferentes endpoints para la red, pero para otros servicios esto puede coincidir.

En nuestro cloud los endpoints públicos/privados/internos coinciden en la red privada de administración del cloud. Cualquier servicio como nuestro Broker OpenVDI debe estar en esta red.

El script anterior se limita a:

- Crear los proyectos (tenants) necesarios: admin y service.
- Crear todos los usuarios y asignarles contraseña y dirección de correo electrónico. Los usuarios creados son: admin, glance y nova.
- Crear los roles necesarios: admin y _member_.
- Asignar roles a los usuarios en los proyectos tal como se mostró en la tabla anterior.
- Crear los servicios (keystone, glance y nova). Define los endpoints y los asigna a los servicios.
  - En nuestra infraestructura los tres endpoints (public, internal y admin) se han configurado en la misma VLAN, la destinada a la gestión interna del cloud.

8.- Verificamos que todo se ha creado correctamente a través de los siguientes comandos (los identificadores no coincidirán):

```
root@jupiter:~# keystone tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| bafa223e38b74c3383154a8582efbfa8 |  admin  |   True  |
| 7fb59063ffb540739e647a3cda5015d0 | service |   True  |
+----------------------------------+---------+---------+
root@jupiter:~# keystone user-list
+----------------------------------+--------+---------+--------------------+
|                id                |  name  | enabled |       email        |
+----------------------------------+--------+---------+--------------------+
| 2607b04e15c64c3fae886e324fc195ea | admin  |   True  | alex@iescierva.net |
| fd34ac2486fd4c68aa1998b54cb9ffe6 | glance |   True  | alex@iescierva.net |
| cc5de1b46bd443eb8c52dd950d398731 |  nova  |   True  | alex@iescierva.net |
+----------------------------------+--------+---------+--------------------+
root@jupiter:~# keystone role-list
+----------------------------------+----------+
|                id                |   name   |
+----------------------------------+----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| a56f88473d554592a90f7f0abae127cf |  admin   |
+----------------------------------+----------+
root@jupiter:~# keystone endpoint-list
+----------------------------------+-----------
+------------------------------------------
+------------------------------------------
+------------------------------------------+----------------------------------+
|                id                |   region  |               publicurl
|             internalurl                |                adminurl
|           service_id            |
+----------------------------------+-----------
+------------------------------------------
+------------------------------------------
+------------------------------------------+----------------------------------+
| 3e55ea7809f242a394aa2da37d9ad9bb | regionOne |
http://127.0.0.1:5000/v2.0       |       http://127.0.0.1:5000/v2.0       |
http://127.0.0.1:35357/v2.0       | e52ab1bce09a4439801c96b3caf7cb08 |
| c1445396f569476982e637dbcce311ad | regionOne | http://127.0.0.1:8774/v2/%
```

```
(tenant_id)s | http://127.0.0.1:8774/v2/%(tenant_id)s |
http://127.0.0.1:8774/v2/%(tenant_id)s | 6d054e6c7ec84ac59528d3b2110a2f37 |
| e06527d3f01e4f8b9ecea23d11fcc1af | regionOne |         http://127.0.0.1:9292
|         http://127.0.0.1:9292          |         http://127.0.0.1:9292
| e6e49d32b665485a842e4801eb2a806e |
+----------------------------------+-----------
+--------------------------------------
+--------------------------------------
+--------------------------------------+----------------------------------+
root@jupiter:~# keystone service-list
+----------------------------------+----------+----------
+--------------------------+
|                id                |   name   |   type   |        description
|
+----------------------------------+----------+----------
+--------------------------+
| e6e49d32b665485a842e4801eb2a806e |  glance  |  image   |    Glance Image
Service   |
| e52ab1bce09a4439801c96b3caf7cb08 | keystone | identity | Keystone Identity
Service |
| 6d054e6c7ec84ac59528d3b2110a2f37 |   nova   | compute  |    Nova Compute
Service   |
+----------------------------------+----------+----------
+--------------------------+
root@jupiter:~#
```

9.- Para verificar que el servicio funciona correctamente ejecutamos:

```
root@jupiter:~# unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
root@jupiter:~# keystone --os-username=admin --os-password='OpenStack_PASSWORD'
--os-auth-url=http://172.20.253.190:35357/v2.0 token-get
```

Además de eliminar las variables de entorno, también podemos comentar el token administrativo configurado en el fichero de configuración de Keystone así como reiniciar el servicio:

```
root@jupiter:~# head /etc/keystone/keystone.conf
[DEFAULT]
# A "shared secret" between keystone and other openstack services
# admin_token = ADMIN
#admin_token = 0fe694ac0df86d4835d0

...
root@jupiter:~/bin# service keystone restart
keystone stop/waiting
keystone start/running, process 10782
root@jupiter:~/bin#
```

El resultado debe ser que se obtiene un token junto al ID del usuario. También se puede probar la autenticación sobre un tenant así:

```
root@jupiter:~# keystone --os-username=admin --os-password='OpenStack_PASSWORD'
--os-tenant-name=admin --os-auth-url=http://172.20.253.190:35357/v2.0 token-get
```

10.- Para simplificar la ejecución de comandos podemos crear un fichero con el siguiente contenido.

```
root@jupiter:~# cat .keystonerc
# Variables de entorno para la autenticación de OpenStack
OS_USERNAME=admin
```

```
OS_PASSWORD='OpenStack_PASSWORD'
OS_TENANT_NAME=admin
OS_AUTH_URL=http://localhost:35357/v2.0

export OS_USERNAME OS_PASSWORD OS_TENANT_NAME OS_AUTH_URL
```

Para autenticarnos basta con ejecutar:

```
root@jupiter:~# source .keystonerc
root@jupiter:~# keystone token-get
```

Comprobamos que el usuario admin tiene autorización para realizar tareas administrativas, para ello el usuario admin debe tener el rol admin:

```
root@jupiter:~# keystone user-list
+----------------------------------+--------+---------+--------------------+
|                id                |  name  | enabled |       email        |
+----------------------------------+--------+---------+--------------------+
| 2607b04e15c64c3fae886e324fc195ea | admin  |   True  | alex@iescierva.net |
| fd34ac2486fd4c68aa1998b54cb9ffe6 | glance |   True  | alex@iescierva.net |
| cc5de1b46bd443eb8c52dd950d398731 |  nova  |   True  | alex@iescierva.net |
+----------------------------------+--------+---------+--------------------+
root@jupiter:~# keystone user-role-list
+----------------------------------+-------+----------------------------------
+----------------------------------+
|                id                |  name |             user_id              |
tenant_id             |
+----------------------------------+-------+----------------------------------
+----------------------------------+
| a56f88473d554592a90f7f0abae127cf | admin | 2607b04e15c64c3fae886e324fc195ea |
bafa223e38b74c3383154a8582efbfa8 |
+----------------------------------+-------+----------------------------------
+----------------------------------+
```

Instalación y configuración de Glance

Esta sección describe la instalación y configuración del módulo de OpenStack, Glance. Este servicio es el encargado de la gestión y registro de las imágenes que posteriormente se van a poder instanciar en máquinas virtuales.

Una infraestructura OpenStack solo necesita un servidor que ejecute el servicio Glance, en nuestra instalación el servicio se ejecutará en el controlador, es decir, en la máquina *jupiter*.

Se puede utilizar el servicio `Swift` como backend para el almacenamiento de las imágenes de Glance, pero por simplificación y debido a que no vamos a tener una gran cantidad de imágenes, vamos a utilizar `file` como backend, por lo que las imágenes se almacenarán en el directorio `/var/lib/glance/images`.

El servicio Glance cuenta con los siguientes componentes:

- glance-api
- glance-registry
- Base de datos.
- Repositorio de almacenamiento para las imágenes.

Para instalar y configurar el servicio seguimos los siguientes pasos:

1.- Instalamos todo el software necesario:

```
root@jupiter:~# apt-get install glance python-glanceclient
```

2.- Configuramos la base de datos para los dos servicios: glance-api y glance-registry cada uno en sus ficheros de configuración. Editamos:
/etc/glance/glance-api.conf

```
#sql_connection = sqlite:////var/lib/glance/glance.sqlite
sql_connection = mysql://glance:OpenStack_PASSWORD@172.20.253.190/glance
```

/etc/glance/glance-registry.conf

```
sql_connection = sqlite:////var/lib/glance/glance.sqlite
sql_connection = mysql://glance:OpenStack_PASSWORD@172.20.253.190/glance
```

3.- Borramos la base de datos SQLite que se crea por defecto:

```
root@jupiter:~# rm /var/lib/glance/glance.sqlite
```

4.- Creamos la base de datos que el servicio necesita en MySQL. Para ello iniciamos el cliente MySQL:

```
root@jupiter:~# mysql -u root -p
```

Desde el prompt de MySQL ejecutamos las siguientes sentencias:

```
mysql> CREATE DATABASE glance;
mysql> GRANT ALL PRIVILEGES ON glance.* to 'glance'@'127.0.0.1' IDENTIFIED BY
'OpenStack_PASSWORD';
mysql> GRANT ALL PRIVILEGES ON glance.* to 'glance'@'%' IDENTIFIED BY
'OpenStack_PASSWORD';
mysql> FLUSH PRIVILEGES;
```

5.- Creamos el esquema de la base de datos:

```
root@jupiter:~# glance-manage version_control 0
root@jupiter:~# glance-manage db_sync
```

6.- Añadimos las credenciales a los ficheros de configuración del servicio y modificamos las interfaces de escucha:
/etc/glance/glance-api.conf

```
...
# Address to bind the API server
#bind_host = 0.0.0.0
bind_host = 172.20.253.190
...
# Address to find the registry server
#registry_host = 0.0.0.0
registry_host = 172.20.253.190
...

[keystone_authtoken]
auth_host = 172.20.253.190
auth_port = 35357
```

```
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = OpenStack_PASSWORD
...

[paste_deploy]
flavor=keystone
```

/etc/glance/glance-registry.conf

```
...
# Address to bind the registry server
#bind_host = 0.0.0.0
bind_host = 172.20.253.190

[keystone_authtoken]
auth_host = 172.20.253.190
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = OpenStack_PASSWORD
...
[paste_deploy]
flavor=keystone
```

7.- Añadimos también las credenciales a los ficheros:
/etc/glance/glance-api-paste.ini

```
...
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
delay_auth_decision = true
auth_host=172.20.253.190
admin_user=glance
admin_tenant_name=service
admin_password=OpenStack_PASSWORD
...
```

/etc/glance/glance-registry-paste.ini

```
...
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host=172.20.253.190
admin_user=glance
admin_tenant_name=service
admin_password=OpenStack_PASSWORD
...
```

8.- Reiniciamos los demonios del servicio Glance:

```
root@jupiter:~# service glance-registry restart
root@jupiter:~# service glance-api restart
```

9.- Las credenciales ya están en el fichero `.keystonerc`, no hay que añadir nada más.

10.- Probamos el servicio.

Nada más instalar, no hay ninguna imagen dada de alta, pero el siguiente comando, que muestra la lista de imágenes disponible, nos debería dar una lista vacía:

```
root@jupiter:~# glance index
```

```
root@jupiter:~# glance image-list
```

Podemos asegurarnos que todo ha ido bien (incluyendo la autenticación con Keystone) si tras ejecutar el comando anterior se muestra una cabecera y ninguna imagen, el comando `echo $?` nos devuelve como código de salida cero.

Probando el servicio glance

Podemos probar el servicio Glance, y posteriormente Nova, a través de una imagen sencilla y pequeña como Cirros.

Nota:
CirrOS es una proyecto FLOSS cuyo objetivo principal es la construcción de una pequeña distribución GNU/Linux especializada en su ejecución en infraestructuras de Cloud Computing. Viene acompañada de herramientas para la depuración, desarrollo y despliegue en este tipo de infraestructuras.

Para crear la imagen de CirrOS seguimos estos pasos:

1.- Descargamos la imagen Cirros desde la siguiente URL:

https://launchpad.net/cirros/+download

La última versión en Launchpades la 0.3.0, concretamente este enlace:

https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img

Nos descargamos el fichero cirros-X.X.X-x86_64-disk.img (bootable qcow disk image for x86_64).

También podemos probar con la imagen:

http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img

Tenemos más imágenes prefabricadas en sitios web como: Ubuntu Cloud Images. (http://uec-images.ubuntu.com/).

2.- Damos la imagen de alta en Glance a través del siguiente comando:

```
root@jupiter:~# glance image-create --name="Cirros (test) 0.3.0 64 bits"
--is-public=true --container-format=bare --disk-format=qcow2 < cirros-
0.3.0-x86_64-disk.img
```

Como formato de contenedor (--container-format) casi siempre indicaremos bare, ya que raramente utilizaremos un formato de imagen que contenga metadatos, como formato de disco (--disk-format) indicaremos el formato de disco a elegir entre: *qcow2*, *raw*, *vhd*, *vmdk*, *vdi*, *iso*, *aki*, *ari*, y *ami* .

3.- Si todo es correcto, podremos ver ahora la imagen recién subida en la lista de imágenes que proporciona el servicio Glance:

```
root@jupiter:~# glance index
ID                                   Name                           Disk
Format          Container Format     Size
------------------------------------ ------------------------------
-------------------- -------------------- --------------
b75d513e-2fbf-44d2-88a6-05f085bc47b3 Cirros (test) 0.3.0 64 bits    qcow2
bare                         9761280
```

Glance pone a nuestra disposición otros comandos interesantes como los siguientes:

```
glance index            <<<< Devuelve un listado de las imágenes almacenadas.
Glance image-list       <<<< Devuelve un listado de las imágenes almacenadas.
glance image-show       <<<< Muestra los detalles de una imagen.
glance image-delete     <<<< Borra una imagen.
```

Instalación y configuración de Nova

Este capítulo describe la instalación y configuración del módulo de OpenStack: **Nova**.

Este servicio es el encargado de gestionar las instancias (máquinas virtuales) del Cloud, por lo que lo convierte en el servicio central de toda la infraestructura de Openstack.

Una infraestructura OpenStack puede ejecutar tantos nodos nova como desee. Como mínimo uno, esto hace que sea posible tener una infraestructura de OpenStack en un solo nodo, pero podemos desplegar tantos nodos de computación como servidores tengamos.

El máximo depende de diversos factores como la carga a soportar, la disponibilidad de servidores, factores económicos, ancho de banda de nuestra red, uso del Cloud, y un largo etcétera. En nuestra instalación el servicio nova-compute se ejecutará en cuatro nodos, concretamente en las máquinas io, europe, ganimedes y callisto.

El reparto exacto de servicios de Nova entre los nodos será el siguiente:

Nodo	Servicios
jupiter (controlador)	nova-api nova-consoleauth nova-scheduler nova-cert nova-novncproxy nova-conductor
io (nodo computación 1)	nova-api-metadata nova-compute nova-network nova-dhcpbridge
europe (nodo computación 2)	nova-api-metadata nova-compute nova-network nova-dhcpbridge
ganimedes	nova-api-metadata

(nodo computación 3)	nova-compute nova-network nova-dhcpbridge
callisto (nodo computación 4)	nova-api-metadata nova-compute nova-network nova-dhcpbridge

El nodo controlador ejecuta todos los servicios de Nova salvo los relativos a máquinas virtuales (VM) y red. Además el nodo controlador es el encargado de ejecutar todos los servicios básicos de la infraestructura del Cloud como son:

- MySQL.
- RabbitMQ
- NoVNC.

Los nodos de computación ejecutan dos servicios:

- **nova-compute.** A través de este servicio los nodos pueden ejecutar máquinas virtuales. En nuestro caso utilizando KVM como hipervisor.
- **nova-network**. En una instalación de Nova debe existir al menos un nodo encargado de ejecutar este servicio. En nuestra instalación todos los nodos de computación ejecutan nova-network ya que queremos que cada uno de los nodos sea responsable de la red de sus máquinas virtuales.
    - Para configuraciones avanzadas de red se recomienda el uso de **Neutron**.

Tanto para la instalación de Nova, como para la instalación del resto de servicios, partiremos de los repositorios oficiales de Havana para Ubuntu 12.04, basta con que sigamos los siguientes pasos según el nodo en el que nos encontremos.

Instalación en el nodo controlador

1.- Instalamos todo el software necesario:

```
root@jupiter:~# apt-get install nova-novncproxy novnc nova-api nova-ajax-
console-proxy nova-cert nova-conductor nova-consoleauth nova-doc nova-scheduler
python-novaclient
```

2.- Editamos el fichero /etc/nova/nova.conf para indicar la base de datos a utilizar y las credenciales de autenticación.

```
...
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:OpenStack_PASSWORD@172.20.253.190/nova

[keystone_authtoken]
auth_host = 172.20.253.190
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = OpenStack_PASSWORD
```

```
...
```

3.- Editamos también en el mismo fichero /etc/nova/nova.conf la configuración del broker RabbitMQ en la sección [DEFAULT]:

```
[DEFAULT]
...
rpc_backend = nova.rpc.impl_kombu
rabbit_host = 172.20.253.190
rabbit_password = OpenStack_PASSWORD
...
```

4.- Borramos la base de datos SQLite creada por defecto.

```
root@jupiter:~# rm /var/lib/nova/nova.sqlite
```

5.- Creamos la base de datos que el servicio necesita en MySQL. Para ello iniciamos el cliente MySQL:

```
root@jupiter:~# mysql -u root -p
```

Desde el prompt de MySQL ejecutamos las siguientes sentencias:

```
mysql> CREATE DATABASE nova;
mysql> GRANT ALL PRIVILEGES ON nova.* to 'nova'@'localhost' IDENTIFIED BY
'OpenStack_PASSWORD';
mysql> GRANT ALL PRIVILEGES ON nova.* to 'nova'@'%' IDENTIFIED BY
'OpenStack_PASSWORD';
mysql> FLUSH PRIVILEGES;
```

6.- Creamos el esquema de la base de datos que Nova necesita:

```
root@jupiter:~# nova-manage db sync
```

Ignoramos la salida del comando.

7.- Indicamos en el fichero de configuración los siguientes parámetros relativos a la configuración del proxy VNC, las IP deben ser de la red interna:

```
[DEFAULT]
...
my_ip=172.20.253.190
vncserver_listen=172.20.253.190
vncserver_proxyclient_address=172.20.253.190
...
```

8.- Configuramos la autenticación con Keystone, para ello modificamos el fichero /etc/nova/nova.conf añadiendo:

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

… y las credenciales en el fichero /etc/nova/api-paste.ini

```
...
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
```

```
auth_host = 172.20.253.190
auth_port = 35357
auth_protocol = http
auth_uri = http://172.20.253.190:5000/v2.0
admin_tenant_name = service
admin_user = nova
admin_password = OpenStack_PASSWORD
...
```

Nos aseguramos también de que el fichero /etc/nova/nova.conf contenga la directiva:

```
api_paste_config=/etc/nova/api-paste.ini
```

9.- El fichero de configuración de Nova en el controlador debe quedar así:

```
[DEFAULT]

# AUTHENTICATION
auth_strategy=keystone

# SCHEDULER
scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler

# VOLUMES
volumes_path=/var/lib/nova/volumes
iscsi_helper=tgtadm

# COMPUTE
libvirt_type=kvm
libvirt_use_virtio_for_bridges=true
connection_type=libvirt
api_paste_config=/etc/nova/api-paste.ini
start_guests_on_host_boot=true
resume_guests_state_on_host_boot=true

# APIS
ec2_private_dns_show_ip=True
enabled_apis=ec2,osapi_compute,metadata

# RABBITMQ
rpc_backend = nova.rpc.impl_kombu
rabbit_host=172.20.253.190
rabbit_password=OpenStack_PASSWORD

# GLANCE
image_service=nova.image.glance.GlanceImageService
glance_api_servers=172.20.253.190:9292

# NETWORK
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
my_ip=172.20.253.190
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# LOGS
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
```

```
# VNC
vnc_keymap=es
vncserver_listen=172.20.253.190
vncserver_proxyclient_address=172.20.253.190

[database]
connection=mysql://nova:OpenStack_PASSWORD@172.20.253.190/nova

[keystone_authtoken]
auth_host = 172.20.253.190
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = OpenStack_PASSWORD
```

10.- Revisamos los permisos del directorio /etc/nova y de sus ficheros:

```
root@jupiter:~# ls -ld /etc/nova/
drwxr-x--- 3 nova nova 4096 ene  3 17:44 /etc/nova/
root@jupiter:~# ls -l /etc/nova/
total 40
-rw-r----- 1 nova nova  3867 ene  3 17:43 api-paste.ini
-rw-r--r-- 1 nova nova  1346 oct 22 18:21 logging.conf
-rw-r----- 1 nova nova   878 ene  3 17:40 nova.conf
-rw-r----- 1 nova nova 15650 oct 17 16:15 policy.json
-rw-r--r-- 1 root root   934 oct 17 16:15 rootwrap.conf
drwxr-xr-x 2 root root  4096 ene  3 17:27 rootwrap.d
root@jupiter:~#
```

11.- Reiniciamos todos los demonios del servicio Nova. Para el inicio y parada de servicios en el nodo controlador podemos utilizar el siguiente script:

```
#!/bin/bash
SERVICIOS="nova-api nova-cert nova-consoleauth nova-scheduler nova-conductor
nova-novncproxy"

function iniciarServiciosNova()
{
      service rabbitmq-server start
      for servicio in $SERVICIOS
      do
            service $servicio start
      done
}

function pararServiciosNova()
{
      for servicio in $SERVICIOS
      do
            service $servicio stop
      done
      service rabbitmq-server stop
}


case $1 in
      start)
            echo "Iniciando todos los servicios nova"
```

```
            iniciarServiciosNova
      ;;

      stop)
            echo "Parando todos los servicios nova"
            pararServiciosNova
      ;;

      restart)
            service rabbitmq-server restart
            for servicio in $SERVICIOS
            do
                  service $servicio restart
            done
      ;;

      *) echo "Opción desconocida, uso $0 start|stop|restart"
      ;;
esac
```

8.- Comprobamos que podemos acceder a la lista de imágenes de Glance con el comando:

```
nova image-list
```

Verficamos también que los servicios de Nova están en ejecución:

```
root@jupiter:~# nova-manage service list
Binary           Host                      Zone             Status     State Updated_At
nova-cert        jupiter                   internal         enabled    :-)   2014-01-03 16:53:34
nova-consoleauth jupiter                   internal         enabled    :-)   2014-01-03 16:53:34
nova-scheduler   jupiter                   internal         enabled    :-)   2014-01-03 16:53:34
nova-conductor   jupiter                   internal         enabled    :-)   2014-01-03 16:53:34
```

9.- Es posible que algún servicio no arranque de inmediato, y muestre en el anterior listado la cadena "XXX" en el campo *State*, en ese caso esperamos pero comprobamos mientras que los servicio está realmente en ejecución con el comando:

```
root@jupiter:~# ps -eo pid,command | grep nova
 7422 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova.conf
 7431 /usr/bin/python /usr/bin/nova-cert --config-file=/etc/nova/nova.conf
 7440 /usr/bin/python /usr/bin/nova-consoleauth --config-file=/etc/nova/nova.conf
 7449 /usr/bin/python /usr/bin/nova-scheduler --config-file=/etc/nova/nova.conf
 7458 /usr/bin/python /usr/bin/nova-conductor --config-file=/etc/nova/nova.conf
 7467 /usr/bin/python /usr/bin/nova-novncproxy --config-file=/etc/nova/nova.conf
 7492 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova.conf
 7508 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova.conf
 7509 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova.conf
 7577 grep --color=auto nova
```

Nota :
El servicio nova-compute suele mostrar los caracteres XXX durante bastante tiempo .

Si el servicio no está en ejecución, algo ha ido mal (o bastante mal), por lo que tendremos que revisar los logs:

```
root@jupiter:~# less /var/log/nova/nova-compute.log
```

Instalación en los nodos de computación

La instalación en los nodos de computación es "un poco" más sencilla, tan solo hay que

seguir los siguientes pasos:

0.- Como pasos previos:

- Configuramos correctamente el nombre de cada uno de los nodos de computación.
- Configuramos NTP.

1.- Activamos los repositorios de Havana para Ubuntu e instalamos los siguientes paquetes:

```
root@io:~# apt-get install python-software-properties
root@io:~# add-apt-repository cloud-archive:havana
root@io:~# apt-get update
root@io:~# apt-get upgrade
root@io:~# apt-get dist-upgrade
root@io:~# reboot
```

```
root@io:~# apt-get install nova-compute-kvm python-guestfs python-mysqldb
```

...a la pregunta "¿Desea actualizar o crear una instancia supermin ahora?" responder "Yes".

Instalamos también el siguiente paquete para evitar un error en los logs por parte de libvirtd:

```
apt-get install pm-utils
```

2.- Debido a este bug (https://bugs.launchpad.net/ubuntu/+source/linux/+bug/759725), el kernel no puede ser leído por otro usuario que no sea root. El servicio Nova debe poder leer el fichero del kernel, le damos permiso de lectura:

```
root@io:~# dpkg-statoverride --update --add root root 0644 /boot/vmlinuz-`uname
-r`
```

Esta solución es temporal, para que sea permanente creamos el fichero: `/etc/kernel/postinst.d/statoverride` con el siguiente contenido:

```
#!/bin/sh
version="$1"
# passing the kernel version is required
[ -z "${version}" ] && exit 0
dpkg-statoverride --update --add root root 0644 /boot/vmlinuz-${version}
```

... y le damos permiso de ejecución:

```
chmod +x /etc/kernel/postinst.d/statoverride
```

3.- Editamos el fichero /etc/nova/nova.conf y realizamos los siguientes cambios relativos a la autenticación con Keystone, el acceso a base de datos, la configuración de la cola de mensajes y la dirección del servidor Glance:

```
[DEFAULT]
...
auth_strategy=keystone
...
rpc_backend=nova.rpc.impl_kombu
rabbit_host=172.20.253.190
```

```
rabbit_password=OpenStack_PASSWORD

[database]
# The SQLAlchemy connection string used to connect to the database
connection=mysql://nova:OpenStack_PASSWORD@172.20.253.190/nova
```

4.- Configuramos Nova para proporcionar acceso remoto a las consolas de las máquinas virtuales:

```
[DEFAULT]
...
my_ip=172.20.253.191
vnc_enabled=True
vnc_keymap=es
vncserver_listen=172.20.253.191
vncserver_proxyclient_address=172.20.253.191
novncproxy_base_url=http://jupiter.iescierva.net:6080/vnc_auto.html

[database]
...
```

Siendo jupiter.iescierva.net el nombre DNS del controlador que se corresponde con la dirección IP 172.20.254.190 (red de acceso público).

5.- Editamos el fichero /etc/nova/api-paste.ini para añadir las credenciales:

```
...
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = 172.20.253.190
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = OpenStack_PASSWORD
# signing_dir is configurable, but the default behavior of the authtoken
# middleware should be sufficient.  It will create a temporary directory
# in the home directory for the user the nova process is running as.
#signing_dir = /var/lib/nova/keystone-signing
# Workaround for https://bugs.launchpad.net/nova/+bug/1154809
auth_version = v2.0
```

6.- El fichero /etc/nova/nova.conf debe quedar así:

```
[DEFAULT]

# AUTHENTICATION
auth_strategy=keystone

# SCHEDULER
scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler

# VOLUMES
volumes_path=/var/lib/nova/volumes
iscsi_helper=tgtadm

# COMPUTE
libvirt_type=kvm
libvirt_use_virtio_for_bridges=true
connection_type=libvirt
```

```
api_paste_config=/etc/nova/api-paste.ini
start_guests_on_host_boot=true
resume_guests_state_on_host_boot=true

# APIS
ec2_private_dns_show_ip=True
enabled_apis=ec2,osapi_compute,metadata

# RABBITMQ
rpc_backend = nova.rpc.impl_kombu
rabbit_host=172.20.253.190
rabbit_password=OpenStack_PASSWORD

# GLANCE
glance_host=172.20.253.190

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
network_size=254
allow_same_net_traffic=False
multi_host=True
send_arp_for_ha=True
share_dhcp_address=True
force_dhcp_release=True
flat_network_bridge=br100
flat_interface=bond0.61
public_interface=bond0.61
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
my_ip=172.20.253.191
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# LOGS
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True

# VNC
vnc_enabled=true
vnc_keymap=es
vncserver_listen=172.20.253.191
vncserver_proxyclient_address=172.20.253.191
novncproxy_base_url=http://jupiter.iescierva.net:6080/vnc_auto.html

[database]
connection=mysql://nova:OpenStack_PASSWORD@172.20.253.190/nova

[keystone_authtoken]
auth_host = 172.20.253.190
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = OpenStack_PASSWORD
```

7.- Reiniciamos el servicio:

```
root@io:~# service nova-compute restart
```

8.- Una vez iniciados todos los servicios, la salida del comando nova-manage en el nodo controlador debería ser la siguiente:

```
root@jupiter:~# nova-manage service list
Binary           Host                                 Zone             Status     State Updated_At
nova-cert        jupiter                              internal         enabled    :-)   2014...
nova-consoleauth jupiter                              internal         enabled    :-)   2014...
nova-scheduler   jupiter                              internal         enabled    :-)   2014...
nova-conductor   jupiter                              internal         enabled    :-)   2014...
nova-compute     io                                   nova             enabled    :-)   2014...
root@jupiter:~#
```

9.- Borramos la base de datos SQLite del nodo:

```
root@io:~# rm /var/lib/nova/nova.sqlite
```

Configuración de la red en los nodos de computación: nova-network

Para la configuración de prueba vamos a poner en marcha el servicio nova-network en la modalidad FlatDCHP. Para ello:

1.- Instalamos el servicio:

```
root@io:~# apt-get install nova-network nova-api-metadata
```

2.- Editamos el fichero /etc/nova/nova.conf y añadimos las directivas de red en la sección default, tal como hemos hecho antes:

```
[DEFAULT]
...
# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
network_size=254
allow_same_net_traffic=False
multi_host=True
send_arp_for_ha=True
share_dhcp_address=True
force_dhcp_release=True
flat_network_bridge=br100
flat_interface=bond0.61
public_interface=bond0.61
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
my_ip=172.20.253.191
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
...
```

3.- Reiniciamos el servicio:

```
root@io:~# service nova-network restart
```

4.- Creamos una red, esto se hace desde el controlador:

```
root@jupiter:~# source .keystonerc
root@jupiter:~# nova network-create vmnet --fixed-range-v4=10.0.0.0/24
--bridge-interface=br100 --multi-host=T
```

5.- Verificamos la creación de la red:

```
root@jupiter:~# nova network-list
+--------------------------------------+-------+-------------+
| ID                                   | Label | Cidr        |
+--------------------------------------+-------+-------------+
| 1947900a-af23-4a95-8788-ee5d2f420270 | vmnet | 10.0.0.0/24 |
+--------------------------------------+-------+-------------+
root@jupiter:~#
```

¿Cómo hacer que el rango de IP de las VMs coincida con el externo?

Con nova-network se puede conseguir que el rango de IPs que se asigna a las máquinas virtuales coincida con el rango de direcciones de la red externa a la que están conectadas los nodos de computación.

Supongamos que los nodos de computación están además conectados a la red 172.18.0.0/16 a través de una nueva interfaz, si las interfaces de los nodos de computación es la siguiente:

Red	Red IP	Direcció IP	Interfaz
Red externa con acceso a Internet	- red externa- 192.168.166.0/24	192.168.166.191	eth0
Red de gestión de los nodos del cloud	172.20.253.0/24	172.20.253.191	eth1
Red para el tráfico de las VMs	--no aplicable--	Direcciones VMs	eth2 (br100)
Red externa con acceso a Internet a la que conectar las VMs	172.18.0.0/24	172.18.0.1 – 172.18.0.254	eth2

... bastaría con que hiciéramos los siguientes cambios:

1.- Configuramos en el fichero /etc/nova/nova.conf la misma interfaz para el tráfico público e interno:

```
#my_ip=
flat_network_bridge=br100
flat_interface=eth2
public_interface=eth2
```

A la interfaz eth2 no se le configura ninguna dirección.

2.- Creamos la red en el mismo rango que la red externa:

```
nova network-create vmnet --fixed-range-v4=172.18.0.0/24 --bridge-
interface=br100 --multi-host=T
```

Administración de las redes

Podemos mostrar la(s) red(es) a través de los comandos:

```
nova-manage network list
```

```
nova network-list
```

Podemos borrar una red a través de su UUID con el comando:

```
nova-manage network delete --uuid=537da331-7c9a-4dff-9462-e918ff83f352
```

Podemos borrar la red creada con los siguientes comandos:

1.- Borrar las instancias conectadas.

2.- Desasociar la red del proyecto.

```
nova-manage network modify CIDR_range --disassociate-project
```

3.- Borrar la red:

```
nova-manage network delete CIDR_range
```

Arranque de instancias

1.- Generamos las claves SSH para que el acceso a las VM se realice sin pedir contraseña. La clave se inyecta en la VM al iniciarse.

```
root@jupiter:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
...
root@jupiter:~# nova keypair-add --pub_key .ssh/id_rsa.pub mykey
root@jupiter:~# nova keypair-list
+-------+-------------------------------------------------+
| Name  | Fingerprint                                     |
+-------+-------------------------------------------------+
| mykey | 57:5c:5c:c9:db:ac:b5:02:bd:4c:eb:07:c7:87:ca:d6 |
+-------+-------------------------------------------------+
root@jupiter:~#
```

2.- Obtenemos los "flavors" actuales para seleccionar con el que queremos lanzar la instancia, por ejemplo el 1 para una imagen Cirros.

```
root@jupiter:~# nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
root@jupiter:~#
```

3.- Obtenemos el ID de la imagen a lanzar:

```
root@jupiter:~# nova image-list
+--------------------------------------+-----------------------------+--------+--------+
| ID                                   | Name                        | Status | Server |
```

```
+--------------------------------------+------------------------------+--------+--------+
| 8589a87c-19a9-4766-b580-a495cb8f8f3e | Cirros (test) 0.3.0 64 bits  | ACTIVE |        |
| 769aa37c-0741-4efb-9ef5-4a6b85ff62e3 | Ubuntu 12.04.3 Server 64 bits | ACTIVE |        |
+--------------------------------------+------------------------------+--------+--------+
root@jupiter:~#
```

4.- Para poder utilizar SSH y hacer pings a las máquinas tenemos que configurar los grupos de reglas de seguridad:

```
root@jupiter:~# nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
...
root@jupiter:~# nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
...
root@jupiter:~# nova secgroup-list-rules default
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
root@jupiter:~#
```

5.- Lanzamos la instancia:

```
root@jupiter:~# nova boot --flavor 1 --key_name mykey --image 8589a87c-19a9-
4766-b580-a495cb8f8f3e --security_group default cirrOS-01
```

Comprobamos que la instancia se ha lanzado con éxito:

```
root@jupiter:~# nova list
+--------------------------------------+-----------+--------+------------+-------------
+----------------+
| ID                                   | Name      | Status | Task State | Power State | Networks
|
+--------------------------------------+-----------+--------+------------+-------------
+----------------+
| 9bcf5c69-c3b3-45b1-96ef-728527cc2200 | cirrOS-01 | ACTIVE | None       | Running     |
vmnet=10.0.0.2 |
+--------------------------------------+-----------+--------+------------+-------------
+----------------+
```

6.- Podemos acceder a la instancia desde cualquier nodo que ejecute nova-network, la máquina tiene la IP 10.0.0.2, y el nodo de computación ha configurado la dirección 10.0.0.1 en el bridge br100:

```
root@io:~# ifconfig br100
br100     Link encap:Ethernet  HWaddr 00:25:90:6a:42:1a
          inet addr:10.0.0.1  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::5874:12ff:feb6:c845/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:74 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7763 (7.7 KB)  TX bytes:8021 (8.0 KB)

root@io:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=0.538 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=0.407 ms
^C
--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.407/0.472/0.538/0.069 ms
```

```
root@io:~#
```

7.- Las credenciales para iniciar sesión en las máquinas CirrOS son:
Usuario: cirros
Password: cubswin:)

8.- En el caso de utilizar direcciones que coincidan con un rango externo, en todos los nodos de computación se habrá configurado en el bridge la misma dirección, la 172.18.0.1:

```
root@io:~# ifconfig br100
br100     Link encap:Ethernet  HWaddr 00:25:90:6a:42:19
          inet addr:172.18.0.1  Bcast:172.18.255.255  Mask:255.255.0.0
          inet6 addr: fe80::5c48:aff:fe79:de04/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:568287 errors:0 dropped:15 overruns:0 frame:0
          TX packets:49459 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2443281065 (2.4 GB)  TX bytes:7824490 (7.8 MB)

root@io:~# ifconfig bond0.61
bond0.61  Link encap:Ethernet  HWaddr 00:25:90:6a:42:19
          inet6 addr: fe80::225:90ff:fe6a:4219/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:475535 errors:0 dropped:0 overruns:0 frame:0
          TX packets:79438 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:286503028 (286.5 MB)  TX bytes:8073622 (8.0 MB)

root@io:~#
```

Instalación de Horizon

Para la instalación de Horizon, la interfaz administrativa vía web de OpenStack, seguimos los siguientes pasos. Vamos a suponer que realizamos la instalación en el controlador.

1.- Instalamos los paquetes:

```
apt-get install memcached libapache2-mod-wsgi openstack-dashboard
```

En Ubuntu desinstalamos además:

```
apt-get remove --purge openstack-dashboard-ubuntu-theme
```

2.- Realizamos las siguientes modificaciones en el fichero /etc/openstack-dashboard/local_settings.py:

```
...
CACHES = {
   'default': {
       'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',
       'LOCATION' : '127.0.0.1:11211',
   }
}
...
#OPENSTACK_HOST = "127.0.0.1"
OPENSTACK_HOST = "172.20.253.190"
```

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v2.0" % OPENSTACK_HOST
#OPENSTACK_KEYSTONE_DEFAULT_ROLE = "Member"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "_member_"
...
TIME_ZONE = "UTC"
...
# By default, validation of the HTTP Host header is disabled.  Production
# installations should have this set accordingly.  For more information
# see https://docs.djangoproject.com/en/dev/ref/settings/.
ALLOWED_HOSTS = '*'
```

3.- Reiniciamos los servicios asociados:

```
service apache2 restart
service memcached restart
```

Explotación de OpenStack

Creación de un nuevo proyecto

Para crear un nuevo proyecto seguimos los siguientes pasos, los podemos realizar a través de la línea de comandos o a través de la interfaz gráfica.

En línea de comandos ejecutaríamos los siguientes comandos:

1.- Creamos el nuevo proyecto (tenant).

```
root@jupiter:~# keystone tenant-create --name dam1 --description "DAM1 Tenant"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           DAM1 Tenant            |
|   enabled   |               True               |
|      id     | ecdcf22881b34ed1b4b82e7a2e20e01c |
|     name    |               dam1               |
+-------------+----------------------------------+
```

2.- Creamos un usuario (o varios) que pertenezcan al tenant anterior, le asignamos el rol por defecto, en este caso no le asignamos el rol administrador (admin):

```
root@jupiter:~# keystone user-create --name dam1 --pass PASSWORD_dam1 --email
alex@iescierva.net
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |        alex@iescierva.net        |
| enabled  |               True               |
|    id    | 7ebc57544684831b85e136f43e48ead |
|   name   |               dam1               |
+----------+----------------------------------+

root@jupiter:~# keystone user-role-add --user=dam1 –tenant=dam1 --role=_member_
```

3.- Creamos el fichero con las credenciales del usuario en la cuenta que nos interese.

```
alex@jupiter:~$ cat .keystonercDAM1
# Variables de entorno para la autenticación de OpenStack
```

```
OS_USERNAME=dam1
OS_PASSWORD=PASSWORD_dam1
OS_TENANT_NAME=dam1
OS_AUTH_URL=http://172.20.253.190:35357/v2.0

export OS_USERNAME OS_PASSWORD OS_TENANT_NAME OS_AUTH_URL
alex@jupiter:~$
```

Y nos autenticamos:

```
alex@jupiter:~$ . .keystonercDAM1
```

4.- Generamos una nueva clave SSH copiando la nuestra:

```
alex@jupiter:~$ nova keypair-add --pub_key .ssh/id_rsa.pub mykey
alex@jupiter:~$ nova keypair-list
+-------+-------------------------------------------------+
| Name  | Fingerprint                                     |
+-------+-------------------------------------------------+
| mykey | e0:bd:47:2e:bf:50:71:a6:f1:50:8e:8a:12:1d:09:ba |
+-------+-------------------------------------------------+
```

5.- Definimos los grupos de seguridad del tenant.

```
alex@jupiter:~$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
alex@jupiter:~$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
alex@jupiter:~$ nova secgroup-list-rules default
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
alex@jupiter:~$
```

6.- Arrancamos una imagen de prueba:

```
alex@jupiter:~$ nova boot --flavor 1 --key_name mykey --image 7481197b-3714-
4285-90c3-26968192ac63 --security_group default cirrOS-01
```

Script para la creación de máquinas virtuales

A la hora de crear máquinas virtuales para los alumnos viene muy bien tener algún script capaz de crear las máquinas virtuales añadiendo información como el nombre de la instancia y el alumno al que pertenece.

Si partimos de un fichero de texto con el siguiente formato:

```
alex@jupiter:~$ cat alumnos.txt
Apellido1 Apellido1, Nombre1
Apellido2 Apellido2, Nombre2
Apellido3 Apellido3, Nombre3
Apellido4 Apellido4, Nombre4
...
Apellido20 Apellido20, Nombre20
```

Podemos crear a todos los usuarios a través del siguiente script:

```
alex@jupiter:~$ ruby bin/createUSERS.rb --help
Script for the batch creation of Virtual Machines

Usage: createUSERS.rb [options] filename
    -h, --help                       Show how to use the command
    -s, --simulate                   Don't create the users, only show the
commamnds to execute
alex@jupiter:~$
```

Para comprobar los comandos a ejecutar y cómo se comportaría el script:

```
ruby bin/createUSERS.rb -s bin/alumnos.txt
```

Y crear a los usuarios con una llamada al script de esta forma:La creación se

```
ruby bin/createUSERS.rb bin/alumnos.txt
```

El código del script Ruby es el siguiente:

```
#!/usr/bin/ruby

# Alejandro Roca Alhama
# Script to create VM for our students
# Version 0.2.
# Last modifications: 12/may/2014.

require 'optparse'

# Check command line parameters: file with the users to create and the group
name.
options = {}
option_parser = OptionParser.new do |opts|
  executable_name = File.basename($PROGRAM_NAME)
  opts.banner = "Script for the batch creation of Virtual Machines

Usage: #{executable_name} [options] filename"

  opts.on('-h', '--help', 'Show how to use the command') do
    options[:help] = true
  end

  opts.on('-s', '--simulate', 'Don\'t create the users, only show the commamnds
to execute') do |opts|
    options[:simulate] = true
  end

  #opts.on('-f FILE', 'File with the users to create') do |file|
  #  options[:file] = file
  #end
end
```

```
option_parser.parse!
if options[:help]
  puts option_parser.help
  exit(0)
elsif ARGV.empty?
  puts 'You must provide a filename'
  puts
  puts option_parser.help
  exit(0)
else
  filename = ARGV[0]
end

# Read the file. From the file we must generate instance name and description
id = '00'
group = 'asir1'
begin
  IO.foreach(filename) do |line|
    user = line.chomp.split(',')
    last_name = user[0]
    first_name = user[1]
    # To generate the id with zeros left padding we can use string.rjust(3,
"0")
    command = 'nova boot --flavor 6 --key_name mykey --image 89a1f50c-89f6-
482d-b17e-2146a074b085 --security_group default '
    command += "--meta name='#{last_name}, #{first_name}' "
    command += "#{group}-#{id}"
    if options[:simulate]
      puts command
    else
      system(command)
    end
    id = id.next
  end
rescue Errno::ENOENT
  STDERR.puts "File #{filename} doesn't exist"
end
```