Administración de OpenStack

1.0 (Jun 2012, 25)

Proyecto de innovación

Implantación y puesta a punto de la infraestructura
de un cloud computing privado para
el despliegue de servicios en la nube

IES Gonzalo Nazareno
Dos Hermanas (Sevilla)

IES Los Albares
Cieza (Murcia)

IES La Campiña
Arahal (Sevilla)

IES Ingeniero de la Cierva
Murcia

Cofinanciado por:

Administración de OpenStack

Carlos Álvarez Barba
Miguel Ángel Ibáñez Mompeán
Alberto Molina Coballes
Jesús Moreno León
José Domingo Muñoz Rodríguez
Cayetano Reinaldos Duarte
Alejandro Roca Alhama

1.0 (25-06-2012)

Guía de introdución a la instalación, configuración y explotación de OpenStack.

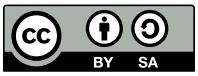


Table of Contents

1. OpenStack 1
¿Qué es el "Cloud Computing"? 1
¿Qué es OpenStack? 1
Componentes de OpenStack 2
Arquitectura de OpenStack 3
Arquitectura conceptual 4
Arquitectura lógica 5
Dashboard 6
Compute 7
Object Storage 8
Image Storage 9
Identity 9
Nuevos componentes 10
Introducción a OpenStack Compute 10
Hipervisores 10
Usuarios y proyectos (Tenants) 11
Imágenes e instancias 11
Almacenamiento de bloques y OpenStack Compute 12
Introducción a OpenStack Keystone 12
Introducción al módulo keystone 12
¿Qué es el ADMIN_TOKEN y para qué se utiliza? 13
Uso del ADMIN_TOKEN 13
Introducción a OpenStack Glance 14
Introducción a OpenStack Swift 14
2. Instalación de OpenStack en Debian GNU/Linux Wheezy 15
Pasos previos 15
Nombres de los equipos 15
Esquema de red 16
Direcciones IP de los equipos 17
Instalación y configuración inicial de MySQL 17
Instalación de otros paquetes 18
Sincronización de la hora de los equipos con ntp 18
Instalación manual de python-prettytable 19
Keystone 19
Instalación de keystone 19
Configuración de keystone 19
Creación de proyectos, usuarios y roles 19
Configuración de los servicios 21
Método de autentificación 22
Utilización de la API 22
Glance 23
Introducción al módulo glance 23
Instalación de Glance 23
Configuración de Glance 23
Método de autentificación y prueba de funcionamiento 24
Nova en el nodo controlador 25
Introducción al módulo nova 25
Instalación 25

Configuración 25
Nova en los nodos de computación 31
Instalación 31
Horizon (dashboard) 31
Instalación 31
3. Instalación de OpenStack en GNU/Linux Ubuntu 12.04 33
Introducción 33
Prerrequisitos 33
Servicios y configuración básica 34
Nombres de los equipos y configuración de la red 34
Configuración del bonding y de las VLAN 38
Instalación de Ubuntu 12.04. 45
NTP 47
MySQL 47
Instalación de KeyStone 48
Instalación y configuración 49
Instalación de Glance 55
Instalación y configuración 55
Instalación de Nova 57
Instalación y configuración 58
Instalación de Nova: problemas y puntualizaciones 69
Instalación de Horizon 71
Instalación y configuración 71
Instalación del cliente administrativo 72
4. Gestión de la identidad 74
Autenticación con LDAP 74
Configuración del directorio 74
Referencias 75
5. Gestión de imágenes 76
Formato de Disco y Contenido 76
Formato del disco 76
Formato del contenedor 77
Imágenes disponibles en Internet 77
Procedimiento general de provisión de imágenes 78
Creación de imágenes GNU/Linux 78
Creación de imágenes Windows 79
Amazon Machine Images 80
6. Networking 82
7. Gestión de Volúmenes 83
Sobre iSCSI 84
Instalación y configuración de nova-volume 88
Comandos Nova para la gestión de volúmenes 94
Resolución de problemas 97
Controladores de Volúmenes (Volume Drivers) 100
Iniciando una instancia desde un volumen 101
8. Administración de OpenStack 102
Gestión de instancias 102
Gestión de plantillas 106
Gestión de usuarios 107
Gestión de proyectos 108
Gestión de roles y asignación a usuarios 109

Gestión de cuotas 110
Monitorización de instancias y nodos 111
Servicios VNC y VNCProxy 113

List of Figures

3.1. Infraestructura Cloud Computing (IES Cierva/IES Los Albares) 36
3.2. Infraestructura de nuestro Cloud Computing (IES Cierva/IES Los Albares) 38

List of Tables

2.1. Direcciones IP de los equipos del cloud 17
3.1. LANs Virtuales del Cloud Computing 37
3.2. Usuarios, proyectos y roles 50

1. OpenStack

OpenStack es una colección de tecnologías Open Source que proporcionan un software para el despliegue escalable de un cloud computing. OpenStack proporciona Infraestructura como Servicio ó IaaS (Infrastructure as a Service) y es un proyecto que se inició en el año 2010 por la empresa *Rackspace Cloud* y por la agencia espacial norteamericana, NASA. Actualmente más de 150 empresas se han unido al proyecto, entre las que se encuentran empresas tan importantes como AMD, Intel, Canonical, SUSE Linux, Red Hat, IBM, Dell, HP, Cisco, etc. OpenStack es software libre bajo los términos de la licencia Apache.

Actualmente OpenStack desarrolla dos proyectos relacionados: *OpenStack Compute*, que proporciona recursos computacionales a través de máquinas virtuales y gestión de la red, y *OpenStack Object Storage*, que proporciona un servicio de almacenamiento de objetos redundante y escalable. Muy relacionado con el proyecto "OpenStack Compute" tenemos otros proyectos complementarios como Keytone ó Glance que describiremos en breve.

OpenStack puede ser utilizado por cualquiera organización que busque desplegar un cloud de gran escala tanto para uso privado como público. OpenStack es un proyecto intersante casi para cualquier tipo de organización: pequeñas y medianas empresas, administración, grandes corporaciones, proveedores de servicio, empresas de valor añadido, centros de cálculo y un largo etcétera.

¿Qué es el "Cloud Computing"?

¿Qué es OpenStack?

Básicamente OpenStack es un software Open Source usado para la construcción de clouds públicas y privadas. OpenStack representa tanto a una comunidad y un proyecto de Software Libre, como un software para ayudar a las organizaciónes a ejecutar sus propios clouds para computación o almacenamiento virtual.

Desde el punto de vista de software, OpenStack es una colección de proyectos de software libre mantenidos por la comunidad que incluyen varios componentes, siendo los más importantes:

- OpenStack Compute, con nombre en clave **Nova**.
- OpenStack Object Storage, con nombre en clave **Swift**.
- OpenStack Image Service, con nombre en clave **Glance**.

A través de estos servicios, OpenStack proporciona una completa plataforma operativa para la administración y gestión de clouds.

Definir a OpenStack es mucho más sencillo una vez que los principales conceptos sobre Computación en la Nube se hacen más aparentes. La misión principal del proyecto es proporcionar un software que cubra el ciclo completo de este tipo de despliegues y que

proporcione el poder desplegar de forma sencilla, escalable, elástica y de cualquier tamaño, tanto clouds públicos como clouds privados.

Para alguien que se acerca por primera vez a OpenStack, esta aproximación puede resultar difícil y abrumadora. Hay conceptos difíciles, y pueden surgir multitud de dudas, en cuanto a instalación, despliegue y uso. Lo bueno es que al tratarse de un proyecto abierto, un proyecto de Sofwtare Libre mantenido por y para la Comunidad, hay un montón de documentación, guías, foros, para ayudar a resolver cualquier problema o incidencia que surja. Este propio documento, trata precisamente de allanar el camino de todo aquel que se inicie en el despliegue de este tipo de soluciones.

OpenStack es muy joven, por lo que el propio software, e incluso la propia documentación están en constante revisión. Hay que estar muy atento a la página oficial del proyecto:

- Home: OpenStack Open Source Cloud Computing Software.

Componentes de OpenStack

Actualmente, hay cinco componentes principales de OpenStack: *Compute*, *Object Storage*, *Identity*, *Image Service* y *Dashboard*.

OpenStack Compute es el controlador de la estructura básica del Cloud. Es el encargado de iniciar las instancias (máquinas virtuales) de los usuarios y grupos. También es el servicio encargado de la gestión de la red virtual para cada instancia o para las múltiples instancias que formen parte de un proyecto (tenant).

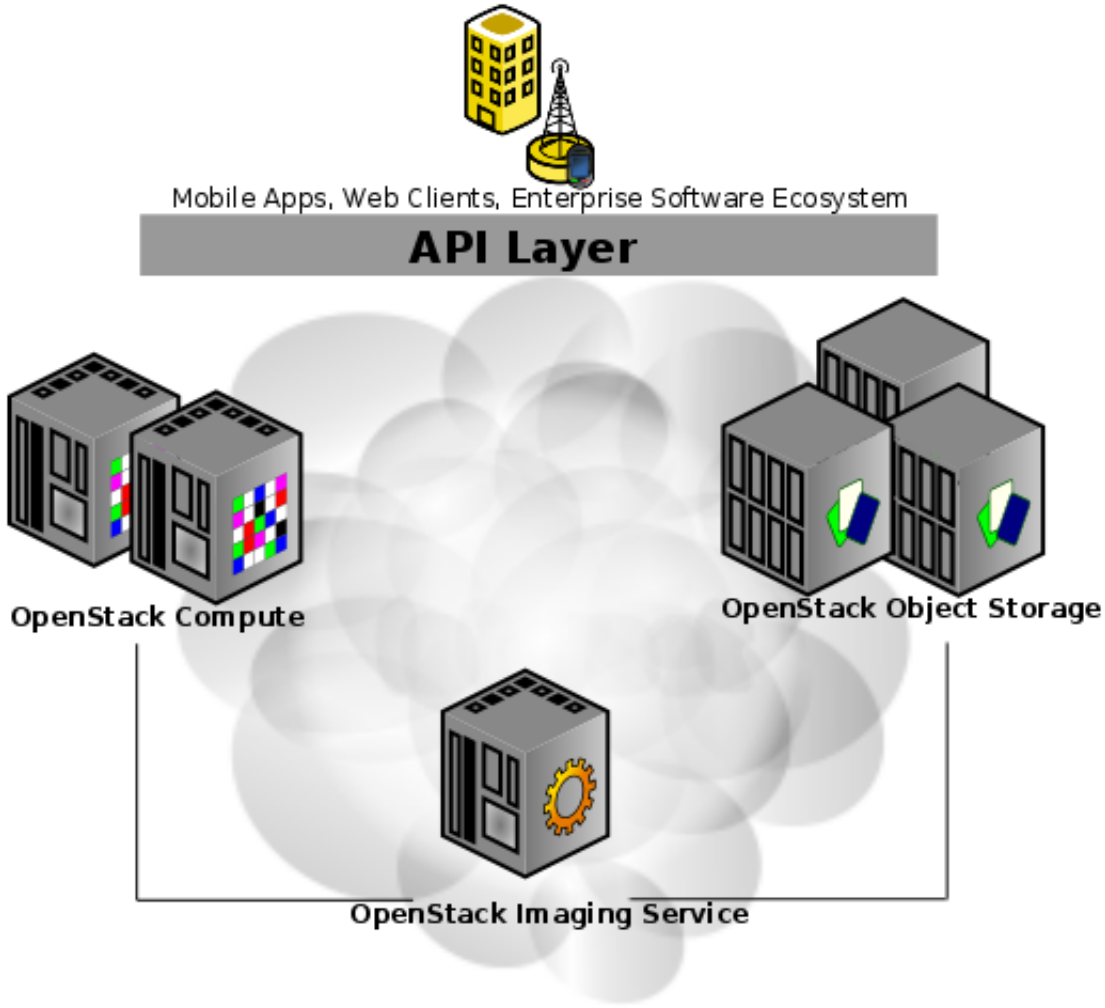
OpenStack Object Storage es el servicio encargado del almacenamiento masivo de objetos a través de un sistema escalable, redundante y tolerante a fallos. Las posibles aplicaciones de Object Storage son numerosas, como por ejemplo: almacenamiento simple de ficheros, copias de seguridad, almacenamiento de streamings de audio/vídeo, almacenamiento secundario/terciario, desarrollo de nuevas aplicaciones con almacenamiento integrado, etc.

OpenStack Identity Service es un servicio usado para la autenticación entre el resto de componentes. Este servicio utiliza un sistema de autenticación basado en tokens y se incorporó en la versión 2012.1 de OpenStack.

OpenStack Image Service es un servicio para la búsqueda y recuperación de imágenes de máquinas virtuales. Este servicio puede almacenar las imágenes directamente o utilizar mecanismos más avanzados como: usar Object Storage como servicio de almacenamiento, usar Amazon's Simple Storage Solution (S3) directamente, ó usar Object Storage como almacenamiento intermedio de S3.

OpenStack Dashboard es un panel web para el manejo de instancias y volúmenes. Este servicio es realmente una aplicación web desarrollada en django que permite comunicarse con las diferentes APIs de OpenStack de una forma sencilla. OpenStack Dashboard es fundamental para usuarios noveles y en general para realizar acciones sencillas sobre las instancias.

El siguiente diagrama muestra las relaciones entre los componentes principales (Nova, Glance y Swift), cómo están relacionados y cómo pueden cumplir los objetivos propuestos por OpenStack para el despliegue de infraestructuras de cloud computing.



Arquitectura de OpenStack

Antes de revisar los componentes de OpenStack, conviene revisar un poco la historia del proyecto. Fundado en 2010 por la empresa Rackspace y por la NASA, el proyecto ha tenido hasta la fecha cuatro versiones, actualmente se encuentra en su quinta revisión, lanzada en abril con el nombre en clave *Essex* (ó 2012.1). Originalmente el proyecto consistía en tan solo tres servicios principales:

- Object Store (*"Swift"*): proporciona almacenamiento de objetos. Swift nos permite almacenar y/o recuperar ficheros, pero no montar directorios como un sistema de ficheros basado en NFS ó CIFS. Varias compañías proporcionan servicios de almacenamiento comercial basado en Swift, tales como la propia Rackspace (desde la que se inició este proyecto), KT, ó Internap entre otras. Una página web puede fácilmente mostrar imágenes almacenadas en un servidor Swift.

- Image (*"Glance"*): proporciona un catálogo y un repositorio de imágenes de discos virtuales. Muy utilizado por Nova y de forma casi exclusiva, aunque es un servicio

técnicamente opcional, cualquier infraestructura de cloud de un tamaño considerable lo necesita.

- Compute (*"Nova"*): proporciona máquinas virtuales bajo demanda. Similar al servicio EC2 de Amazon. Nova también es capaz de proporcionar gestión de volúmenes de discos a través de uno de sus servicios, de forma similar al EBS (Elastic Block Service).

Estos son los servicios básicos hasta la versión Essex de OpenStack, que además incluye dos servicios básicos adicionales:

- Dashboard (*"Horizon"*): proporciona una interfaz de usuario modular, basada en la web, para la gestión de todos los servicios de OpenStack.

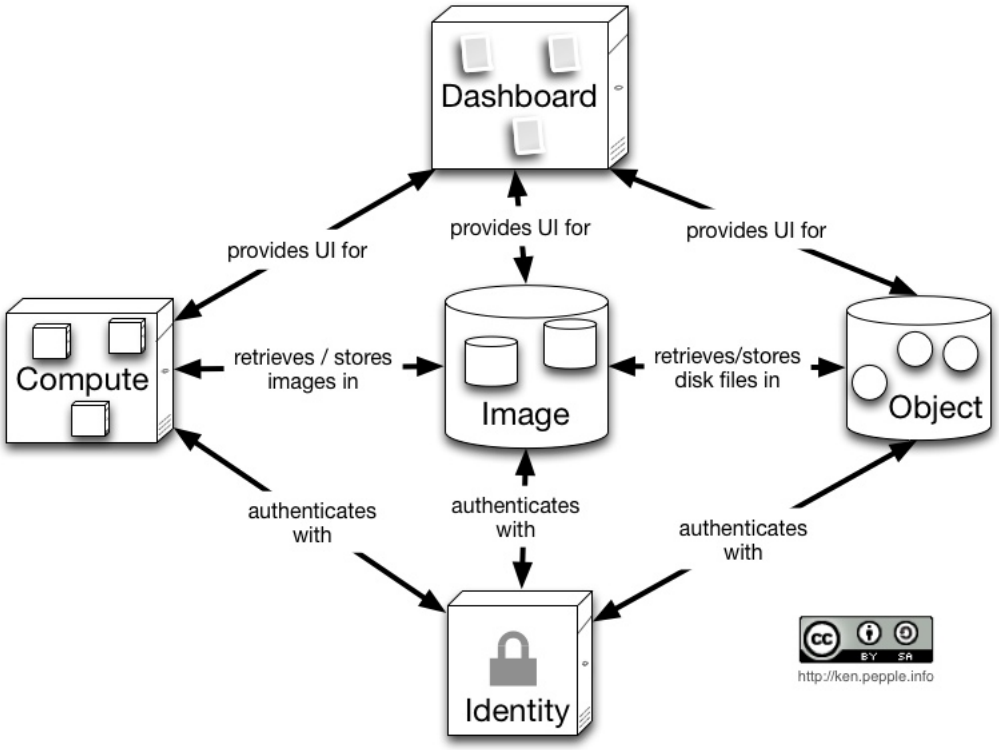
- Identity (*"Keystone"*): proporciona servicios de autenticación y autorización a todos los servicios de OpenStack. Keystone proporciona, además, un catálogo de los servicios ofrecidos en un despliegue de Openstack en concreto.

Estos dos proyectos adicionales, ya incluidos en la última versión de OpenStack, proporcionan una infraestructura adicional para los tres proyectos originales. Básicamente servicios de autenticación y un frontal basado en web.

Arquitectura conceptual

Desde una perspectiva global, OpenStack está diseñado para "entregar un sistema operativo para el despliegue de clouds masivamente escalables". Para poder lograrlo, cada uno de los servicios que conforman OpenStack están diseñados para trabajar conjuntamente y poder proporcionar una *Infraestructura* como Servicio (IaaS, Infrastructure as a Service) completa. Esta integración se consigue a través de APIs (Application Programming Interfaces) que cada servicio ofrece, y que cada servicio puede consumir. Mientras que estas APIs permiten a cada uno de los servicios utilizar el resto, también permiten al desarrollador poder reemplazar cualquier servicio con otra implementación, siempre y cuando se respeten estas APIs. Dichas APIs también se encuentran disponibles para el usuario final del cloud.

Conceptualmente, se pueden representar las relaciones entre los servicios a través del siguiente diagrama:



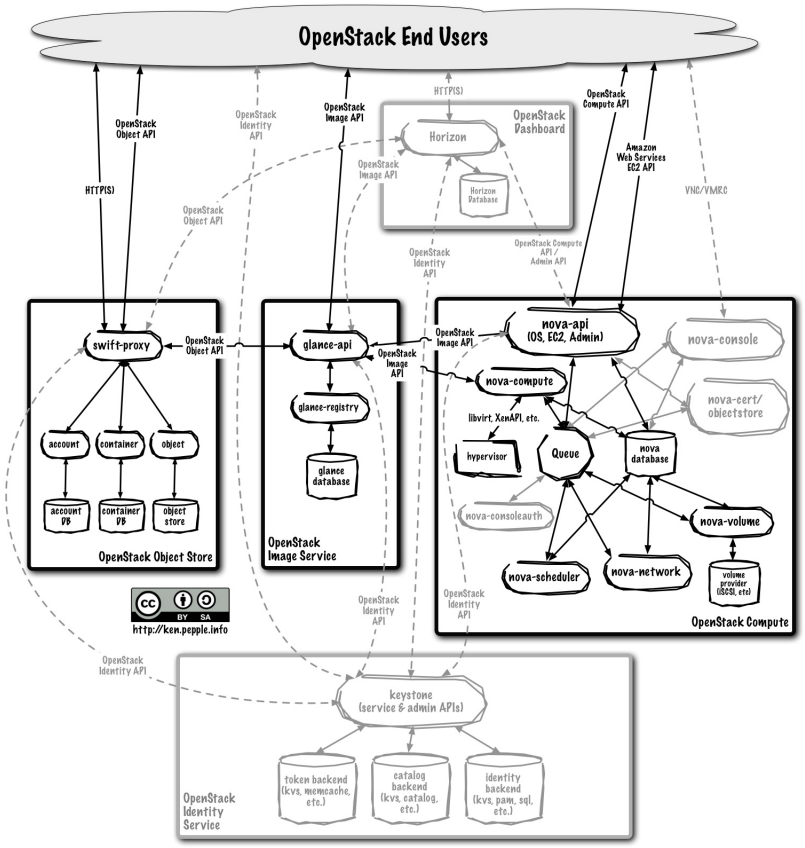
En la figura quedan claras las siguientes relaciones:

- *Horizon* proporciona un frontal gráfico basado en web para la gestión del resto de servicios de OpenStack
- Nova almacena y recupera imágenes de discos virtuales y sus datos asociados (metadatos) a través del servicio *Glance*.
- *Glance* almacena las imágenes en un directorio en disco, pero puede hacerlo a través del servicio *Swift*.
- El servicio *Keystone* es el encargado de la autenticación de todos los servicios.

Esta es una visión muy simplificada de toda la arquitectura, asumiendo además que utilicemos todos los servicios. Por otro lado, muestra únicamente el lado "operador" del cloud, la imagen no representa cómo los consumidores del cloud pueden realmente usarlo, por ejemplo, se puede hacer uso del servicio Swift de forma intensa y directa.

Arquitectura lógica

Como nos podemos imaginar, la arquitectura real del cloud, su arquitectura lógica, es mucho más complicada que la mostrada anteriormente. Como cualquier arquitectura orientada a servicios, cualquier diagrama que intente ilustrar todas las posibles combinaciones de comunicación de servicios, enseguida se vuelve muy confuso. El siguiente diagrama trata de mostrar el escenario más común, mostrando arquitectura perfectamente integrada de un cloud basado en OpenStack:



Este dibujo es perfectamente consistente con la arquitectura conceptual descrita anteriormente:

- Los usuarios finales del cloud pueden interactuar con él a través de la interfaz web (Horizon) o directamente con cada servicio a través de su API.
- Todos los servicios se autentican a través de un mismo servicio, el proporcionado por Keystone.
- Los servicios individuales interactúan con el resto a través de sus APIs públicas (excepto en los casos en los que se requieran comandos administrativos con privilegios).

En los siguientes apartados se describirán las arquitecturas para cada uno de los servicios.

Dashboard

Horizon es una aplicación web modular desarrollada con el framework de Python Django, cuyo objetivo principal es proporcionar una interfaz a los servicios de OpenStack al administrador del cloud y a los usuarios.

Horizon no proporciona toda la funcionalidad que podemos conseguir a través del intérprete de comandos, pero lo "poco" que hace lo hace correctamente.

Como cualquier aplicación web, la arquitectura de Horizon es bastante simple:

- Horizon normalmente se despliega a través del módulo de Apache mod_wsgi, el cual implementa la interfaz WSGI que permite al servidor Apache ejecutar aplicaciones Python.

  El código de Horizon está separado en dos módulos Python reutilizables, uno de ellos mantiene toda la lógica de la aplicación y es el encargado de interactuar con varias de las APIs de OpenStack, mientras que el otro es el encargado de la presentación, permitiendo fácilmente la adaptabilidad e integración con la apariencia del sitio web.

- Una base de datos. Horizon almacena muy pocos datos, ya que utiliza los datos del resto de servicios.

Desde el punto de vista de la red, este servicio debe ser accesible por los usuarios a través de la web (tráfico HTTP), de la misma forma que necesita poder acceder a las APIs públicas del resto de servicios. Si además se usa la funcionalidad de administración, necesita además conectividad a las APIs de administración de los endpoints (las cuales no son accesibles por los usuarios finales).

Compute

Nova no ha cambiado mucho desde las anteriores versiones, se han añadido ciertas mejoras en determinados servicios para la compatibilidad de EC2 y servicios de consola.

Nova depende de los siguientes demonios para su funcionamiento:

- `nova-api` es la encargada de aceptar y responder a las llamadas del usuario final a las APIs de `nova-compute` y `nova-volume`. El demonio `nova-api` soporta la API de OpenStack, la API EC2 de Amazon y la API especial de administración (para usuarios con privilegios que realicen tareas administrativas).

  Además, este demonio es el encargado de la coordinación de ciertas actividades (como la ejecución de una instancia) y la aplicación de ciertas políticas (como la comprobación de cuotas).

  En Essex, `nova-api` se ha modularizado, permitiendo únicamente la ejecución de determinadas APIs.

- El demonio `nova-compute` es el principal encargado de crear y acabar con las máquinas virtuales (instancias) utilizando para ello las APIs del hipervisor utilizado. `nova-compute` utiliza libvirt para KVM/QEMU, XenAPI para XenServer/XCP y VMwareAPI para Vmware.

  El proceso completo de creación/destrucción de instancias es bastante complejo, pero la base es muy simple: aceptar acciones de la cola de mensajes y ejecutar un conjunto de comandos del sistema asociados (como lanzar una instancia de KVM), todo mientras que se actualiza el estado en la base de datos.

- `nova-volume` gestiona la creación, conexión y desconexión de volúmenes persistentes a las instancias, de forma similar a como lo realizar el servicio EBS (Elastic Block Storage) de Amazon. Se pueden utilizar volúmenes de diferentes proveedores como iSCSI ó RADOS Block Device (RBD) de Ceph .

- El demonio `nova-network` es muy parecido a los demonios `nova-compute` y `nova-volume`. Acepta tareas de red desde la cola de mensajes y realiza ciertas que modifican

el estado de la red, como por ejemplo configurar una interfaz bridge ó cambiar las reglas de iptables.

- El demonio `nova-scheduler` es conceptualmente la pieza de código más simple de *Nova*. A partir de un mensaje de solicitud de creación de una instancia, determina qué nodo de OpenStack debe ejecutar dicha instancia de acuerdo a un algoritmo previamente seleccionado. La elección se realiza entre todos los nodos que ejecutan el demonio `nova-compute`.

- La cola de mensajes (queue) proporciona un hub centralizado para el intercambio de mensajes entre todos los demonios. Como cola de mensajes, se utiliza actualmente RabbitMQ, pero se puede utilizar cualquier otra cola de mensajes compatible con AMQP como por ejemplo Apache Qpid.

- Una base de datos SQL. El sistema gestor de BBDD será el encargado de almacenar toda la información del cloud así como el estado inicial y de ejecución. Esto incluye los tipos de instancia que están disponibles para el uso, las instancias creadas en un momento determinado, las redes disponibles, los proyectos existentes, etc. Teóricamente, *Nova* soporta cualquier base de datos soportada por SQL-Alchemy, pero las bases de datos realmente utilizadas actualmente son PostgreSQL, MySQL y sqlite3 (esta última solo para pruebas y desarrollo).

Note

SQL-Alchemy es un toolkit SQL y un sistema de mapeo objeto/relacional para el lenguaje de programación Python. Permite utilizar objetos desde Python al mismo tiempo que se aprovecha la velocidad, el rendimiento y la fiablidad de las bases de datos relacionales.

Durante las últimas dos revisiones, Nova ha visto aumentado sus servicios de consola. Los servicios de consola permiten a los usuarios finales acceder a sus máquinas virtuales a través de una consola de texto (caso de GNU/Linux) o consola gráfica (caso de máquinas virtuales Linux y Windows). Este acceso se realiza utilizando un proxy y se basa en los demonios `nova-console` y `nova-consoleauth`

Nova interactúa con el resto de servicios de la forma esperada: con Keystone para la autenticación, con Glance para la recuperación de imágenes y con Horizon para la interfaz web. La interacción con Glance es interesante, El proceso `nova-api` puede subir imágenes y consultar a Glance, mientras que `nova-compute` se descargará la imagen necesaria para lanzar una nueva instancia.

Object Storage

La arquitectura de *Swift* es distribuida tanto para prevenir cualquier punto simple de fallo como para posibilitar escalar horizontalmente. *Swift* incluye los siguientes componentes:

- Un servidor proxy. El proxy acepta solicitudes a través de la API OpenStack Object o directamente a través de HTTP. Una solicitud puede ser una subida de un fichero, modificación de los metadatos o la creación de un contenedor. También acepta solicitudes de descarga de ficheros o del listado de objetos del contenedor a través de un navegador web. Para mejorar el rendimiento, el servidor proxy puede, de forma optativa, utilizar una caché (normalmente memcache).

- Un servidor de cuentas de usuario. Este servidor gestiona las cuentas de usuario definidas con el servicio de almacenamiento de objetos.

- Servidores de objetos. Los servidores de objetos gestionan los objetos reales (como ficheros o contenedores) en los nodos de almacenamiento.

- Hay también un conjunto de procesos que se ejecutan de forma periódica que realizan ciertas tareas de limpieza sobre los datos. El más importante de estos procesos es el servicio de replicación, los cuales aseguran consistencia y disponibilidad en todo el cluster. Otros procesos periódicos incluyen auditores, actualizadores y reapers.

La autenticación se realiza normalmente a través de Keystone.

Image Storage

La arquitectura de *Glance* se ha mentenido relativamente estable desde la versión *Cactus* de OpenStack. El mayor cambio lo representa la incorporación de Keystone como sistema de autenticación y autorización, la cual se añadió en la versión *Diablo*. Glance está formado por cuatro componentes principales:

- El demonio `glance-api`. Encargado de aceptar peticiones a través de su API para el descubrimiento, recuperación y almacenamiento de imágenes.

- `glance-registry`. Encargado de almacenar, procesar y recuperar metainformación sobre las imágenes (tamaño, tipo, etc.).

- Una base de datos para almacenar dicha metainformación. De la misma forma que Nova, la base de datos es optativa, pero la decisión siempre gira en torno a MySQL ó PostgreSQL para entornos en producción.

- Un repositorio de almacenamiento para los ficheros de imágenes. Este almacenamiento es configurable y pueden utilizarse desde directorios locales al propio servicio Swift. Otras soluciones pasan por volúmenes iSCSI, directorios NFS ó CIFS, RADOS block device, Amazon S3 ó HTTP.

Existen también un conjunto de servicios para la gestión de la caché.

Glance representa un papel central en la arquitectura de OpenStack, ya que acepta peticiones para la gestión de imágenes tanto de los usuarios finales como de otros servicios como Nova.

Identity

Keystone permite la integración de los serviciós de OpenStack en un único punto en aspectos tan importantes como proporcionar servicios de autenticación, gestión de tokens y el mantenimiento de un catálogo y un repositorio de políticas de identidad.

Cada función de Keystone puede conectarse a un backend distinto que permite realizar esa misma función de diferentes formas utilizando un servicio distinto. De esta forma, Keystone puede integrarse fácilmente con diferentes almacenamiento como SQL, LDAP ó KVS (Key Value Stores).

Esto es muy útil en cuanto a la integración de los servicios de autenticación de OpenStakc con los servicios de autenticación existentes en un despliegue en concreto.

Nuevos componentes

Estos servicios enumerados son los que se encuentran en la versión de OpenStack utilizada en este proyecto (2012.1 (*Essex*)), pero el desarrollo de OpenStack continua de forma intensa. Para la siguiente revisión de OpenStack, con nombre en clave *Folsom*, se han añadido los siguientes nuevos servicios:

- Network. El servicio Network, con nombre *Quantum*. El objetivo principal de Quantum es proporcionar "conectividad de red como servicio" entre las interfaces de red gestionadas por otros servicios como Nova. Esto permitirá una gran flexibilidad a la hora de que los usuarios finales puedan crear sus propias redes e interconectar entre sí las instancias.

- Block Storage. De forma complementaria al almacenamiento de objetos que realiza swift, este componente de nombre *Cinder* es el encargado del almacenamiento de bloques, que se utilizan en las instancias de OpenStack, es equivalente al servicio de pago Elastic Block Storage (EBS) de Amazon.

Introducción a OpenStack Compute

OpenStack Compute proporciona una herramienta para orquestar un cloud, incluyendo la ejecución de instancias, administración de redes y control de acceso a usuarios y proyectos. El nombre del proyecto es Nova y proporciona el software que controla una plataforma de cloud computing de IaaS. Nova tiene un ámbito similar a Amazon EC2 y Rackspace Cloud Servers. OpenStack Compute no incluye ningún software de virtualización, en lugar de eso, define controladores que interactúan hipervisores que se ejecutan en otros equipos e interactúa a través de una API web.

Hipervisores

OpenStack Compute necesita al menos un hipervisor para funcionar y lo controla a través de una API. Hay varios hipervisores soportados, aunque los más utilizados son KVM y Xen-based hypervisors. Puede utilizarse http://wiki.openstack.org/HypervisorSupportMatrix para obtener una lista detallada de los hipervisores soportados.

Con OpenStack Compute, se pueden organizar clouds con más de un hipervisor a la vez. Los tipos de virtualización que se pueden utilizar con OpenStack Compute son:

- KVM - Kernel-based Virtual Machine
- LXC - Linux Containers (through libvirt)
- QEMU - Quick EMUlator
- UML - User Mode Linux
- VMWare ESX/ESXi 4.1 update 1
- Xen - Xen, Citrix XenServer and Xen Cloud Platform (XCP)

Usuarios y proyectos (Tenants)

OpenStack Compute está diseñado para que lo utilcen usuarios muy diversos, a los que se les pueden asignar diferentes roles. Mediante el rol de un usuario se puede controlar las acciones que esté puede realizar. En la configuración por defecto, la mayoría de las acciones no llevan asociadas ningún rol, pero es responsabilidad del administrador del cloud, configurar apropiadamente estas reglas a través del fichero `policy.json`. Por ejemplo, una regla puede limitar al rol de administrador la solicitud de una dirección IP pública o el acceso de un usuario a determinadas imágenes.

Note

Las versiones anteriores de OpenStack utilizaban el término "project" en lugar de "tenant", por lo que algunas herramientas de línea de comandos todavía utilizan `--project_id` en lugar de tenant.

Imágenes e instancias

Las imágenes son imágenes de discos que son plantillas para las máquinas virtuales que se van a crear. El servicios que proporciona las imágenes, Glance, es el responsable de almacenar y gestionar las imágenes en OpenStack.

Las instancias son las máquinas virtuales que se ejecutan en los nodos de computación. El servicio de computación, Nova, gestiona estas instancias. Se pueden lanzar cualquier número de instancias a partir de una determinada imagen. Cada instancia se ejecuta de una copia de una imagen base, por lo que las modificaciones que se realicen en la instancia no alteran la imagen en la que se basa. Mediante el uso de instantáneas (*snapshots*) de las instancias, se pueden crear nuevas imágenes que sí guardan todas las modificaciones realizadas hasta ese momento en la instancia.

Cuando se lanza una instancia, se debe seleccionar un conjunto de recursos virtuales, conocido como sabor (*flavor*). Un sabor define para una instancia el número de CPUs virtuales, la RAM, si dispone o no de discos efímeros, etc. OpenStack preinstala una serie de sabores, que el administrador puede modificar.

Recursos adicionales como volúmenes persistentes o direcciones IP se pueden añadir o quitar a instancias que se estén ejecutando.

Lanzar una instancia

Para lanzar una instancia hay que elegir una imagen, un sabor y opcionalmente otros atributos. OpenStack copia la imagen base al disco que utilizará la instancia como primer disco (vda), cuanto más pequeña sea la imagen, más rápido será el lanzamiento. Dependiendo del sabor, también se crea un nuevo disco vacío (vdb). El nodo de computación conecta en su caso mediante iSCSI con nova-volume y mapea el volumen escogido como vdc.

Obviamente hay posibles variaciones de este típico lanzamiento, sobre todo con respecto al almacenamiento. Por ejemplo, es posible que los discos vda y vdb del ejemplo anterior no estén alojados localmente sino en un recurso en red.

Una vez que decidimos terminar la instancia, todos los recursos utilizados por la instancia son liberados (RAM, almacenamiento, etc.), salvo los volúmenes persistentes que permanecen almacenados y es posible asignarlos a otra instancia posteriormente.

Almacenamiento de bloques y OpenStack Compute

OpenStack proporciona dos tipos de almacenamiento de bloques: almacenamiento efímero y volúmenes persistentes. El almacenamiento efímero existe sólo mientras se ejecuta la instancia, se mantendrá cuando se reinicie la instancia, pero se borrará en el momento que se borre la instancia para la que se creó. Todas las instancias tienen almacenamiento efímero y es posible, dependiendo del sabor, que tengan más de un disco efímero a su disposición. Los volúmenes persistentes dispositivos de bloques independientes de la instancia. Los volúmenes se pueden asociar a una determinada instancia, pero posteriormente se pueden desasociar y asociar a cualquier otra instancia manteniendo los datos, como si fuera una unidad USB.

Introducción a OpenStack Keystone

Este capítulo describe la instalación y configuración del módulo de OpenStack, Keystone. Este módulo es el encargado del sistema de autentificación y autorización de los distintos módulos que conforman el sistema.

Introducción al módulo keystone

Keystone, es el componente de OpenStack encargado de la autentificación y la autorización de los distintos componentes desde la versión Essex y tiene dos funciones principales:

- Gestión de usuarios: Keystone es el encargado de mantener un registro de usuarios y los permisos que tienen cada uno de ellos.
- Registro los servicios ofrecidos: Keystone ofrece un catálogo de los servicios ofrecidos, así como la forma de acceder a sus APIs.

Los conceptos fundamentales de la *gestión de usuarios* son:

- Usuario: Podemos guardar su nombre, correo electrónico y contraseña.
- Proyecto (*tenant* en la jerga de OpenStack): En un proyecto podemos ejecutar un conjunto de instancias con características en común, por ejemplo pueden estar todas las instancias en el misma red, pueden utilizar una serie de imágenes de sistemas o tener limitado el uso de recursos del cloud.
- Rol: Nos indica qué operaciones puede realizar cada usuario. A un usuario se le pueden asignar diferentes roles en cada proyecto.

Los conceptos fundamentales del *registro de servicio* son:

- Servicio: Corresponde a un componente de OpenStack que puede utilizar el módulo de autentificación.

- Endpoints: Representa las URL que nos permiten acceder a las API de cada uno de los servicios o componentes de OpenStack

¿Qué es el ADMIN_TOKEN y para qué se utiliza?

Keystone introduce en OpenStack un sistema de autenticación basado en tokens, de manera que todos los elementos del cloud (usuarios y servicios principalmente), no se autentican directamente unos a otros, sino que lo hace con un actor intermedio mediante tokens, este actor intermedio encargado de verificar la autenticidad de cada uno de los elementos es Keystone. Un proceso típico de autenticación en OpenStack puede verse en la siguiente imagen, en la que se muestran los pasos que se dan desde que el usuario se acredita frente a Keystone hasta que lanza una instancia.

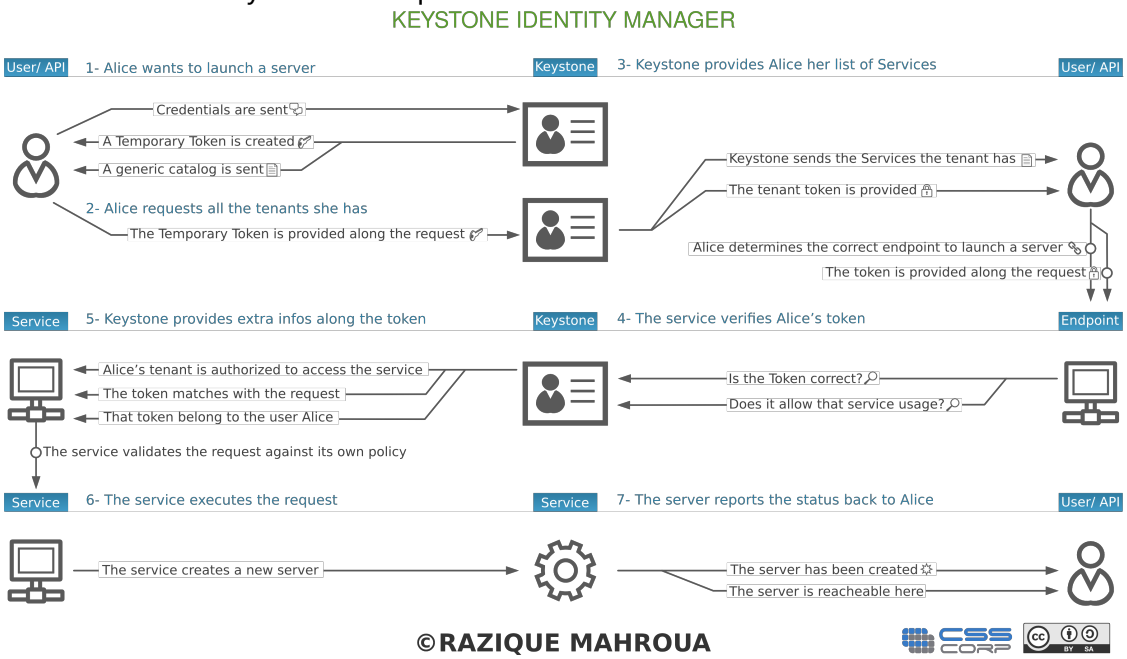


Diagrama que describe el funcionamiento típico de keystone.

Con Keystone recién instalado no tenemos ningún usuario con privilegios de administración en la base de datos de keystone, por lo no podríamos hacer ninguna modificación. El mecanismo que proporciona keystone para solventar esta situación es definir en el fichero de configuración un token con privilegios de administración, que recibe el nombre de ADMIN_TOKEN, que puede definirse directamente en la instalación o cuando queramos en el fichero `/etc/keystone/keystone.conf`, el ADMIN_TOKEN puede tener cualquier valor, pero hay que custodiarlo adecuadamente mientras esté activo, ya que otorga privilegios de administración sobre Keystone a quien lo conozca. Posteriormente, una vez definidos los usuarios en keystone y los demás elementos, podemos borrar este campo del fichero de configuración de keystone y no volver a utilizarlo.

Uso del ADMIN_TOKEN

De forma general en Keystone, un usuario deberá tener algo con que demostrar su autenticidad (una contraseña o un token) y la URL de un API con la que desea interaccionar, en este caso vamos a definir con dos variables de entorno el token con privilegios de administrador y la URL del servicio de administración de keystone:

```
export SERVICE_ENDPOINT=http://127.0.0.1:35357/v2.0/
```

```
export SERVICE_TOKEN=12345678
```

donde 12345678 representa el valos que hayamos elegido para nuestro ADMIN_TOKEN y tendrá el mismo valor que el parámetro ADMIN_TOKEN que tenemos guardado en el fichero de configuración. Una vez realizado esto el usuario de esa sesión actúa como administrador de keystone, por lo que los pasos siguientes serán crear usuarios, darles roles, definir los servicios, etc.

Introducción a OpenStack Glance

¿De dónde?

Introducción a OpenStack Swift

¿De dónde?

2. Instalación de OpenStack en Debian GNU/Linux Wheezy

Para la realización de esta sección se han tomado como referencias principales las siguientes referencias:

- OpenStack on Debian GNU/Linux testing de la wiki de Debian, que es la referencia "oficial" para la instalación en Debian Wheezy y que está mantenida por el grupo de desarrolladores debian encargados del empaquetamiento del proyecto OpenStack y cuya actividad puede seguirse a través de alioth.
- OpenStack Install and Deploy Manual del proyecto OpenStack, pero pensado para la instalación en Ubuntu 12.04.

El documento de la wiki de Debian es correcto, pero describe muchos aspectos de forma demasiado breve por lo que no es adecuado para personas que comienzan en OpenStack, mientras que la segunda referencia es específica para la instalación en Ubuntu, por lo que hay algunas diferencias en la instalación en Debian.

Pasos previos

Antes de comenzar con la instalación de paquetes en sí, hay que plantear claramente la estructura de red y los componentes que se instalarán en cada equipo. La descripción física de los equipos que componen el cloud ya se realizó en el documento que describe la infraestructura para el cloud de este mismo proyecto, por lo que nos centraremos ahora en la instalación en sí.

Nombres de los equipos

Se han elegido los siguientes nombres para los equipos en los que se va a realizar la instalación del Cloud:

- jupiter: para el nodo controlador, que 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, pero en el que no se ejecutarán 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.

  En jupiter instalaremos los siguientes componentes:

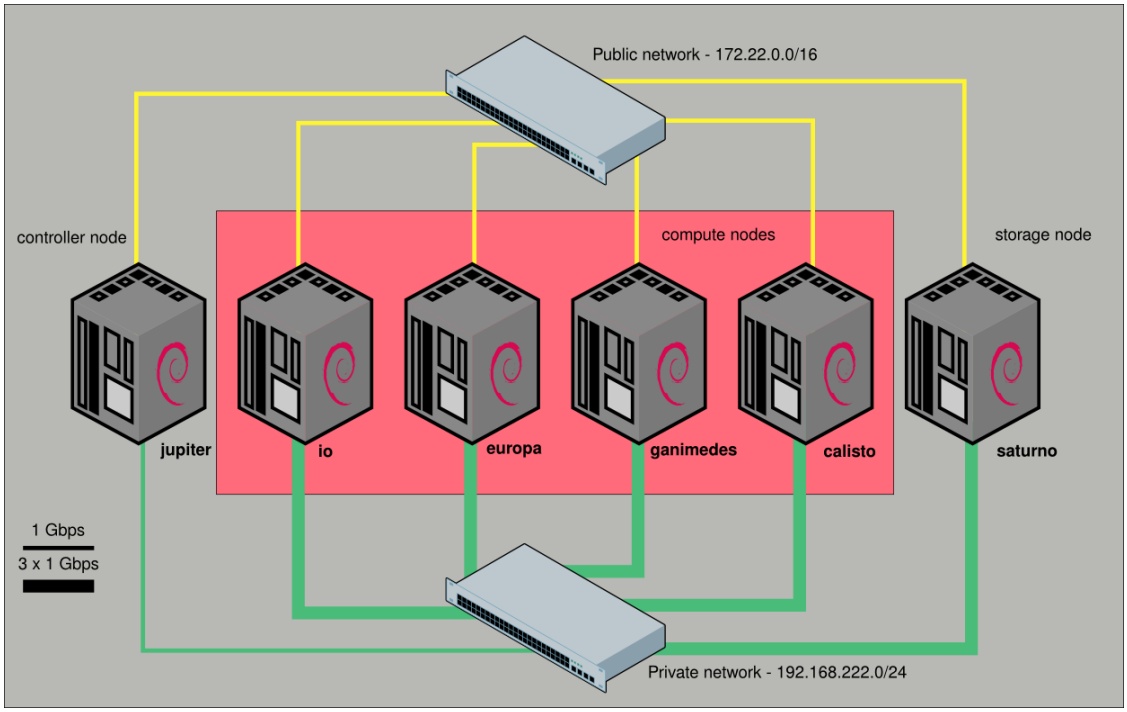
  - nova-api
  - nova-cert
  - nova-console
  - nova-scheduler

    - nova-volume
    - keystone
    - glance
    - horizon
- io, europa, ganimedes y calisto (son las 4 lunas principales de júpiter): para los 4 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 en ellos y estarán esperando las órdenes de jupiter, en concreto se instalarán los componentes:
    - nova-api
    - nova-cert
    - nova-compute
    - nova-network
- saturno: para el nodo de almacenamiento, ya que es tan importante como júpiter y a la vez independiente. En este equipo se instalará software independiente del proyecto OpenStack para el manejo de volúmenes persistentes a través del protocolo iSCSI.
- venus: 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.

Esquema de red

Cada uno de estos equipos está conectado a dos redes (salvo venus), lo que en terminología de OpenStack se conoce como una red *pública* y una red *privada*, términos a los que no se debe dar el sentido habitual que tienen en redes IPv4, ya que en OpenStack la red pública se refiere a la que interaccionará con los clientes y la privada la que se utiliza para la intercomunicación de los componentes del cloud y en nuestro caso muy destacadamente, de la transferencia de imágenes a los nodos en los que se deben ejecutar las instancias. La red privada es una red aislada que no está conectada con otras redes, mientras que la red pública sí lo está, bien a una red local bien a Internet. Las transferencias de datos más importante se realizan a través de la red privada, por lo que se ha optado por conectar cada equipo (salvo jupiter) mediante tres interfaces de red Gigabit Ethernet, para que configurándolas adecuadamente en modo *link aggregation* de acuerdo al estándar 802.3ad, puedan aumentar las tasas de transferencia en la red privada.

En la siguiente imagen, se puede ver de forma esquemática los equipos y las redes a las que están conectados:



Esquema de red del Cloud, en la que aparecen los 6 equipos conectados a las redes privada y pública del cloud.

Direcciones IP de los equipos

En la siguiente tabla aparecen las direcciones IPv4 privadas y públicas elegidas para los nodos del cloud.

Table 2.1. Direcciones IP de los equipos del cloud

	IP pública	IP privada
jupiter	172.22.222.1	192.168.222.1
saturno	172.22.222.2	192.168.222.2
venus		192.168.222.10
io	172.22.222.11	192.168.222.11
europa	172.22.222.12	192.168.222.12
ganimedes	172.22.222.13	192.168.222.13
calisto	172.22.222.14	192.168.222.14

Instalación y configuración inicial de MySQL

Todos los componentes de OpenStack (incluso glance y keystone que por defecto utilizan sqlite) guardarán sus datos en bases de datos MySQL. El servidor de bases de datos se instala en jupiter, el nodo controlador:

```
root@jupiter:~# aptitude install mysql-server
```

Introducimos la contraseña del usuario root de MySQL cuando se requiera y esperamos a que finalice la instalación. Una vez concluida, abrimos el fichero `/etc/mysql/my.cnf` y realizamos la siguiente modificación:

```
bind_address = 192.168.222.0/24
```

para que la base de datos sea accesible sólo a través de la red privada.

Entramos en MySQL con el usuario root que se crea durante la instalación y creamos diferentes bases de datos para los componentes keystone, glance y nova y un usuario para OpenStack, que tendrá todos los permisos sobre las bases de datos:

```
mysql> CREATE DATABASE keystone;
mysql> CREATE DATABASE glance;
mysql> CREATE DATABASE nova;
mysql> CREATE USER "usuario_admin_openstack" IDENTIFIED BY 'password';
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'openstackadmin'@'localhost' \
-> IDENTIFIED BY 'password';
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'opentackadmin"'@'localhost';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'openstackadmin"'@'localhost';
```

Instalación de otros paquetes

Antes de instalar propiamente los paquetes relacionados con los diferentes componentes de Openstack, es recomendable instalar los siguientes paquetes:

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

rabbitmq-server se encarga de la gestión de mensajes entre los diferentes componentes de OpenStack (es un paquete obligatorio que si no instalamos ahora se instalará por dependencias) y memcached se encarga de cachear en memoria peticiones a bases de datos o a APIs

Sincronización de la hora de los equipos con ntp

Es muy importante que todos los equipos del cloud tengan sus relojes sincronizados, por lo que lo más sencillo es configurar un servidor local como servidor ntp, que se sincronice con los servidores de hora públicos que hay disponibles en Internet y ofrezca la hora a todos los equipos del cloud. Realmente no es fundamental que la hora de los equipos del cloud sea muy exacta, pero sí que estén siempre sincronizados.

Ya que en la red local existía previamente un servidor ntp, simplemente hay que instalar el paquete ntp en *todos los nodos*, comentar todas las líneas que empiecen por server en el fichero `/etc/ntp.conf` y añadir una línea del tipo:

```
server ntp.your-provider.example
```

Podemos comprobar el funcionamiento correcto tras reiniciar el servicio ntp y realizar una consulta:

```
root@jupiter:~# ntpq -np
```

De forma paulatina, el reloj del equipo se irá sincronizanco con el del servidor de hora de la red.

Instalación manual de python-prettytable

Uno de los paquetes que se instalan por dependencias (python-prettytable), tiene abierto el bug 673790 en la versión 0.6 que actualmente está en Debian Wheezy, por lo que es necesario instalar la versión 0.5 de forma manual y ponerla en estado *hold* para que no se actualice, mientras persista este bug:

```
root@jupiter:~# wget http://ftp.es.debian.org/debian/pool/main/p/prettytable/
python-prettytable_0.5-1_all.deb
 root@jupiter:~# apt-get install python-prettytable=0.5-1
 root@jupiter:~# echo python-prettytable hold |  dpkg --set-selections
```

Este paso hay que realizarlo en jupiter y en los cuatro nodos de computación.

Keystone

Instalación de keystone

Vamos a instalar Keystone utilizando el paquete del repositorio oficial de Debian Wheezy. Para ello ejecutamos la siguiente instrucción como administrador del sistema:

```
root@jupiter:~# aptitude install keystone
```

Durante la instalación nos pide el ADMIN_TOKEN, que nos servirá durante la configuración inicial y que se guarda en la directiva admin_token del fichero de configuración `/etc/keystone/keystone.conf`.

Configuración de keystone

El fichero de configuración de keystone lo encontramos en `/etc/keystone/keystone.conf`. La primera configuración que realizamos será la conexión con la base de datos, ya que como dijimos anteriormente, vamos a utilizar bases de datos en MySQL para cada uno de los componentes de OpenStack:

```
connection = mysql://"usuario":"password"@127.0.0.1:3306/keystone
```

Reiniciamos keystone y ejecutamos el comando que sincroniza la BBDD de keystone, es decir, crea la tablas necesarias para el funcionamiento de Keystone:

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

Esto simplemente crea las tablas necesarias en la base de datos que hemos llamado keystone, pero no añade ningún registro, este procedimiento todavía no está automatizado y lo haremos en las siguientes secciones.

Creación de proyectos, usuarios y roles

Creación de proyectos (tenants)

Comenzamos creando los dos proyectos (tenants) inciales con los que vamos a trabajar: admin y service. Para ello ejecutamos las siguientes intrucciones:

```
root@jupiter:~# keystone tenant-create --name admin
```

```
root@jupiter:~# keystone tenant-create --name service
```

Los valores de los id resultantes de cada instrucción, los asociamos a las variables de entorno ADMIN_TENANT y SERVICE_TENANT (podría hacerse en un solo paso utilizando la función get_id que recomiendan en la wiki de Debian), para que sea más cómodo utilizarlo luego:

```
root@jupiter:~# export ADMIN_TENANT="id del tenant admin"
```

```
root@jupiter:~# export SERVICE_TENANT="id del tenant service"
```

Creación de usuarios

A diferencia de la documentación de OpenStack, vamos a crear dos usuarios (uno que tendrá el rol de admin sobre el tenant admin y otro que tendrá el rol de admin sobre el tenant service que utilizan el resto de componentes de OpenStack):

```
root@jupiter:~# keystone user-create --name "nombre gran jefe" --pass
 "contraseña" --email "correo-e"
```

```
root@jupiter:~# keystone user-create --name "nombre jefe" --pass "contraseña"
 --email "correo-e"
```

De nuevo asociamos los id resultantes de ambas instrucciones a variables de entorno que utilizaremos después (ADMIN_USER y SERVICE_USER):

```
root@jupiter:~# export ADMIN_USER="id del usuario gran jefe"
```

```
root@jupiter:~# export SERVICE_USER="id del usuario jefe"
```

Creación de roles

Creamos los roles admin y Member que tendrán diferentes privilegios. De momento sólo utilizaremos el rol admin, aunque el rol más habitual para trabajar en el cloud será el de Member:

```
root@jupiter:~# keystone role-create --name admin
```

```
root@jupiter:~# keystone role-create --name
 Member
```

Listamos los roles y asignamos el rol de admin a la variable ADMIN_ROLE:

```
root@jupiter:~# ADMIN_ROLE=$(keystone role-list|awk '/ admin / { print $2 }')
```

Asignación de los roles

Asignamos el rol admin en el tenant admin al usuario que queremos que sea el administrador:

```
root@jupiter:~# keystone user-role-add --user $ADMIN_USER --role $ADMIN_ROLE
 --tenant_id $ADMIN_TENANT
```

Asignamos el rol admin en el tenant service al otro usuario:

```
root@jupiter:~# keystone user-role-add --user $SERVICE_USER --role $ADMIN_ROLE
 --tenant_id $SERVICE_TENANT
```

Configuración de las políticas de autorización

Es posible ajustar los privilegios de cada rol, realizando ajustes en el fichero `/etc/keystone/policy.json`, tal como se explica en basics concepts de la documentación oficial de OpenStack.

Configuración de los servicios

En Debian Wheezy inicialmente, los "endpoints" se definen de forma estática en el fichero `/etc/keystone/default_catalog.templates` y los servicios en ram, mientras que en la documentación oficial de OpenStack, se explican los pasos para incluirlos en la base de datos MySQL. Es lo que vamos a hacer nosotros, para lo que editamos el fichero `/etc/keystone/keystone.conf`:

```
[catalog]
#driver = keystone.catalog.backends.templated.TemplatedCatalog
#template_file = /etc/keystone/default_catalog.templates
driver = keystone.catalog.backends.sql.Catalog
```

Creación de servicios

Creamos los servicios keystone, nova, volume y grance (de momento obviamos swift y ec2):

```
root@jupiter:~# keystone service-create --name keystone --type identity --
description 'OpenStack Identity Service'
 root@jupiter:~# keystone service-create --name nova --type compute --
description 'OpenStack Compute Service'
 root@jupiter:~# keystone service-create --name volume --type volume --
description 'OpenStack Volume Service'
 root@jupiter:~# keystone service-create --name glance --type image --
description 'OpenStack Image Service'
```

Creación de los "endpoints"

Los endpoints son las urls para el manejo de las diferentes APIS. Para cada componente de OpenStack se definen tres URLs (la pública, la de administración y la interna), en algunos casos el puerto es el mismo, pero en otros no. Es necesario revisar muy bien este paso porque es bastante propenso a errores. En nuestro caso, utilizaremos la dirección IP pública de jupiter para la url pública y la IP privada para la de administración e interna, además definimos una sola región con el nombre iesgn (OpenStack permite crear cloud de gran tamaño en los que pueden definirse regiones, cada una de ellas con parámetros propios, pero no es nuestro caso):

```
root@jupiter:~# keystone endpoint-create --region iesgn --service_id "id de
 keystone" --publicurl http://172.22.222.1:5000/v2.0 --adminurl http://192.
168.222.1:35357/v2.0 --internalurl http://192.168.222.1:5000/v2.0
 root@jupiter:~# keystone endpoint-create --region iesgn --service_id "id de
 nova"  --publicurl 'http://172.22.222.1:8774/v2/$(tenant_id)s' --adminurl
 'http://192.168.222.1:8774/v2/$(tenant_id)s' --internalurl 'http://192.168.
222.1:8774/v2/$(tenant_id)s'
 root@jupiter:~# keystone endpoint-create --region iesgn --service_id "id
 de nova-volume" --publicurl 'http://172.22.222.1:8776/v1/$(tenant_id)s' --
adminurl 'http://192.168.222.1:8776/v1/$(tenant_id)s' --internalurl 'http://
192.168.222.1:8776/v1/$(tenant_id)s'
```

```
root@jupiter:~# keystone endpoint-create --region iesgn --service_id "id de
glance" --publicurl 'http://172.22.222.1:9292/v1' --adminurl 'http://192.168.
222.1:9292/v1' --internalurl 'http://192.168.222.1:9292/v1'
```

Método de autentificación

Un vez que tenemos añadidos nuestros usuarios, con sus respectivos roles en los distintos proyectos, la forma normal de acceder es autentificándose con algunos de estos usuarios.

Para ello creamos dos ficheros de configuración de las variables de entorno de los dos usuarios creados, lo llamamos por ejemplo /root/.granjefe:

```
#!/bin/bash
export OS_AUTH_URL=http://172.22.222.1:5000/v2.0
export OS_TENANT_NAME=admin
export OS_USERNAME="usuario gran jefe"
export OS_VERSION=1.1

# With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password: "
read -s OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT
```

Y para el otro usuario creamos /root/.jefe:

```
export OS_USERNAME="usuario jefe"
export OS_PASSWORD="password"
export OS_TENANT_NAME=service
export OS_AUTH_URL=http://192.168.222.1:5000/v2.0/
export OS_VERSION=1.1
```

Para que se Obviamente protegemos estos ficheros:

```
root@jupiter:~# chmod 600 /root/.granjefe
root@jupiter:~# chmod 600 /root/.jefe
```

Utilización de la API

Como hemos visto a lo largo de este manual, podemos utilizar el cliente keystone para gestionar los usuarios, roles, proyectos, servicios y endpoints. En concreto, hemos visto las instrucciones que nos permiten crear nuevos elementos.

Otros comandos interesantes nos permiten listar los objetos que hemos creado:

```
root@jupiter:~# keystone role-list
root@jupiter:~# keystone user-list
root@jupiter:~# keystone tenant-list
root@jupiter:~# keystone service-list
root@jupiter:~# keystone endpoint-list
```

Otro ejemplo de comando que podemos usar a menudo, es el que nos permite cambiar la contraseña de un usuario:

```
root@jupiter:~# keystone user-password-update uuid --pass
      nueva_contraseña
```

Para encontrar una descripción detallada de todos los comandos que podemos usar con el cliente keystone, podemos visitar el siguiente enlace: http://docs.openstack.org/essex/openstack-compute/admin/content/adding-users-tenants-and-roles-with-python-keystoneclient.html

Glance

Este capítulo describe la instalación y configuración del módulo de OpenStack, Glance. Este módulo es el encargado de la gestión y registro de las imágenes que posteriormente se van a poder instanciar.

Introducción al módulo glance

El componente Glance de OpenStack es el responsable de la localización, registro y gestión de las imágenes de máquinas virtuales que vamos a poder instanciar.

Las imágenes gestionadas por Glance se pueden almacenar en gran variedad de ubicaciones desde un simple sistema de ficheros hasta un sistema de almacenamiento de objetos como puede ser el proyecto OpenStack Swift.

El componente Glance posee una API REST que nos permite hacer consultas sobre los metadatos de las imágenes, así como la recuperación de la imagen real.

Instalación de Glance

Antes de realizar la instalación de Glance, vamos a repasar la configuración de Keystone para que podemos usarlo para realizar la autentificación:

- El usuario que va a administrar Glance será el usuario jefe, al que se le asignó el rol "admin" en el tenant "service".
- El servicio Glance ya fue creado en Keystone.
- Los endpoints de Glance también fueron definidos durante la instalación de Keystone.

Por lo tanto ya tenemos todo preparado para la instalación de Glance:

```
root@jupiter:~# aptitude install glance
```

Seleccionamos Keystone como servicio de autenticación, introducimos la dirección del servicio de Keystone (esto se puede cambiar más adelante, de momento dejaremos localhost), definimos el admin_token (Glance) y esperamos a que finalice la instalación.

Configuración de Glance

Una vez terminada la instalación de Glance realizamos los siguientes cambios en los ficheros de configuración:

etc/glance/glance-api-paste.ini

```
admin_tenant_name = service
admin_user = "usuario jefe"
admin_password = "password"
```

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

```
admin_tenant_name = service
admin_user = "usuario jefe"
admin_password = "password"
```

/etc/glance/glance-registry.conf

```
sql_connection =
mysql://"usuario_admin_openstack":"password"@127.0.0.1:3306/"base de datos de
glance"
[paste_deploy]
flavor = keystone
```

/etc/glance/glance-api.conf

```
[paste_deploy]
flavor = keystone
```

A continuación creamos el modelo de datos de glance e reiniciamos los servicios, para ello:

```
root@jupiter:~#glance-manage version_control 0
root@jupiter:~#glance-manage db_sync
root@jupiter:~#service glance-api restart
root@jupiter:~#service glance-registry restart
```

Método de autentificación y prueba de funcionamiento

Para que el usuario jefe administre el servicio Glance vamos a utilizar las variables que habíamos definido en el fichero /root/.jefe:

```
export OS_USERNAME="usuario jefe"
export OS_PASSWORD="password"
export OS_TENANT_NAME=service
export OS_AUTH_URL=http://192.168.222.1:5000/v2.0/
export OS_VERSION=1.1
```

Para finalizar podemos ejecutar la siguiente instrucción:

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

Nova en el nodo controlador

Este capítulo describe la instalación y configuración del módulo nova de OpenStack. Este módulo es el encargado de gestionar las instancias del cloud y por tanto es el elemento central para un cloud de infraestructura como el nuestro.

Introducción al módulo nova

Se explica posteriormente.

Instalación

Instalamos desde el repositorio de Debian:

```
#aptitude install nova-api
nova-scheduler nova-cert nova-console
```

Durante la configuración de los paquetes se nos pregunta si queremos configurar la base de datos con **dbconfig-common**, a lo que respondemos que no, posteriormente configuraremos la base de datos directamente sobre los ficheros de configuración de nova.

Además de los paquetes del repositorio Debian anteriormente indicado, debemos instalar los siguientes paquetes, cuyas versiones actuales que se encuentran en el repositorio testing de Debian no funcionan correctamente. Además se debe activar la propiedad "hold" para que estos paquetes no se actualicen en futuras actualizaciones:

```
#dpkg -i novnc_2012.1~e3+dfsg-1_amd64.deb dpkg -i python-
novnc_2012.1~e3+dfsg-1_all.deb echo novnc hold | dpkg --set-
selections echo python-novnc hold | dpkg --set-selections
```

Configuración

El fichero de configuración lo encontramos en el fichero `/etc/nova/nova.conf` cuyo contenido inicial es el siguiente:

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
connection_type=libvirt
root_helper=sudo nova-rootwrap
auth_strategy=keystone
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
iscsi_helper=tgtadm
sql_connection=mysql://usuario:pass@127.0.0.1/nova
```

Veamos las modificaciones necesarias en la configuración de nova para que funcione nuestro sistema adecuadamente:

`/etc/nova/nova.conf`

```
 [DEFAULT]

# LOGS/STATE
logdir=/var/log/nova
verbose=true
state_path=/var/lib/nova
lock_path=/var/lock/nova

# AUTHENTICATION
auth_strategy=keystone

# SCHEDULER
scheduler_driver=nova.scheduler.simple.SimpleScheduler

# DATABASE
sql_connection=mysql://"user_bd":"contraseña"@192.168.222.1/nova

# COMPUTE
libvirt_type=kvm
connection_type=libvirt
instance_name_template=instance-%08x
api_paste_config=/etc/nova/api-paste.ini

# RABBITMQ
rabbit_host=192.168.222.1

# GLANCE
glance_api_servers=192.168.222.1:9292
image_service=nova.image.glance.GlanceImageService

# NETWORK
network_manager=nova.network.manager.VlanManager
vlan_interface=eth1
public_interface=eth0
my_ip=192.168.222.1
routing_source_ip=172.22.222.1
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
multi_host=true

# NOVNC CONSOLE
vnc_enabled=true
novncproxy_base_url=http://172.22.222.1:6080/vnc_auto.html
vncserver_proxyclient_address=127.0.0.1
vncserver_listen=127.0.0.1

# MISC

root_helper=sudo nova-rootwrap

# QUOTAS
quota_cores=10
quota_gigabytes=1000
quota_injected_file_content_bytes=10240
quota_injected_file_path_bytes=255
quota_injected_files=5
quota_instances=10
```

```
quota_metadata_items=128
quota_ram=65000
quota_security_group_rules=10
quota_security_groups=5
quota_volumes=5


# APIS

# VOLUMEN
```

Veamos detalladamente algunos de los parámetros que hemos indicado en el fichero de configuración:

- En la sección LOGS/STATE configuramos algunos directorios de trabajo: donde guardamos los logs del sistema (logdir), el directorio base donde podemos encontrado el contenido del cloud (state_path) y el directorio donde podemos encontrar los ficheros de bloqueo (lock_path). Los logs que generan cada uno de los distintos subcomponentes de nova se guardan en ficheros separados dentro del directorio donde hemos indicado en la configuración, además el nivel de explicación de los logs se indica con el parámetro verbose.
- En la sección AUTHENTICATION con el parámetro auth_strategy indicamos el módulo que vamos a usar para el servicio de autentificación y autorización, en nuestro caso vamos a usar Keystone.
- En la sección SCHEDULER con el parámetro scheduler_driver indicamos el mecanismo de planificación que vamos a seguir para instanciar las máquinas virtuales en nuestro caso vamos a utilizar el simple.
- En la sección DATABASE con el parámetro sql_connection configuramos la cadena de conexión para acceder a la base de datos nova.
- En la sección COMPUTE configuramos el sistema de virtualización que vamos a utilizar con el parámetro libvirt_type, en nuestro caso kvm, vamos a utilizar libvirt (parámetro connection_type), la plantilla que vamos a usar para crear el nombre de las instancias (parámetro instance_name_template) y por último indicamos donde se encuentra el fichero de configuración api-paste.ini (api_paste_config).
- En la sección RABBITMQ indicamos, en el parámetro rabbit_host donde se encuentra el servidor Rabbit responsable de la mensajería interna de los distintos componentes.
- En la sección GLANCE se indica que aplicación va a gestionar las imágenes, en nuestro caso será Glance y el parámetro donde se indica es image_service, además se indica donde se encuentra el servicio glance (glance_api_servers).
- En la sección NOVNC CONSOLE se configura el acceso por la consola vnc, para ello indicamos la URL donde se encuentra el servicio novnc (novncproxy_base_url) y la dirreción del cliente vnc (vncserver_proxyclient_address y vncserver_listen).
- Con el parámetro root_helper indicamos un comando que internamente va a usar el sistema para ejecutar ciertas instrucciones con privilegio de superusuario. Por defecto es **sudo nova-rootwrap**

- En la sección QUOTAS podemos configurar las cuotas generales del sistema, aunque posteriormente se podrán sobreescribir para cada uno de los proyectos.Veamos que significan algunos de estos parámetros:

1. quota_gigabytes: Tamaño máximo del disco por proyecto.
2. quota_instances: Número máximo de instancias por proyecto.
3. quota_ram: Memoria RAM máxima utilizable por proyecto.
4. quota_security_group_rules: Número máximo de reglas por grupo de seguridad.
5. quota_security_group: Número máximo de grupos de seguridad por proyecto.
6. quota_volumes: Número máximo de volúmenes asociables a un proyecto.

- En la sección NETWORK se indica la configuración de red, que será explicada con detalle posteriormente. Veamos los parámetros que se deben indicar: el tipo de configuración de red (network_manager) en nuestro caso VLAN, la interfaz de red por la que se va a gestionar la vlna (vlan_interface) en nuestro caso es la privada eth1, la interfaz de red pública (public_interface), la dirección ip privada del nodo (my_ip), la dirección ip pública del nodo (routing_source_ip), utilización de varios nodos (multi_host) e información sobre el bridge (dhcpbridge_flagfile y dhcpbridge).

- Queda por configurar el uso de las APIS y el componente nova-volumen.

Veamos otro fichero de configuración.

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

En este fichero se añaden las mismas líneas que pusimos en la configuración de glance, donde se indica el usuario, la contraseña y el tenant con el que nos vamos a autentificar.

```
admin_tenant_name = service
admin_user = "usuario jefe"
admin_password = "password de usuario jefe"
```

Una vez definida la configuración es momento de reiniciar los servicios:

```
service nova-api restart
service nova-cert restart
service nova-console restart
service nova-consoleauth restart
service nova-scheduler restart
```

Para finalizar el proceso de configuración y después de habernos autentificado, debemos crear la tablas necesarias en la base de datos, para ello ejecutamos el siguiente comando:

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

Para ver que los servicios están activos podemos ejecutar el comando siguiente:

```
jupiter:~#nova-manage service list
```

Que nos debe ofrecer una salida parecida a la siguiente:

```
Binary           Host                                 Zone             Status
    State Updated_At
nova-scheduler   jupiter                              nova
enabled    :-)   2012-06-20 18:10:24
nova-console     jupiter                              nova
enabled    :-)   2012-06-20 18:10:24
nova-cert        jupiter                              nova
enabled    :-)   2012-06-20 18:10:25
nova-consoleauth jupiter                              nova
enabled    :-)   2012-06-20 18:10:22
```

Configuración de red: VLAN

Como hemos vistos en puntos anteriores, los usuarios organizan sus recursos en el cloud en proyectos. Un royecto es un conjunto de instancias o máquinas virtuales creadas por el usuario. A cada instancia se le asigna una ip privada.

Actualmente podemos configurar la red de nuestra nube de tres formas distintas:

- Flat Network Manager
- Flat DHCP Network Manager
- VLAN Network Mananger

Tenemos que señalar la diferencia entre ip fija o privada, que es la dirección que se le asigna a la instancia desde su creación hasta su destrucción, y las ip flotantes, que son direcciones que dinámicamente se pueden asignar a una instancia y que nos va a permitir acceder a ellas desde una red pública.

En los dos primeros modos de red (Flat Mode) el administrador debe indicar un conjunto de direcciones que serán asignadas como ip fijas o privadas cuando se creen las instancias. La diferencia entre los dos modos es, que mientras en el primero la ip se intecta en la configuración de la máquina (es decir se escribe en el fichero /etc/network/interfaces), en la segunda opción exite un servidor DHCP que es el responsable de asignar las ip a las distintas instancias.

En el modo VLAN, se crea una vlan y un bridge para cada proyecto (tenant), de esta manera conseguimos que las máquinas virtuales pertenecientes a un proyecto reciban direcciones privadas de un rango que será sólo accesible desde la propia vlan, se decir cada proyecto está relacionado con vlan, con lo que conseguimos aislar las instancias de los diferentes proyectos. Esta características es lo que nos ha decidido a escoger este modo de red para nuestra infraestructura.

Configuración del tipo de red VLAN

Los requisitos para utilizar el modo VLAN como configuración de red son los siguientes:

- El ip forwading debe estar habilitado.
- Los nodos de computación (que tienen instalado nova-network y nova-compute) tienen que tener cargado el módulo del kernel 8021q.
- Los switches de la red deben soportar la configuración de vlan.
- Los switches de de la red deben estar configurados con las mismas vlan que hayamos creado.

Para configurar este modo de red tenemos que añadir al fichero de configuración `/etc/nova/nova.conf`:

```
network_manager=nova.network.manager.VlanManager
vlan_interface=eth1
fixed_range=10.0.0.0/8
network_size=256
```

En la primera línea se especifica que vamos a usar el modo de red VLAN. Con la directiva **vlan_interface** indicamos la interfaz de red con la que se va asociar el bridge que se ha creado. La directiva **fixed_range** indica el rango completo de ip que se va a dividir en subredes para cada vlan creada. Por último, la directiva **network_size** indica el tmaña de cada subred asociada a cada vlan.

Creación de un VLAN

Como cada proyecto (tenant) se asocia a una vlan, debemos crear el número de vlan suficientes para los proyectos que en el futuro vayamos a crear. Ahora mismo, vamos a crear una sola vlan que se asociará a un sólo proyecto, para ello usamos el comando **nova-manage network create**:

```
nova-manage network create --label=vlan1
--fixed_range_v4=10.0.1.0/24 --vlan=1 --bridge=br1
```

Con este comando hemos creado la vlan1 que tendrá la etiqueta vlan "1" y que posee como rango de ip la 10.0.1.0/24. Todas las instancias que usen esta vlan estarán conectadas al bridge br1.

Cuando creemos una instancia de un proyecto, dicho proyecto se asocia a una vlan que este libre, a partir de entonces todas las instancias de ese proyecto utilizarán la misma vlan.

Creación de un rango de ip flotantes (ip públicas)

Como indicábamos anteriormente a las instancias se le puede asignar dinámicamente una ip pública o flotante que nos permite el acceso a ella desde la red pública. En esta sección vamos a indicar como se crea el rango de ip flotantes.

Nosotros vamos a crear un rango de ip flotante con el siguiente **nova-manage floating create** de la siguiente manera:

```
nova-manage floating create --ip_range=172.22.221.0/24
```

Posteriormente estudiaremos la utilización de estas ips en las distintas instancias.

Nova en los nodos de computación

Este capítulo describe la instalación y configuración de los módulos de nova de OpenStack necesarios en los nodos de computación: nova-network y nova-compute.

Instalación

Instalamos desde el repositorio de Debian:

```
# aptitude install nova-api nova-cert nova-network
     nova-compute
```

De la misma manera que encontramos descrito en la sección de "Nova en el nodo controlador" instalamos en cada nodo de computación los paquetes python-novnc y novnc.

Para continuar la instalación podemos copiar del nodo controlador el fichero `/etc/nova/nova.conf` haciendo las siguiente modificaciones:

- En el parámetro my_ip tenemos que indicar la ip privada del nodo.
- En el parámetro routing_source_ip tenemos que indicar la ip pública del nodo.
- Por último para configurar el acceso por vnc, tenemos que indicar en los parámetros vncserver_proxyclient_address y vncserver_listen la dirección ip pública del servidor.

El siguiente paso es sincronizar la base de datos:

```
nova-manage db sync
```

Y por último reiniciamos los servicios. Realizando un nova-manage service list deberían aparecer los nuevos servicios.

Horizon (dashboard)

Este capítulo describe la instalación y configuración de la aplicación web Horizon, que nos permite realizar distintas funciones en nuestro cloud. Desde ella se pueden editar casi todo lo relativo a la gestión de proyectos (creación, eliminación, gestión de cuotas y usuarios, ...), usuarios, gestión de instancias, imágenes, volúmenes, ...

Instalación

El dashboard Horizon lo vamos a instalar en el nodo controlador Jupiter, para ello instalamos desde el repositorio de Debian:

```
# aptitude install apache2
     openstack-dashboard openstack-dashboard-apache
```

Como podemos observar utilizamos el servidor web Apache2, en el cual se configura un virtual host que podemos configurar en el fichero */etc/apache2/sites-availables/openstack-*

dashboard. Para acceder a la aplicación web podemos acceder desde una navegador a la URL: `http://direccion_ip_jupiter:8080/`.

Como observamos por defecto utiliza el puerto 8080 para el acceso, si queremos utilizar el puerto 80, simplemente debemos modificar el fichero */etc/apache2/sites-availables/openstack-dashboard* y realizar las siguientes modificaciones en las dos primeras líneas:

```
Listen 80
<VirtualHost *:80>
```

Y a continuación desactivamos el virtual host por defecto e reiniciamos el servidor web.

```
# a2dissite default
# service apache2 restart
```

3. Instalación de OpenStack en GNU/ Linux Ubuntu 12.04

Introducción

Esta sección muestra el proceso detallado de instalación y configuración de OpenStack basado en la plataforma Ubuntu 12.04 (Precise Pangolin) usando 5 servidores (nodos). Se usará uno de los servidores como nodo controlador, ejecutando los componentes Nova, Glance, Swift, Keystone y Horizon y el resto de nodos como nodos de computación, ejecutando únicamente Nova Compute.

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: FlatDHCPNetwork.
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:

- OpenStack Open Source Cloud Computing Software.
- OpenStack Docs: Essex.
- OpenStack in Launchpad.

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 tres nodos con Ubuntu 12.04 LTS instalado.
- Uno de los nodos será el nodo controlador, que ejecutará todos los servicios excepto `nova-compute`.
- 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: `calex2010!!`
- El sistema deberá estar actualizado antes de empezar con el proceso de instalación/ configuración:

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

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**: para el nodo controlador, que 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, pero en el que no se ejecutarán 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, europa, ganimedes y calisto** (las 4 lunas principales de júpiter): para los 4 nodos de virtualización o nodos de computación, como se les denomina habitualmente en la jerga

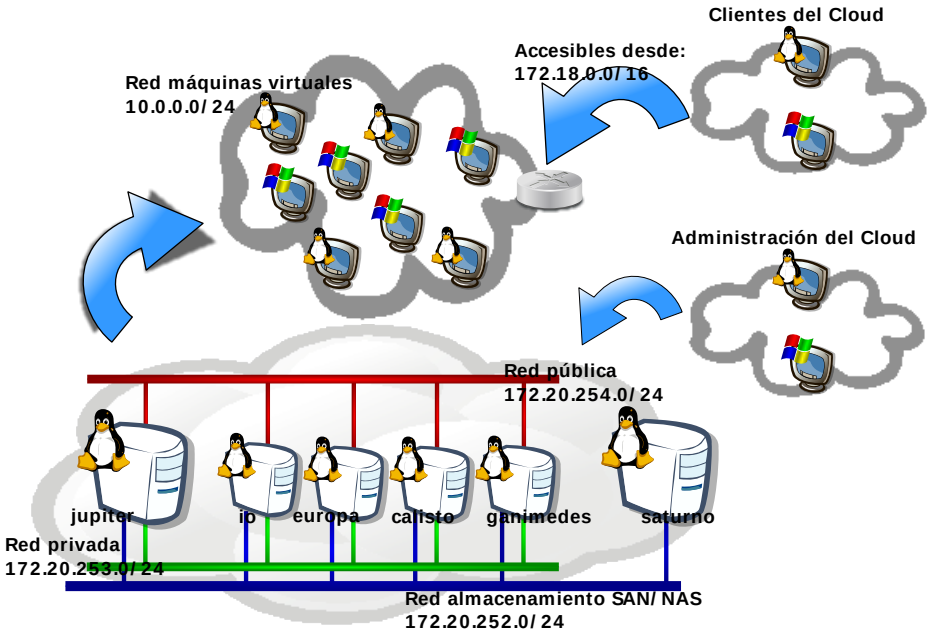
propia de OpenStack. En estos equipos se instalarán sólo los componentes necesarios para que se ejecuten las instancias (máquinas virtuales) en ellos y estarán esperando las órdenes de `jupiter`.

- **saturno**: para el nodo de almacenamiento, ya que es tan importante como júpiter y a la vez independiente. En este equipo todavía no está definido el software que se instalará, pero lo más probable es que sea una distribución especializada en infraestructuras SAN/NAS como `OpenFiler` o `FreeNAS`. Este nodo ofrecerá espacio de almacenamiento extra a través de los protocolos iSCSI y NFS.

- **venus**: 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:

Figure 3.1. Infraestructura Cloud Computing (IES Cierva/IES Los Albares)



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):

Table 3.1. LANs Virtuales del Cloud Computing

Red Virtual	Rango de direcciones IP	Interfaz virtual	Interfaz física (id. VLAN)
Red Pública	172.20.254.0/24	—	bond0 (untagged)
Red Privada Red Máquinas Virtuales	172.20.253.0/24 10.0.0.0/24	bond0.60	bond0 (60)
Red Almacenamiento SAN/NAS	172.20.252.0/24	bond0.62	bond0 (62)
Direcciones IP flotantes	172.18.0.0/16	bond0.61	bond0 (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 del Cloud y para la comunicación de máquinas virtuales.

  En el caso del controlador la 172.20.253.190, en el caso del primer nodo, la 172.20.253.191.

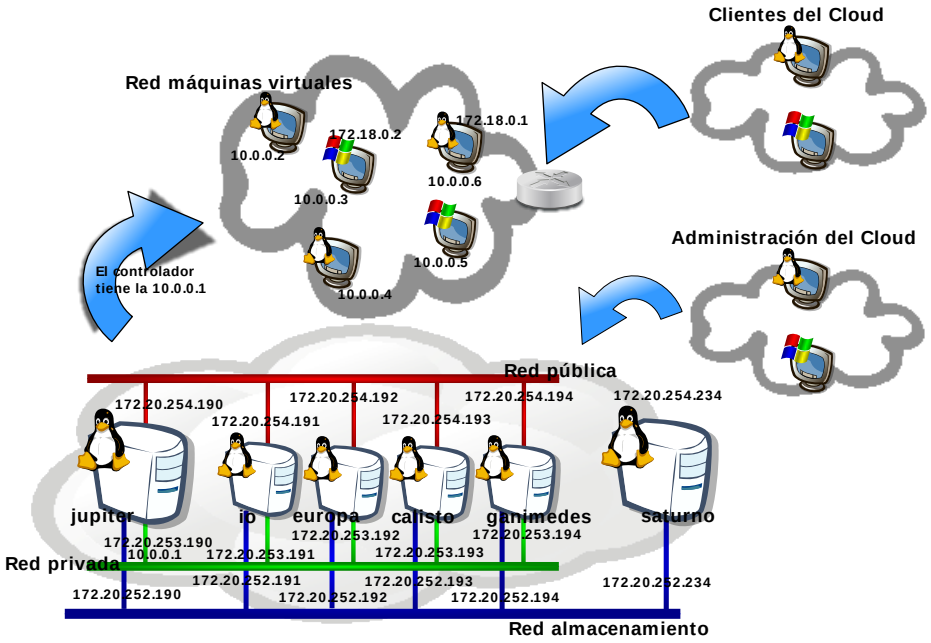
  Esta interfaz también se usa para la comunicación entre las máquinas virtuales a través de la red 10.0.0.0/24.
- La interfaz bond0.62, asociada a la red de almacenamiento, utilizada para el acceso a los servidores que ofrecen SAN y NAS.

  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, utilizada 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:

Figure 3.2. Infraestructura de nuestro Cloud Computing (IES Cierva/IES Los Albares)



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, para tener soporte en Ubuntu tenemos que hacer dos cosas:

1. Instalar el paquete vlan:

```
$ 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:

```
$ apt-get install ifenslave ethtool
```

2. Bajamos las interfaces de red:

```
$ ifdown eth0
$ ifdown eth1
```

3. 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`:

```
# 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 172.20.254.192
      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
```

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:

```
       $ 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
```

7. Podemos comprobar las VLAN activas a través del siguiente comando:

```
$ 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
```

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.

Note

En los nodos de computación solo se han configurado cuatro interfaces de las seis disponibles. Básicamente por si hiciesen falta en un futuro para otro tipo de conectividad.

En `jupiter`, el nodo controlador:

```
#/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 br100
#iface br100 inet dhcp
# bridge_ports bond0.60
#        bridge_stp off
# bridge_maxwait 0
#        bridge_fd 0

# The primary network interface

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

#Manegement Cloud
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

#Public Cloud
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

#Virtual Cloud
auto bond0.61
iface bond0.61 inet static
 address 172.18.0.190
 netmask 255.255.0.0
 broadcast 172.18.255.255
 network 172.18.0.0
 vlan-raw-device bond0

#Floating Cloud
auto bond0.62
iface bond0.62 inet static
 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
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
```

En `io`, uno de los nodos de computación (en el resto de nodos solo habría que cambiar las direcciones IP):

```
#/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 br100
iface br100 inet static
 address 172.20.253.191
 netmask 255.255.255.0
 bridge_stp off
 bridge_fd 0
        bridge_ports bond0.60
        bridge_maxwait 0

#auto br100
#iface br100 inet static
#        bridge_ports bond0.60
#        bridge_stp off
#        bridge_fd 0
#        bridge_maxwait 0

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

#IMPORTANTE: instalar paquetes ifenslave-... y ethtool, y crear /etc/modprobe.
d/bonding.conf con alias bond0 bonding...etc

#Manegement Cloud
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

#Public Cloud
auto bond0.60
iface bond0.60 inet manual
 #address 172.20.253.191
 #netmask 255.255.255.0
 #broadcast 172.20.253.255
 #network 172.20.253.0
 vlan-raw-device bond0

#Virtual Cloud
auto bond0.61
iface bond0.61 inet static
 address 172.18.0.191
 netmask 255.255.0.0
 broadcast 172.18.255.255
 network 172.18.0.0
 vlan-raw-device bond0

#Floating Cloud
auto bond0.62
#iface bond0.62 inet manual
```

```
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
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
```

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:

```
Disposit.   Tamaño    Id    Sistema     SF
/dev/sda1    32GiB    83    Linux       ext4
/dev/sda2     4GiB    82    Linux swap  intercambio
/dev/sda3  -resto-    8e    Linux LVM   volúmenes físicos
```

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.

- `/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.

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 al log del sistema:

```
 [    2.799824] 3ware 9000 Storage Controller device driver for Linux v2.26.
02.014.
 [    2.799867] 3w-9xxx 0000:01:00.0: PCI INT A -> GSI 19 (level, low) -> IRQ
 19
 [    2.799880] 3w-9xxx 0000:01:00.0: setting latency timer to 64
 [    3.080257] 3w-9xxx: scsi6: Found a 3ware 9000 Storage Controller at
 0xfe8df000, IRQ: 19.
 [    3.416117] 3w-9xxx: scsi6: Firmware FE9X 4.08.00.006, BIOS BE9X 4.08.00.
001, Ports: 2.
 [   12.307881] 3w-9xxx: scsi6: ERROR: (0x03:0x0101): Invalid command
 opcode:opcode=0x85.
 [   12.308300] 3w-9xxx: scsi6: ERROR: (0x03:0x0101): Invalid command
 opcode:opcode=0x85.
 [   12.308695] 3w-9xxx: scsi6: ERROR: (0x03:0x0101): Invalid command
 opcode:opcode=0x85.
 [127301.671991] 3w-9xxx: scsi6: AEN: INFO (0x04:0x0029): Verify started:unit=
0.
 [131817.864145] 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:

- Nos descargamos el software desde el link "SUPPORT" de la página http://www.3ware.com (ahora LSI). Se trata de la controladora RAID 3ware 9650SE-2LP.

- Nos descargamos el fichero `3DM2_CLI-Linux_10.2.1_9.5.4.zip`

- Ejecutamos:

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

- 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:

```
$ apt-get install openssh-server
```

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

```
$ apt-get update
$ apt-get upgrade
$ 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 con el siguiente contenido:

   ```
   0 4 * * * ntpdate ntp.iescierva.net ; 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
   # m h  dom mon dow   command
   0 4 * * * ntpdate ntp.iescierva.net ; hwclock -w
   root@jupiter:~#
   ```

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 `calex2010!!`

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 esta otra:

```
bind-address        = 0.0.0.0
```

4. Reiniciamos el demonio de MySQL:

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

5. 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 414
Server version: 5.5.24-0ubuntu0.12.04.1 (Ubuntu)

Copyright (c) 2000, 2011, 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>
```

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`.

Tanto para la instalación de Keystone, como para la instalación del resto de servicios, partiremos de los repositorios oficiales de Ubuntu, basta con que sigamos los siguientes pasos:

1. 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 ON keystone.* to 'keystone'@'%' IDENTIFIED BY
    'calex2010!!';
   mysql> FLUSH PRIVILEGES;
   ```

2. Instalamos todo el software necesario:

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

3. 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
   ```

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

   - La cadena de conexión a la base de datos `keystone` de MySQL.
   - El token administrativo (admin_token).

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

   ```
   admin_token = CALEX2010!!
   connection = mysql://keystone:calex2010!!@172.20.254.190/keystone
   ```

   Ver nota posterior sobre `admin_token`.

5. Reiniciamos el servicio:

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

6. Creamos la base de datos inicial:

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

7. Exportamos las siguientes variables de entorno en el fichero `.bashrc` del usuario que ejecute el cliente, en principio el usuario `root` de la máquina `jupiter`:

```
export SERVICE_ENDPOINT="http://172.20.254.190:35357/v2.0"
export SERVICE_TOKEN=CALEX2010!!
```

8. 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:

Table 3.2. Usuarios, proyectos y roles

Usuarios	Proyectos (tenants)	Roles
admin	admin	admin
nova glance swift	service	admin
admin	admin	Member

   En la tabla podemos leer que al usuario `admin` le daremos el rol `admin` en el proyecto `admin`, o que a los usuarios `nova`, `glance` y `swift` les daremos el rol `admin` en el proyecto `service`.

   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:

```
#!/bin/bash

# Solo hay que modificar estos parámetros
PASSWORD=calex2010!!
EMAIL=alex@iescierva.net
PUBLIC_IP=172.20.254.190
PRIVATE_IP=172.20.253.190
ADMIN_IP=172.20.253.190

# Creación de tenants, usuarios y roles

keystone tenant-create --name admin
keystone tenant-create --name service

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_USER --role $ADMIN_ROLE --tenant_id
 $ADMIN_TENANT

# Añadimos los roles de admin a los usuarios nova, glance y swift en el
 tenant service.
USERS=`keystone user-list | grep True | grep -v admin | tr -d " " | awk -F \
| ' { print $2 } '`

for USER_ID in $USERS
do
    keystone user-role-add --user $USER_ID --role $ADMIN_ROLE --tenant_id
 $SERVICE_TENANT
done

# Añadimos también el rol Member al usuario admin en el tenant admin
keystone user-role-add --user $ADMIN_USER --role $MEMBER_ROLE --tenant_id
 $ADMIN_TENANT

# Creamos los servicios
keystone service-create --name nova --type compute --description "OpenStack
 Compute Service"
keystone service-create --name volume --type volume --description "OpenStack
 Volume Service"
keystone service-create --name glance --type image --description "OpenStack
 Image Service"
keystone service-create --name swift --type object-store --description
 "OpenStack Storage Service"
keystone service-create --name keystone --type identity --description
 "OpenStack Identity Service"
keystone service-create --name ec2 --type ec2 --description "OpenStack EC2
 Service"

# Creamos los endpoints
for service in nova volume glance swift keystone ec2
do
    ID=`keystone service-list | grep $service | tr -d " " | awk -F \|
 ' { print $2 } '`
    case $service in
    "nova"     ) keystone endpoint-create --region myregion --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 --region myregion --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 --region myregion --service_id $ID
 \
                --publicurl   "http://$PUBLIC_IP"':9292/v1' \
                --adminurl    "http://$ADMIN_IP"':9292/v1' \
                --internalurl "http://$PRIVATE_IP"':9292/v1'
    ;;
    "swift"    ) keystone endpoint-create --region myregion --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 --region myregion --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 --region myregion --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 a Keystone, como no hay ningún usuario creado hasta hora, la única forma de autenticarse es a través del token administrativo definido anteriomente.
- **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**: dirección pública de acceso a los servicios del cloud.
- **PRIVATE_IP**: dirección privada de acceso a los servicios del cloud.
- **ADMIN_IP**: utilizaremos la misma dirección que la proporcionada a la red privada.

El script se limita a:

- Crear los proyectos necesarios: `admin` y `service`.
- Crear todos los usuarios y asignarles contraseña y dirección de correo electrónico. Los usuarios creados son: `admin`, `nova`, `glance` y `swift`.
- 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 (`nova`, `volume`, `glance`, `swift`, `keystone` y `ec2`). Define los endpoints y los asigna a los servicios.

9. Verificamos que todo se ha creado correctamente a través de los siguientes comandos (los identificadores no serán los mismos):

```
root@jupiter:~# keystone tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 634675c752634b53879656c81da70a83 | service | True    |
| e7b1868b24a742318f1d73f612ecfe1d | admin   | True    |
+----------------------------------+---------+---------+
root@jupiter:~# keystone user-list
+----------------------------------+---------+-------------------+--------+
|                id                | enabled |       email       |  name  |
+----------------------------------+---------+-------------------+--------+
| 05743001bbf14700bcdf2ecc43edbf9b | True    | alex@iescierva.net | admin  |
| 246ba4e3d81c4ae8bdde8ec5784e74d3 | True    | alex@iescierva.net | swift  |
| 291c58f7258747758d109ecee2eb4a69 | True    | alex@iescierva.net | glance |
| 404dafc53b364a7e9f1476aa98082966 | True    | alex@iescierva.net | nova   |
+----------------------------------+---------+-------------------+--------+
root@jupiter:~# keystone role-list
+----------------------------------+--------+
|                id                |  name  |
+----------------------------------+--------+
| 2a716d669ce349eb88e44049723e0cb7 | admin  |
| 622c78f77bbe4a298452783f25cb4635 | Member |
+----------------------------------+--------+
root@jupiter:~# keystone endpoint-list
+----------------------------------+----------
+--------------------------------------------------
+--------------------------------------------------
+-------------------------------------------+
|                id                |  region  |                    publicurl
                    |                  internalurl                  |
               adminurl                 |
+----------------------------------+----------
+--------------------------------------------------
+--------------------------------------------------
+-------------------------------------------+
| 0dd9427a0a624b38abf1876651ac3b69 | myregion | http://172.20.254.190:8776/
v1/$(tenant_id)s      | http://172.20.253.190:8776/v1/$(tenant_id)s      |
 http://172.20.253.190:8776/v1/$(tenant_id)s |
| 2d471f9b23c84bac91c17615e157bb9f | myregion | http://172.20.254.190:5000/
v2.0                  | http://172.20.253.190:5000/v2.0                  |
 http://172.20.253.190:35357/v2.0            |
```

```
| c9f799e2c6444bbda40dbc059207de09 | myregion | http://172.20.254.190:8773/
services/Cloud        | http://172.20.253.190:8773/services/Cloud        |
 http://172.20.253.190:8773/services/Admin   |
| e72307edc9b84c2d9403b019c3a2e5a7 | myregion | http://172.20.254.190:9292/
v1                    | http://172.20.253.190:9292/v1                    |
 http://172.20.253.190:9292/v1               |
| ec8b9d8d41ea496492730b20e1c34380 | myregion | http://172.20.254.190:8080/
v1/AUTH_$(tenant_id)s | http://172.20.253.190:8080/v1/AUTH_$(tenant_id)s |
 http://172.20.253.190:8080/v1               |
| edf05355fc274c02a13f900605c56769 | myregion | http://172.20.254.190:8774/
v2/$(tenant_id)s      | http://172.20.253.190:8774/v2/$(tenant_id)s      |
 http://172.20.253.190:8774/v2/$(tenant_id)s |
+----------------------------------+----------
+--------------------------------------------------
+--------------------------------------------------
+-----------------------------------------------+
root@jupiter:~# keystone service-list
+----------------------------------+----------+--------------
+---------------------------+
|                id                |   name   |     type     |
 description          |
+----------------------------------+----------+--------------
+---------------------------+
| 04b121f16e6045a79a4601c5171998d3 | glance   | image        | OpenStack
 Image Service    |
| 298de2e4a0f04b42b54818fd57d1bf2e | keystone | identity     | OpenStack
 Identity Service |
| 368b238c02734ee4ac1a6ac6bf440ffa | nova     | compute      | OpenStack
 Compute Service  |
| a075b6add5564f729253dd0339b1d5d9 | volume   | volume       | OpenStack
 Volume Service   |
| c23937017a7e4baa8811a24ff9c8086c | swift    | object-store | OpenStack
 Storage Service  |
| e998168d3d264e3aab37d4b8413fe736 | ec2      | ec2          | OpenStack EC2
 Service      |
+----------------------------------+----------+--------------
+---------------------------+
root@jupiter:~#
```

¿Qué es el ADMIN_TOKEN?

Keystone introduce en OpenStack un sistema de autenticación basado en tokens, de manera que todos los elementos del cloud (usuarios y servicios principalmente), no se autentican directamente unos a otros, sino que lo hace con un actor intermedio mediante tokens, este actor intermedio encargado de verificar la autenticidad de cada uno de los elementos es Keystone.

Con Keystone recién instalado no tenemos ningún usuario con privilegios de administración en la base de datos de Keystone, por lo no podríamos hacer ninguna modificación. El mecanismo que proporciona keystone para solventar esta situación es definir en el fichero de configuración un token con privilegios de administración, que recibe el nombre de ADMIN_TOKEN, que puede definirse directamente en la instalación o cuando queramos en el fichero `/etc/keystone/keystone.conf`, el ADMIN_TOKEN puede tener cualquier valor, pero hay que custodiarlo adecuadamente mientras esté activo, ya que otorga privilegios de administración sobre Keystone a quien lo conozca. Posteriormente, una vez

definidos los usuarios en keystone y los demás elementos, podemos borrar este campo del fichero de configuración de keystone y no volver a utilizarlo.

Instalación 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.

Instalación y configuración

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`.

Tanto para la instalación de Glance, como para la instalación del resto de servicios, partiremos de los repositorios oficiales de Ubuntu, basta con que sigamos los siguientes pasos:

1. 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 ON glance.* to 'glance'@'%' IDENTIFIED BY
    'calex2010!!';
       mysql> FLUSH PRIVILEGES;
   ```

2. Instalamos todo el software necesario:

   ```
       root@jupiter:~# apt-get install glance glance-api glance-client glance-
   common glance-registry python-glance
   ```

3. Modificamos las siguientes líneas del fichero de configuración `/etc/glance/glance-api-paste.ini`, modificando estas líneas:

   ```
   admin_tenant_name = %SERVICE_TENANT_NAME%
   admin_user = %SERVICE_USER%
   admin_password = %SERVICE_PASSWORD%
   ```

   Por estas otras:

   ```
   admin_tenant_name = service
   admin_user = glance
   admin_password = calex2010!!
   ```

4. El mismo cambio anterior hay que hacerlo también en el fichero `/etc/glance/glance-registry-paste.ini`, dejando las últimas líneas así:

```
admin_tenant_name = service
admin_user = glance
admin_password = calex2010!!
```

5. Editamos el fichero `/etc/glance/glance-registry.conf` y modificamos la cadena de conexión a la base de datos MySQL recién creada:

```
sql_connection = mysql://glance:calex2010!!@172.20.254.190/glance
```

   También añadimos la siguiente configuración para que `Glance` utilice `Keystone` como servicio de autenticación.

```
[paste_deploy]
flavor = keystone
```

6. Realizamos también este último cambio al fichero `/etc/glance/glance-api.conf`:

```
[paste_deploy]
flavor = keystone
```

7. Creamos el esquema de la base de datos:

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

   Podemos ignorar los warnings de Python.

8. Reiniciamos los demonios del servicio Glance:

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

9. Añadimos las siguientes variables de entorno en el fichero .bashrc del usuario que ejecute el cliente glance, en principio al usuario `root` de la máquina `jupiter`:

```
export SERVICE_TOKEN=CALEX2010!!
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=calex2010!!
export OS_AUTH_URL="http://172.20.254.190:5000/v2.0/"
export SERVICE_ENDPOINT=http://172.20.254.190:35357/v2.0
```

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
```

Podemos asegurarnos que todo ha ido bien (incluyendo la autenticación con Keystone) si tras ejecutar el comando anterior, la salida del comando **echo $?** nos devuelve como código de salida el cero.

Solo nos queda probar el servicio Glance subiendo alguna imagen de prueba. Para subir una imagen, podemos seguir estos pasos:

1. Descargamos la imagen Cirros desde la siguiente URL:

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

   ### Note

   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.

   Tenemos más imágenes prefabricadas en sitios web como: Ubuntu Cloud Images.

   Más información sobre la creación de imágenes en la sección: "Gestión de Imágenes".

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

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

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
------------------------------------ ------------------------------
 -------------------- -------------------- --------------
fe22ea9c-ebb2-4bb8-be34-ea7732a8a3d2 Cirros (test) 0.3.0 64 bits    qcow2
             bare                         9761280
root@jupiter:~#
```

Instalación 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.

Instalación y configuración

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`, `europa`, `ganimedes` y `calisto`. Inicialmente configuraremos al controlador (`jupiter`) como nodo de computación, pero lo eliminaremos una vez que vayamos añadiendo a los otros nodos.

Tanto para la instalación de Nova, como para la instalación del resto de servicios, partiremos de los repositorios oficiales de Ubuntu, basta con que sigamos los siguientes pasos en la máquina `jupiter`:

1. Instalamos todo el software necesario:

```
root@jupiter:~# apt-get install nova-api nova-cert nova-compute nova-
compute-kvm nova-doc nova-network nova-objectstore nova-scheduler nova-
volume rabbitmq-server novnc nova-consoleauth
```

2. 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 ON nova.* to 'nova'@'%' IDENTIFIED BY 'calex2010!!';
mysql> FLUSH PRIVILEGES;
```

3. Editamos el fichero `/etc/nova/nova.conf` con el siguiente contenido:

```
[DEFAULT]

# LOGS/STATE
verbose=True

# AUTHENTICATION
auth_strategy=keystone
use_deprecated_auth=false

# SCHEDULER
#compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
#scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
scheduler_driver=nova.scheduler.simple.SimpleScheduler

# VOLUMES
volume_group=nova-volumes
```

```
volume_name_template=volume_name_template=volume-%08x
iscsi_helper=tgtadm
iscsi_ip_prefix=10.0.1
iscsi_ip_address=10.0.1.1

# DATABASE
sql_connection=mysql://nova:calex2010!!@172.20.254.190/nova

# COMPUTE
libvirt_type=kvm
libvirt_use_virtio_for_bridges=true
start_guests_on_host_boot=true
resume_guests_state_on_host_boot=true
connection_type=libvirt
instance_name_template=instance-%08x
api_paste_config=/etc/nova/api-paste.ini
allow_resize_to_same_host=True

# APIS
osapi_compute_extension=nova.api.openstack.compute.contrib.
standard_extensions
ec2_host=172.20.254.190
#ec2_dmz_host=172.20.254.190
s3_host=172.20.254.190
ec2_url=http://172.20.254.190:8773/services/Cloud
keystone_ec2_url=http://172.20.254.190:5000/v2.0/ec2tokens
cc_host=172.20.254.190
nova_url=http://172.20.254.190:8774/v1.1/

# RABBITMQ
rabbit_host=172.20.254.190
rabbit_password=guest

# GLANCE
image_service=nova.image.glance.GlanceImageService
glance_api_servers=172.20.254.190:9292

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
my_ip=172.20.254.190
public_interface=bond0
#vlan_interface=eth0
flat_network_bridge=br100
flat_interface=bond0.60
fixed_range=10.0.0.0/24
floating_range=172.18.0.0/27
#floating_range=172.20.254.0/24
routing_source_ip=172.20.254.190
start_guests_on_host_boot=true
resume_guests_state_on_host_boot=true
network_size=32
flat_network_dhcp_start=10.0.0.2
flat_injected=False
force_dhcp_release=True
root_helper=sudo nova-rootwrap
```

```
# LOGS
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
allow_admin_api=true

# VNC
novnc_enabled=true
vnc_keymap=es
novncproxy_base_url=http://172.20.254.190:6080/vnc_auto.html
vncserver_proxyclient_address=172.20.254.190
vncserver_listen=172.20.254.190
vnc_console_proxy_url=http://172.20.254.190:6080
```

4. Para el trabajo con volúmenes, debemos tener configurado un grupo de volúmenes LVM denominado `nova-volumes`, tal como hemos definido a través del parámetro `volume-group` en el fichero `nova.conf`. Para crear el volumen ejecutamos los siguientes comandos, suponiendo una partición `sda4` libre:

```
root@jupiter:~#  pvcreate /dev/sda4
root@jupiter:~#  vgcreate nova-volumes /dev/sda4
```

5. Cambiamos los permisos del directorio /etc/nova y del fichero /etc/nova/nova.conf:

```
root@jupiter:~# chown -R nova.nova /etc/nova
root@jupiter:~# chmod 644 /etc/nova/nova.conf
```

6. Editamos el fichero /etc/nova/

7. Modificamos las siguientes líneas del fichero de configuración `/etc/nova/api-paste.ini`, modificando estas líneas:

```
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
```

   Por estas otras:

```
admin_tenant_name = service
admin_user = nova
admin_password = calex2010!!
```

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

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

   Podemos ignorar los warnings de Python.

9. Reiniciamos todos los demonios del servicio Nova:

```
root@jupiter:~# service novnc stop
root@jupiter:~# service nova-consoleauth stop
root@jupiter:~# service nova-volume stop
root@jupiter:~# service nova-schedule  stop
root@jupiter:~# service nova-objectstore stop
root@jupiter:~# service nova-api stop
root@jupiter:~# service nova-compute stop
root@jupiter:~# service nova-network stop
root@jupiter:~# service nova-cert stop
root@jupiter:~# service libvirt-bin stop

root@jupiter:~# service libvirt-bin start
root@jupiter:~# service nova-cert start
root@jupiter:~# service nova-network start
root@jupiter:~# service nova-compute start
root@jupiter:~# service nova-api start
root@jupiter:~# service nova-objectstore start
root@jupiter:~# service nova-schedule  start
root@jupiter:~# service nova-volume start
root@jupiter:~# service nova-consoleauth start
root@jupiter:~# service novnc start
```

Mientras que trabajemos con el servicio Nova (instalación, configuración, pruebas...), va a ser muy frecuente tener que reiniciar todos sus servicios asociados, podemos facilitar esta tarea a través del siguiente script, que podremos guardar en `/root/bin/nova.sh`:

```
#!/bin/bash
SERVICIOS="libvirt-bin nova-cert nova-network nova-compute nova-api nova-
objectstore nova-scheduler nova-volume nova-consoleauth novnc"

REVERSE="novnc nova-consoleauth nova-volume nova-scheduler nova-objectstore
 nova-api nova-compute nova-network nova-cert libvirt-bin"

function iniciarServiciosNova()
{
 /etc/init.d/rabbitmq-server start
 for servicio in $SERVICIOS
 do
  service $servicio start
 done
}

function pararServiciosNova()
{
 for servicio in $REVERSE
 do
  service $servicio stop
 done
 /etc/init.d/rabbitmq-server stop
}


case $1 in
```

```
 start)
  echo "Iniciando todos los servicios nova"
  iniciarServiciosNova
 ;;

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

 restart)
  echo "Parando todos los servicios nova"
  pararServiciosNova
  echo "Iniciando todos los servicios nova"
  iniciarServiciosNova
 ;;

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

10. Proporcionamos una red (rango de IPs) que asociaremos a las instancias, lo podemos hacer a través del siguiente comando:

```
root@jupiter:~# nova-manage network create private --fixed_range_v4=10.
0.0.0/24 --num_networks=1 --bridge=br100 --bridge_interface=bond0.60 --
network_size=256 --dns1=172.20.254.235
```

11. Nos aseguramos de tener definidas las siguientes variables de entorno en la máquina que haga de cliente Nova:

```
export SERVICE_TOKEN=CALEX2010!!
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=calex2010!!
export OS_AUTH_URL="http://172.20.254.190:5000/v2.0/"
export SERVICE_ENDPOINT=http://172.20.254.190:35357/v2.0
```

12. Volvemos a reiniciar todos los servicios:

```
root@jupiter:~# /root/bin/nova.sh restart
```

13. Comprobamos que todos los servicios estén iniciados, lo podemos hacer a través del comando:

```
root@jupiter:~# nova-manage service list
```

...que nos debería proporcionar una salida similar a la siguiente:

```
root@jupiter:~# nova-manage service list
Binary           Host                                 Zone
 Status     State Updated_At
nova-consoleauth jupiter                              nova
 enabled    :-)   2012-07-28 10:02:04
nova-cert        jupiter                              nova
 enabled    :-)   2012-07-28 10:02:05
nova-scheduler   jupiter                              nova
 enabled    :-)   2012-07-28 10:02:03
nova-compute     jupiter                              nova
 enabled    :-)   2012-07-28 10:01:57
nova-volume      jupiter                              nova
 enabled    :-)   2012-07-28 10:02:04
nova-network     jupiter                              nova
 enabled    :-)   2012-07-28 10:02:03
```

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 el servicio está realmente en ejecución con el comando (en el caso de `nova-compute`):

```
root@jupiter:~# ps ax | grep nova-compute
```

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
```

Note

El servicio `nova-compute` suele mostrar los caracteres XXX durante bastante tiempo, no mostrará una cara feliz mientras haya máquinas virtuales sin iniciar.

14. Arrancamos una imagen de prueba.

Podemos comprobar las imágenes que podemos iniciar a través del servicio Glance con cualquiera de los siguientes comandos:

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

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

Generamos las claves SSH necesarias para conectarnos a través del protocolo SSH a las máquinas virtuales una vez iniciadas:

```
root@jupiter:~# nova keypair-add test > /root/test.pem
```

Para iniciar una instancia necesitamos el nombre de la imagen, el flavor (podemos crear más) y las claves SSH anteriormente generadas, ejecutamos el siguiente comando:

```
root@jupiter:~# nova boot --image "Cirros (test) 0.3.0 64 bits" --flavor m1.
small --key_name test my-first-server
```

Podemos comprobar el estado de la instancia a través del comando:

```
root@jupiter:~# nova list
```

... que debería mostrar una salida como ésta:

```
root@jupiter:~# nova list
+--------------------------------------+-----------------+--------
+------------------+
|                  ID                  |       Name      | Status |
 Networks     |
+--------------------------------------+-----------------+--------
+------------------+
| 95db6cb7-6d29-4728-8a57-00f0a16fdf0f | my-first-server | ACTIVE | private=
10.0.0.2 |
+--------------------------------------+-----------------+--------
+------------------+
root@jupiter:~#
```

Y obtener más información a través del UID de la instancia con el comando:

```
root@jupiter:~# nova show <id>
```

... de esta forma:

```
root@jupiter:~# nova show 95db6cb7-6d29-4728-8a57-00f0a16fdf0f
+-------------------------------------
+-------------------------------------------------------------+
|               Property              |                          Value
                    |
+-------------------------------------
+-------------------------------------------------------------+
| OS-DCF:diskConfig                   | MANUAL
                    |
| OS-EXT-SRV-ATTR:host                | jupiter
                    |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None
                    |
| OS-EXT-SRV-ATTR:instance_name       | instance-00000001
                    |
| OS-EXT-STS:power_state              | 1
                    |
```

```
| OS-EXT-STS:task_state               | None
                    |
| OS-EXT-STS:vm_state                 | active
                    |
| accessIPv4                          |
                    |
| accessIPv6                          |
                    |
| config_drive                        |
                    |
| created                             | 2012-07-25T09:02:58Z
                    |
| flavor                              | m1.small
                    |
| hostId                              |
 3321967f22b94709966c1525bfb8fe73e88019b1a03228af56e8b847 |
| id                                  | 95db6cb7-6d29-4728-8a57-00f0a16fdf0f
                    |
| image                               | Cirros (test) 0.3.0 64 bits
                    |
| key_name                            | test
                    |
| metadata                            | {}
                    |
| name                                | my-first-server
                    |
| private network                     | 10.0.0.2
                    |
| progress                            | 0
                    |
| status                              | ACTIVE
                    |
| tenant_id                           | e7b1868b24a742318f1d73f612ecfe1d
                    |
| updated                             | 2012-07-26T06:49:51Z
                    |
| user_id                             | 05743001bbf14700bcdf2ecc43edbf9b
                    |
+-------------------------------------
+-------------------------------------------------------------+
root@jupiter:~#
```

Nos podemos conectar a través de SSH a la instancia con el comando:

```
$ ssh -i test.pem cirros@10.0.0.2
```

Al iniciar la primera máquina virtual, en el nodo controlador se creará el bridge `br100` con una IP en el rango dado, en nuestro caso la dirección 10.0.0.1. Es importante destacar el hecho de que la dirección IP de la interfaz asociada al bridge "desaparece" pero se sigue utilizando, tal como muestra la ejecución de los siguientes comandos:

```
root@jupiter:~# ifconfig bond0.60
bond0.60  Link encap:Ethernet  direcciónHW 00:25:90:72:2c:47
          Dirección inet6: fe80::225:90ff:fe72:2c47/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:30800 errores:0 perdidos:0 overruns:0 frame:0
```

```
          Paquetes TX:45655 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:0
          Bytes RX:4491363 (4.4 MB)  TX bytes:9278130 (9.2 MB)

root@jupiter:~# ifconfig br100
br100     Link encap:Ethernet  direcciónHW 00:25:90:72:2c:47
          Direc. inet:10.0.0.1  Difus.:10.0.0.255  Másc:255.255.255.0
          Dirección inet6: fe80::c035:49ff:fe37:96e5/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:77878 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:53900 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:0
          Bytes RX:15617344 (15.6 MB)  TX bytes:9926875 (9.9 MB)

root@jupiter:~# ping -c 5 172.20.253.190
PING 172.20.253.190 (172.20.253.190) 56(84) bytes of data.
64 bytes from 172.20.253.190: icmp_req=1 ttl=64 time=0.070 ms
64 bytes from 172.20.253.190: icmp_req=2 ttl=64 time=0.048 ms
64 bytes from 172.20.253.190: icmp_req=3 ttl=64 time=0.046 ms
64 bytes from 172.20.253.190: icmp_req=4 ttl=64 time=0.042 ms
64 bytes from 172.20.253.190: icmp_req=5 ttl=64 time=0.050 ms

--- 172.20.253.190 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3996ms
rtt min/avg/max/mdev = 0.042/0.051/0.070/0.010 ms
root@jupiter:~#
```

Instalación de un segundo nodo

Podemos añadir a nuestra infraestructura tantos nodos de computación como deseemos, para ello basta seguir los siguientes pasos:

1. No instalamos todo el software de Nova, solo un paquete, además de las herramientas para la configuración de bridges:

   ```
   root@io:~# apt-get install nova-compute bridge-utils
   ```

2. El fichero de configuración de Nova, `/etc/nova/nova.conf`, no es exactamente el mismo que en el controlador, hay que realizar los siguientes cambios:

   ```
   my_ip=172.20.254.191
   ...
   vncserver_proxyclient_address=172.20.253.191
   vncserver_listen=172.20.253.191
   ```

   ... quedando de la siguiente forma. Nos aseguramos antes de los permisos (`600`) y del propietario y grupo (`nova.nova`):

   ```
   [DEFAULT]

   # LOGS/STATE
   verbose=True
   ```

```
# AUTHENTICATION
auth_strategy=keystone
use_deprecated_auth=false

# SCHEDULER
#compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
#scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
scheduler_driver=nova.scheduler.simple.SimpleScheduler

# VOLUMES
volume_group=nova-volumes
volume_name_template=volume_name_template=volume-%08x
iscsi_helper=tgtadm
iscsi_ip_prefix=10.0.1
iscsi_ip_address=10.0.1.1

# DATABASE
sql_connection=mysql://nova:calex2010!!@172.20.254.190/nova

# COMPUTE
libvirt_type=kvm
libvirt_use_virtio_for_bridges=true
start_guests_on_host_boot=true
resume_guests_state_on_host_boot=true
connection_type=libvirt
instance_name_template=instance-%08x
api_paste_config=/etc/nova/api-paste.ini
allow_resize_to_same_host=True

# APIS
osapi_compute_extension=nova.api.openstack.compute.contrib.
standard_extensions
ec2_host=172.20.254.190
#ec2_dmz_host=172.20.254.190
s3_host=172.20.254.190
ec2_url=http://172.20.254.190:8773/services/Cloud
keystone_ec2_url=http://172.20.254.190:5000/v2.0/ec2tokens
cc_host=172.20.254.190
nova_url=http://172.20.254.190:8774/v1.1/

# RABBITMQ
rabbit_host=172.20.254.190
rabbit_password=guest

# GLANCE
image_service=nova.image.glance.GlanceImageService
glance_api_servers=172.20.254.190:9292

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
my_ip=172.20.254.191
public_interface=bond0
#vlan_interface=eth0
flat_network_bridge=br100
flat_interface=bond0.60
fixed_range=10.0.0.0/24
```

```
floating_range=172.18.0.0/27
#floating_range=172.20.254.0/24
routing_source_ip=172.20.254.190
start_guests_on_host_boot=true
resume_guests_state_on_host_boot=true
network_size=32
flat_network_dhcp_start=10.0.0.2
flat_injected=False
force_dhcp_release=True
root_helper=sudo nova-rootwrap

# LOGS
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
allow_admin_api=true

# VNC
novnc_enabled=true
vnc_keymap=es
novncproxy_base_url=http://172.20.254.190:6080/vnc_auto.html
vncserver_proxyclient_address=172.20.253.191
vncserver_listen=172.20.253.191
vnc_console_proxy_url=http://172.20.254.190:6080
```

3. Borramos la base de datos SQLite que acompaña a la instalación de nova-compute:

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

4. Revisamos los permisos del directorio `/var/lib/nova`:

```
root@io:~# chown -R nova.nova /var/lib/nova
root@io:~# chmod 770 /var/lib/nova
```

5. Si queremos que funcione el cliente nova, al menos para el usuario `root`, añadimos las siguientes variables de entorno a su fichero `~/.bashrc`:

```
export SERVICE_TOKEN=CALEX2010!!
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=calex2010!!
export OS_AUTH_URL="http://172.20.254.190:5000/v2.0/"
export SERVICE_ENDPOINT=http://172.20.254.190:35357/v2.0
```

6. Reiniciamos el servicio `nova-compute`:

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

7. Si todo ha ido bien, ahora desde el controlador se debemos poder obtener la siguiente salida...

```
root@jupiter:~# nova-manage service list
Binary           Host                                 Zone
 Status     State Updated_At
nova-consoleauth jupiter                              nova
 enabled    :-)   2012-07-31 15:47:23
nova-cert        jupiter                              nova
 enabled    :-)   2012-07-31 15:47:25
nova-scheduler   jupiter                              nova
 enabled    :-)   2012-07-31 15:47:24
nova-compute     jupiter                              nova
 enabled    :-)   2012-07-31 15:47:21
nova-volume      jupiter                              nova
 enabled    :-)   2012-07-31 15:47:25
nova-network     jupiter                              nova
 enabled    :-)   2012-07-31 15:47:27
nova-compute     io                                   nova
 enabled    :-)   2012-07-31 15:47:29
```

... en la que se puede ver el nuevo nodo de computación, `io`.

Instalación de Nova: problemas y puntualizaciones

1. Nada más añadir un nodo de computación, todo funcionará perfectamente, incluyendo la configuración automática del bridge `br100`, pero todo dejará de hacerlo mostrando diversos errores en los logs de `nova-compute` como este: "Cannot get interface MTU on 'br100': No such device", tras reiniciar el nuevo nodo.

   Para solucionar este problema hay que:

   a. Configurar el bridge manualmente y asociarle la interfaz.

   b. Asociar manualmente la dirección IP de la interfaz al bridge.

   Basta con modificar el fichero `/etc/network/interfaces` realizando los siguientes cambios:

```
auto br100
iface br100 inet static
    address 172.20.253.191
    netmask 255.255.255.0
    bridge_stp off
    bridge_fd 0
    bridge_ports bond0.60
    bridge_maxwait 0
```

```
auto bond0.60
iface bond0.60 inet manual
    #address 172.20.253.191
    #netmask 255.255.255.0
    #broadcast 172.20.253.255
    #network 172.20.253.0
    vlan-raw-device bond0
```

Tras estos cambios todo funcionará como debería, parece ser que se trata de un bug de OpenStack documentado, pero aún no solucionado en la versión Essex que acompaña a Ubuntu:

- https://bugs.launchpad.net/nova/+bug/1010927
- https://review.openstack.org/#/c/8423/

El nodo controlador no está afectado por este problema, pero de vez en cuando, el proceso nova-compute no se inicia tras reiniciar el servidor. Basta volver a iniciar el servicio con el comando **service nova-compute start**.

2. Una vez que tengamos máquinas virtuales en ejecución hay un log muy recurrente en los logs del servicio `libvirtd`: "Cannot find 'pm-is-supported' in path: No such file or directory". Este log se elimina instalando el siguiente paquete en todos los nodos de comutación:

```
root@io:~# apt-get install pm-utils
```

3. Podemos borrar una instancia a través del comando de OpenStack **nova delete**, pero en determinadas situaciones este comando puede fallar. Si queremos eliminar de forma manual una instancia debemos seguir los siguientes pasos:

   a. Eliminamos la máquina virtual a través del comando **virsh destroy <domain>**.

   b. Nos hacemos con su identificador a través de la siguiente consulta en la base de datos `nova`:

```
mysql> select id, uuid from instances;
```

   Tomando el id de la instancia que queremos borrar, supongamos que queremos eliminar la máquina virtual con `id=99`.

   c. Ejecutamos las siguientes consultas SQL en la base de datos de `nova`:

```
mysql> delete from instance_info_caches where id=99;
mysql> delete from security_group_instance_association where id=99;
mysql> delete from instances where id=99;
```

   d. Comprobamos a través del comando **nova list** que la instancia ya no está en ejecución.

4. Entre pruebas de configuración puede que sea interesante desinstalar por completo nova y sus servicios asociados, para ello seguiremos los siguientes pasos:

   a. Detenemos todas las máquinas virtuales que estuvieran en ejecución:

```
 root@io:~# for i in `virsh list | tail -n +2 | awk ' { print $2 }'`; do
virsh destroy $i; done
```

b. Paramos todos los servicios:

```
root@io:~# /root/bin/nova.sh stop
```

c. Matamos todos los procesos `dnsmasq`:

```
root@io:~# for i in `pgrep dnsmasq` ; do kill $i ; done
```

d. Eliminamos el bridge `br100`:

```
root@io:~# ifconfig br100 down
root@io:~# brctl delbr br100
```

e. Eliminamos los siguientes ficheros y directorios:

```
root@io:~# rm /etc/libvirt/qemu/instance-*xml -f
root@io:~# rm /etc/libvirt/nwfilter/nova-instance-instance-*xml -f
root@io:~# rm /var/lib/nova/buckets/ -rf
root@io:~# rm /var/lib/nova/CA/ -rf
root@io:~# rm /var/lib/nova/instances/* -rf
root@io:~# rm /var/lib/nova/keys/ -rf
root@io:~# rm /var/lib/nova/networks/* -rf
root@io:~# rm /var/lib/rabbitmq/mnesia/* -rf
```

f. Borramos la base de datos de `nova`:

```
mysql> drop database nova;
```

Instalación de Horizon

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.

Instalación y configuración

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

Tanto para la instalación de Horizon, como para la instalación del resto de servicios, partiremos de los repositorios oficiales de Ubuntu, basta con que sigamos los siguientes pasos en la máquina `jupiter`:

1. Instalamos todo el software necesario:

```
root@jupiter:~# apt-get install openstack-dashboard
```

2. Reiniciamos el servicio:

```
root@jupiter:~# service apache2 restart
```

3. Probamos el servicio visitando con un navegador web la siguiente URL: `http://127.0.0.1` y utilizando como credenciales las de los usuarios creados anteriormente durante la instalación de Keystone. Por ejemplo, usuario `admin` y contraseña `calex2010!!`

4. Acceso desde otra máquina.

   Podemos acceder también al dashboard desde otra máquina a través de la IP asociada desde la red pública. En nuestra infraestructura la siguiente URL: `http:://172.20.254.190`

Instalación del cliente administrativo

Durante la instalación que acabamos de ver, el nodo controlador y posiblemente el nodo de computación se han configurado también como cliente de la infraestructura Cloud Computing que OpenStack proporciona. Si queremos instalar un cliente de administración para OpenStack en un host aparte (con el nombre `venus` tan solo tenemos que seguir los siguientes pasos:

1. Realizar una instalación de Ubuntu 12.04 LTS Desktop (a partir de los CDs `desktop` ó `alternate`).

2. Configurar el cliente NTP tal como se vió para los servidores:

   a. Nos aseguramos que el paquete `ntpdate` esté instalado:

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

   Si no lo estuviera lo instalamos:

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

   b. 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 con el siguiente contenido:

   ```
   0 4 * * * ntpdate ntp.iescierva.net ; 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@venus:~# crontab -l
# m h  dom mon dow   command
0 4 * * * ntpdate ntp.iescierva.net ; hwclock -w
root@venus:~#
```

3. Instalamos las herramientas cliente. Utilizaremos este host para la gestión completa de OpenStack, incluyendo los servicios de gestión de instancias (máquinas virtuales) y de gestión de imágenes. Al tratarse de un host con interfaz gráfica, siempre podremos iniciar un navegador y administrar nuestro cloud a través del Dashboard, pero las herramientas de consola siempre resultan ser más útiles. Las instalamos:

```
root@venus:~# apt-get install glance-client python-novaclient
```

4. Exportamos las siguientes variables de entorno en el fichero `.bashrc` del usuario que vaya a utilizar el cliente de OpenStack:

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=calex2010!!
export OS_AUTH_URL="http://172.20.254.190:5000/v2.0/"
export SERVICE_ENDPOINT="http://172.20.254.190:35357/v2.0"
```

5. Ya podemos probar el cliente con cualquiera de los siguientes comandos: **nova list**, **nova show**, **nova boot**, **glance index**, etc.

   Por supuesto, el Dashboard estará disponible a través del navegador en la URL `http://172.20.254.190`

4. Gestión de la identidad

Autenticación con LDAP

Cuando existen muchas aplicaciones que requieren autenticación de los mismos usuarios, dentro de las organizaciones es habitual optar por alguna solución que permita unificar el mecanismo de autenticación. Una de las opciones más extendidas es la utilización de un servidor LDAP para almacenar la información de los usuarios de la organización, incluyendo el nombre de usuario y la contraseña para las diferentes aplicaciones. En esta sección vamos a explicar los pasos que son necesarios para conseguir que keystone utilice LDAP como *backend* y así todos los usuarios de la organización puedan utilizar directamente el cloud de OpenStack.

Configuración del directorio

El directorio debe almacenar información de los usuarios, proyectos y roles del cloud, por lo que habrá que definir claramente las clases de objetos (*ObjectClass*) que los definirán y las ramas en las que se ubicarán.

- Configuración de usuarios

  Para definir los usuarios se utiliza el *objectClass* estándar

  ```
  inetOrgPerson
  ```

  , que debe contener al menos los atributos ...

Suponemos que en el directorio LDAP ya existe una rama para definir los usuarios y otra para los grupos (que utilizaremos para definir los roles), pero vamos a crear una nueva rama para definir los proyectos:

```
dn: ou=Tenant,dc=gonzalonazareno,dc=org
ou: Tenant
objectClass: top
objectClass: organizationalUnit
```

Para definir cada proyecto, utilizaremos un objeto con los siguientes atributos:

```
dn: cn=prueba,ou=Tenant,dc=gonzalonazareno,dc=org
objectclass: groupOfNames
cn: prueba
description: Proyecto de prueba
member: uid=alberto.molina,ou=People,dc=gonzalonazareno,dc=org
```

Y por último, para definir el rol del usuario, utilizamos un objeto como el siguiente:

```
dn:
```

Nota: No está nada claro la configuración con LDAP. En principio yo pensaba que se utilizaba LDAP sólo para la autenticación de los usuarios, mientras que el resto de la información (roles, tenants, etc) seguía estando en las bases de datos, pero parece ser que no es así, que ldap es un backend completo alternativo a las bases de datos, por lo que la configuración es bastante más compleja.

Referencias

Adam Young. Openstack Keystone LDAP Redux

5. Gestión de imágenes

OpenStack utiliza imágenes de sistemas operativos previamente instalados para crear las instancias que se ejecutan en los nodos de computación. Estas imágenes pueden tener diferentes formatos, pueden ser sistemas operativos limpios recién instalados o por contra sistemas con muchas aplicaciones completamente configuradas y pueden ser incluso instantáneas (*snapshots*) de instancias que se están ejecutando en alguno de los nodos de computación.

El componente de OpenStack encargado de la gestión de instancias es Glance que inicialmente almacena las imágenes en el directorio `/var/lib/glance/images`, aunque es posible utilizar también un componente de almacenamiento de objetos como OpenStack Swift.

Formato de Disco y Contenido

Cuando añadimos una imagen a Glance, necesitamos especificar el formato de disco de la maquina virtual y el formato del contenido de éste.

Formato del disco

Tendremos que dar un formato a nuestra imagen para cuando creemos una máquina virtual. Existen muchos tipos de formatos de imagen de disco, actualmente podemos utilizar los siguientes en OpenStack (Disk and Container Formats):

`raw` Formato de imagen de disco no estructurado. Es el formato básico que puede crearse por ejemplo con la aplicación dd, no está optimizado para su utilización en virtualización, pero puede manejarse con herramientas básicas del sistema (dd, parted, fdisk, mount, kpartx, ...)

`vhd` Usado por herramientas de virtualización como VMWare, Xen, Microsoft, VirtualBox y otros.

`vmdk` Otro formato usado por herramientas de virtualización como por ejemplo VMWare player.

`vdi` Formato soportado por VirtualBox y el emulador QEMU

`iso` Formato de ficheros o contenido de un dispostivos óptico como un CDROM o DVD.

`qcow2` Formato de disco soportado por el emulador QEMU que permite expandir dinámicamente y soporta Copy on Write.

`aki` Indica que la imagen guardada en Glance es una Amazon Kernel Image.

`ari` Indica que la imagen guardada en Glance es una Amazon Ramdisk Image.

`ami` Indica que la imagen guardada en Glance es una Amazon Machine Image.

Formato del contenedor

El formato del contenedor se refiere a si la imagen de la máquina virtual está en un formato de archivo que contiene metadatos.

Nota: El contenido de la imagen no es utilizado por Glance o otros componentes de Openstack actualmente, así que si no estamos seguros de que formato poner, utilizaremos *bare*.

Actualmente podemos utilizar uno de los siguientes tipos de formato para el formato del contenedor:

bare Esto indica que no existe formato sobre los metadatos de la imagen.

ovf *Open Virtualization Format*Estándar abierto que están implementando diferentes soluciones de virtualización/cloud.

aki Indica que la imagen guardada en Glance es una Amazon Kernel Image.

ari Indica que la imagen guardada en Glance es una Amazon Ramdisk Image.

ami Indica que la imagen guardada en Glance es una Amazon Machine Image.

Imágenes disponibles en Internet

La forma más sencilla de añadir imágenes a OpenStack es utilizar imágenes preparadas por terceros para OpenStack y que están disponibles en distintos sitios de Internet. Algunos sistemas operativos, sobre todo distintas distribuciones GNU/Linux, además de proporcionar las imágenes ISO con el instalador del sistema, proporcionan imágenes de disco preparadas para arrancar directamente sobre el cloud, siendo fundamental verificar que el formato de disco y contenido son de los soportados por OpenStack.

Dentro de launchpad existe el proyecto CirrOS que se dedica a desarrollar una distribución GNU/Linux muy ligera orientada a su utilización en la nube y que es ideal para pruebas y situaciones en las que se requieran muchas instancias sencillas. Para instalar la última imagen de CirrOS en el equipo en el que se ejecute glance, basta con descargar la imagen adecuada de https://launchpad.net/cirros/+download, por ejemplo, podemos descargar una imagen en formato de disco qcow para x86_64:

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

y posteriormente agregarla a glance:

```
glance add name="CirrOS x86_64 limpia" is_public=true
container_format=bare disk_format=qcow2 < cirros-0.3.0-x86_64-
disk.img
```

`add` Para añadir una imagen

`name` Nombre con el que se guardará la imagen y que se utilizará para identificarla, por lo que debe ser suficientemente descriptiva.

`is_public` Parámetro booleano que indica si la imagen será pública o no.

`container_format` Hay que indicar el formato del contenedor

`disk_format` Hay que indicar el formato del disco

Dentro de las distribuciones GNU/Linux "convencionales", Ubuntu es la que ha hecho un esfuerzo mayor y ha creado un repositorio completo de sus sistemas en varios formatos adecuados para utilizar en la nube y que se pueden descargar de Ubuntu Cloud Images, basta con elegir una imagen para descargar e instalarla con glance de forma análoga a lo que se ha hecho con CirrOS.

En cualquier caso, por muchas imágenes que tengamos a nuestra disposición en Internet, nunca se pueden cubrir todas nuestras necesidades y antes o después tendremos la necesidad de crear una instancia a medida, que es lo que explicamos en las próximas secciones.

Procedimiento general de provisión de imágenes

En secciones posteriores se explicará la forma específica de crear imágenes del cloud para una distribución GNU/Linux o para un sistema Windows, pero vamos a resumir los pasos generales.

La instalación debemos hacerla en un equipo con KVM instalado y configurado y con un cliente VNC disponible[1]. Los pasos principales para crear una imagen con un sistema son:

1. Crear una imagen de disco con un determinado formato en nuestra máquina. Esta imagen se corresponde con el disco duro de la máquina virtual. Por ejemplo para crear una imagen de 1 GiB en formato qcow2:

   ```
   kvm-img create -f qcow2 Nombre-imagen.img 1G
   ```

2. Arrancar la máquina virtual con KVM incluyendo la iso con el instalador del SO. Esto levantará una conexión VNC a las que nos tendremos que conectar para hacer la instalación. Por ejemplo:

   ```
   kvm -m 512 -cdrom instalador-SO.iso -drive file=Nombre-
   imagen.img,if=scsi,index=0 -boot d -net nic -net user -nographic
   -vnc
   ```

3. Conectamos mediante VNC a la máquina virtual y terminar la instalación

4. Copiar la imagen al nodo de glance y añadirla a la base de datos, especificando si la imagen será pública o no y el formato del contenedor:

   ```
   glance add name="Debian Squeeze 6.0.5" is_public=true
   container_format=ovf disk_format=qcow2 < Nombre-imagen.img
   ```

Creación de imágenes GNU/Linux

En primer lugar creamos un fichero para la imagen del sistema (en este caso se ha optado por formato raw):

[1]Podría hacerse sin vnc directamente con sdl, pero VNC es más versátil y nos permite por ejemplo trabajar en remoto

```
kvm-img create -f raw Debian6.0.5.img 1G
```

Iniciamos una máquina virtual con KVM con la ISO de instalación descargada anteriormente (en este caso la versión netinst de Debian stable), para comenzar instalación y conectarnos por VNC:

```
kvm -m 512 -cdrom debian-6.0.5-amd64-netinst.iso -drive
file=debian6.0.5.img,if=scsi,index=0 -boot d -net nic -net user -
nographic -vnc :0
```

Durante la instalación creamos una sola partición que ocupe todo el disco virtual y la asociamos a /. No creamos partición de swap.

Si fuera necesario volver a iniciar el sistema, pero en este caso, desde el fichero con el disco duro virtual, ejecutaríamos la siguiente instrucción:

```
kvm -m 512 -drive file=debian6.0.5.img,if=scsi,index=0,boot=on -
boot c -net nic -net user -nographic -vnc :0
```

Durante la instalación del sistema, es imprescindible instalar el servidor ssh, ya que será el método utilizado por defecto para acceder a las instancias del cloud una vez que estén iniciadas.

Una vez terminada la instalación del sistema, se copia el fichero con la imagen del nuevo sistema al equipo en el que se ejecuta glance y se agrega a las imágenes disponibles de la siguiente manera:

```
glance add name="Debian Squeeze 6.0.5" is_public=true
container_format=ovf disk_format=raw < Debian6.0.5.img
```

Creación de imágenes Windows

Como en cualquier otro caso, creamos un fichero para la imagen del sistema (en este caso se ha optado por formato qcow2):

```
kvm-img create -f qcow2 win7.qcow2 10G
```

En el caso de instalar un sistema Windows 7 el tamaño mínimo del disco debe ser de 7 GB para poder hacer la instalación, aunque es recomendable al menos 10.

Las instancias de OpenStack con KVM se inician utilizando dispositivos virtuales virtio para los discos duros y las interfaces de red. Los sistemas GNU/Linux los reconocen sin problemas ya que los controladores de dispositivos virtio están incluidos en el kérnel linux, pero los sistemas Windows no incluyen soporte nativo para esos controladores, por lo que debemos crear la imagen con los controladores virtio incorporados. Para ello, nos descargamos previamente los controladores virtio en formato vfd y arrancamos la máquina virtual con la imagen de disco duro, los controladores virtio en una *disquetera* y la iso del instalador del sistema windows:

```
kvm -m 1024 -cdrom Windows7.iso -drive
file=win7.img,if=virtio,boot=on -fda virtio-win-1.1.16.vfd -boot d
-nographic -vnc :0
```

Comenzamos la instalación del sistema Windows a través de la conexión VNC y cuando tengamos que seleccionar el disco duro para el sistema nos encontraremos con que no existe ninguno, entonces deberemos de hacer click en "Cargar controladores" en el botón de la izquierda y cargar los controladores de dispositivos virtio desde A:\amd64\win7.

Una vez acabada la instalación y reiniciado el sistema windows, es necesario activar RDP:

- Panel de Control > Sistema > Configuración avanzada del sistema > Acceso Remoto
- Seleccionar "Permitir que las conexiones desde equipos que se ejecuten en cualquier version de escritorio remoto"
- Por defecto sólo el administrador tiene acceso así que después haremos click en en "Seleccionar usuarios remotos" y agregaremos el nombre de los usuarios con el que se van a querer conectar desde las otras máquinas.

Si fuera necesario realizar algún cambio después de la instalación, arrancaríamos la máquina virtual de la siguiente manera:

```
kvm -m 1024 -drive file=win7.img,if=virtio,index=0,boot=on -boot c
-net nic -net user -nographic -vnc :0
```

Por último, transferimos la imagen del disco con el sistema windows al equipo en el que se encuentra Glance y la añadimos de la misma forma que las otras imágenes:

```
glance add name="Windows 7 Profesional 64bits" is_public=true
container_format=bare disk_format=raw < win7.img
```

Amazon Machine Images

Amazon desarrolló para su sistema EC2 un formato propio de imágenes, del que existen un gran número de imágenes ya realizadas y que puede interesarnos utilizar en algunos casos concretos. Amazon EC2 utiliza Xen como sistema de virtualización, que tiene la peculiaridad de que la máquina virtual que se levanta no tiene kérnel (ni initrd), sino que es el dominio principal (el equipo anfitrión) el que debe seleccionar un kérnel (y en su caso un initrd) y lanzar una máquina virtual pasándole el núcleo, esto hace que las imágenes de Amazon EC2 se compongan realmente de 3 imágenes: Una imagen para el kérnel, una imagen para el initrd y una imagen de disco con el sistema.

Los pasos que hay que realizar para agregar una Imagen de Amazon a OpenStack son los siguientes:

- Descargaremos el archivo que incluye las images ami, ari y aki y procederemos a subirlas a glance.

  ```
  wget https://launchpad.net/cirros/trunk/0.3.0/+download/
  cirros-0.3.0-x86_64-uec.tar.gz
  ```

- Descomprimimos el archivo que contiene 3 ficheros:

  ```
  tar xvzf cirros-0.3.0-x86_64-uec.tar.gz
  ```

  cirros-0.3.0-x86_64-blank.img, cirros-0.3.0-x86_64-initrd, cirros-0.3.0-x86_64-vmlinuz

- Subimos las imágenes a glance, teniendo en cuenta que la imagen con el sistema debe ser la última en subirse, ya que requiere que estén previamente disponibles en glance la imagen con el kérnel y la imagen con el initrd.

  ```
  glance add disk_format=aki container_format=aki
  name="cirros-0.3.0-x86_64-vmlinuz" < cirros-0.3.0-x86_64-vmlinuz
  ```

  ```
  glance add disk_format=ari container_format=ari
  name="cirros-0.3.0-x86_64-initrd" < cirros-0.3.0-x86_64-initrd
  ```

  ```
  glance add disk_format=ami container_format=ami name="cirros-ami-
  img 64bits - javi" kernel_id=cdbd8865-a088-4bb6-ada2-f3bca6b13820
  ramdisk_id=48fadc60-97a2-4250-bbd2-69ca298a94b9 < cirros-0.3.0-
  x86_64-blank.img
  ```

  Donde los identificadores del kérnel y el initrd son los que ha asociado en los pasos anteriores el propio Glance.

Obviamente, las imágenes del kérnel e initrd pueden utilizarse para diferentes imágenes de disco, siempre y cuando sean versiones adecuadas para el sistema que se va a utilizar posteriormente.

6. Networking

Para seguir esta guía se necesita:

7. Gestión de Volúmenes

`Nova-volume` es el servicio de OpenStack que permite proporcionar almacenamiento extra a las instancias, de forma similar al servicio *Elastic Block Storage (EBS)* de *Amazon EC2* pero con una implementación distinta.

Nova-volume es una solución iSCSI, proporciona almacenamiento a nivel de bloque utilizando el Gestor de Volúmenes Lógico (LVM, Logical Volume Manager) de Linux. Al tratarse de almacenamiento a nivel de bloque, un volumen solo puede conectarse con una instancia en un momento determinado, no se trata de almacenamiento compartido como el proporcionado por sistemas de ficheros como NFS ó GlusterFS.

El servicio nova-volume presenta volúmenes lógicos LVM a los nodos de computación que ejecutan las instancias a través del protocolo iSCSI (Internet SCSI, SCSI sobre TCP/IP). En esta arquitectura aparecen dos componentes principales:

- LVM (`lvm2`), que utiliza un grupo de volúmenes denominado `nova-volumes`. El nombre del volumen es configurable a través del fichero de configuración `nova.conf`
- `open-iscsi`, la implementación de iSCSI que gestiona las sesiones iSCSI en los nodos de computación.

Los pasos que se producen desde la creación de un volumen hasta su conexión son:

1. Se crea el volumen a través del comando **nova volume-create**, el cual crea un volumen lógico (LV) sobre el grupo de volúmenes (VG) "nova-volumes".
2. El volumen se conecta a una instancia a través del comando **nova volume-attach**, el cual crea un identificador IQN (iSCSI Qualified Name) único que se presentará al nodo de computación.
3. El nodo de computación que ejecuta la instancia a la que queremos conectar el volumen tiene ahora una sesión iSCSI activa y un nuevo dispositivo de almacenamiento local (normalmente un disco accesible a través de `/dev/sdX`).
4. `libvirt` utiliza dicho almacenamiento local como almacenamiento para la instancia, la instancia obtiene dicho almacenamiento a través de un nuevo disco, normalmente `/dev/vdX`

Para configurar este servicio, nuestra infraestructura debe tener un nodo controlador del Cloud ejecutando los servicios `nova-api`, `nova-scheduler`, `nova-objectstore`, `nova-network` y `nova-volume`. También necesitaremos uno o más nodos ejecutando el servicio `nova-compute`.

El nodo que ejecuta el servicio nova-volume, en nuestro caso el nodo controlador aunque puede ser otro, debe tener una partición de unos 60 GiB o más de espacio de almacenamiento etiquetada del tipo "Linux LVM" (código `0x8e`).

La topología de red para OpenStack, FlatDHCP en nuestro caso, debe estar configurada, así como la conectividad entre los nodos a través de TCP/IP.

Sobre iSCSI

`iSCSI, Internet Small Computer System Interface` es un estándar de almacenamiento en red basado en TCP/IP, básicamente describe cómo transportar comandos SCSI sobre redes IP. El protocolo permite que clientes, *initiators* tal como los denomina la terminología iSCSI, puedan enviar comandos SCSI (CDBs) a dispositivos de almacenamiento de servidores remotos (targets).

iSCSI es un protocolo de redes de almacenamiento SAN (Storage Area Network), una SAN permite a una organización consolidar el almacenamiento de su centro de datos en arrays o cabinas de discos que más tarde se presentan a los servidores que ofrecen servicios (web, bases de datos, ...). Dichos servidores obtienen el almacenamiento como si fuera local, iSCSI cuenta con las ventajas de ser más económico y de poder reutilizar la infraestructura de red, frente a otros protocolos más rápidos pero que necesitan de un hardware y cableado específico como Fibre Channel o Infiniband.

iSCSI es un protocolo de nivel de aplicación que trabaja sobre TCP, normalmente los puertos 860 ó 3260. Básicamente, iSCSI permite a dos hosts negociar sobre ciertos parámetros para poder después intercambiar comandos SCSI utilizando una red IP. De esta forma iSCSI es capaz de tomar un protocolo de alto rendimiento muy utilizado en buses de almacenamiento local y emularlo sobre redes de área extensa creando una red de almacenamiento (SAN). De forma diferente a otros protocolos utilizados en redes de almacenamiento, iSCSI no necesita de cableado especial ya que puede utilizar la infraestructura de red existente (interfaces de red, cableado, switches, etc.), por lo que a menudo se ve como una alternativa más económica a Fibre Channel (exceptuando FCoE, Fibre Channel sobre Ethernet) o Infiniband. Sin embargo, el rendimiento de iSCSI en un despliegue SAN se puede ver reducido en gran medida si no se configura en una red o subred separada (LAN ó VLAN).

Aunque iSCSI permite comunicar cualquier tipo de dispositivos, casi siempre es utilizado por los administradores de sistemas para la conexión de discos. Permitiendo a hosts que ofrecen servicios (web, FTP, bases de datos, servicios de virtualización, ...), poder acceder a volúmenes de discos situados en arrays de almacenamiento. Una SAN iSCSI se diseña normalmente con dos objetivos en mente:

- **Consolidación de almacenamiento.** Consiste en trasladar los diferentes recursos de almacenamiento a un lugar centralizado y más eficiente. Un entorno SAN proporciona un mayor grado de flexibilidad en el almacenamiento ya que es posible localizar nuevos volúmenes de discos para cualquier servidor en cualquier momento, sin tener que realizar ningún cambio ni en el cableado ni en la configuración hardware.

- **Recuperación de desastres.** Un entorno SAN permite un mayor grado de flexibilidad, ya que los recursos de almacenamiento pueden replicarse y migrarse (mirroring) entre localizaciones (incluso alejadas geográficamente) muy fácilmente.

iSCSI viene definido en los RFCs 3720 y 3783.

Conceptos importantes

Entre los conceptos más importantes a la hora de trabajar con iSCSI, destacamos los siguientes:

Conceptos iSCSI

Initiator

Un initiator (iniciador) es un cliente iSCSI. Un initiator en iSCSI cumple el mismo propósito que un adaptador SCSI hardware,exceptuando el cableado de los dispositivos. Un initiator SCSI se limita en enviar comandos SCSI a través de la red IP. Los initiators pueden ser de dos tipos:

- Software. Utilizan código para implementar el protocolo. Normalmente en drivers de dispositivo a nivel de kernel utilizando toda la pila de red del sistema operativo.

  Estos initiators están disponibles para los sistemas operativos más comunes y es el método más utilizado a la hora de desplegar soluciones iSCSI.

- Hardware. Utilizan un hardware dedicado, combinado normalmente con un determinado software (firmware) para la implementación iSCSI. Este tipo de initiators reducen en gran medida la sobrecarga de la CPU en el procesamiento del protocolo TCP y de las interrupciones Ethernet, por lo que aumentan el rendimiento global de los servidores a la hora de utilizar iSCSI.

  Un HBA (iSCSI Host Bus Adapter) es una tarjeta que implementa un initiator por hardware. Un HBA empaqueta en la misma tarjeta una interfaz Ethernet a Gigabit (o 10 Gigabits), una implementación TCP/IP y un adaptador SCSI, que es como se presenta al sistema operativo. Optativamente pueden incluir una ROM que permita al sistema arrancar desde una SAN.

Target

La especificación iSCSI define target como cada uno de los recursos de almacenamiento que se localizan en un servidor.

Un target iSCSI es un dispositivo de almacenamiento conectado a la red, pero también pueden serlo los dipositivos de almacenamiento que un sistema operativo ofrezca a través de iSCSI. De la misma forma que un sistema operativo puede implementar un initiator por software, también puede implementar un target. Esta solución está implementada en los sistemas operativos más comunes como Linux, BSD, Solaris o Windows Server.

También existen ciertas distribuciones como FreeNAS, Openfiler u OpenMediaVault especializadas en ofrecer soporte iSCSI.

LUN

En terminología SCSI, un LUN (Logical Unit Number) es una dirección lógica. Un LUN representa un dispositivo SCSI direccionable individualmente, que es parte de un target (de un dispositivo físico SCSI). En un entorno iSCSI un LUN representa una unidad de disco. Cuando se inicia la negociacion initiator/target, se establece la conectividad hacia un LUN, el resultado es una conexión iSCSI que simula una conexión con un disco duro. Los initiators tratan los LUN iSCSI de la misma forma que un disco SCSI físico o un disco IDE. Esto permite a los sistemas iSCSI formatear y gestionar sistemas de ficheros directamente sobre LUN iSCSI, en vez de montar de forma remota directorios tal como lo hace NFS o CIFS.

En despliegues más grandes, los LUNs normalmente representan "rodajas" (slices) de arrays de discos RAID, reservadas de forma individual a los servidores cliente.

iSCSI no impone ningún tipo de regla o restricción sobre la compartición de un LUN por varios servidores, el acceso concurente por parte de varios servidores a un mismo sistema de ficheros es una tarea del sistema operativo y depende principalmente del tipo de acceso soportado por el sistema de ficheros.

IQN

IQN, iSCSI Qualified Name.

Hay tres formas de especificar un nombre para los targets y los initiators en un entorno iSCSI: IQN (iSCSI Qualified Name), EUI (Extended Unique Identifier) y NAA (T11 Network Address Authority).

IQN se describe en los RFC 3720 y 3721 y es la más utilizada, consiste básicamente en asignar a cada target e initiator una cadena que contiene los siguientes campos:

- El literal `iqn`.
- La fecha en la que se creó el dominio o se creó el target/initiator. Se especifica con el formato: `yyyy-mm`
- Nombre del dominio DNS con los componentes en orden inverso.
- De forma optativa el carácter ':' usado como prefijo para una cadena de texto con el nombre asignado al almacenamiento.

Los siguientes ejemplos se pueden obtener del RFC 3720:

```
                  Naming     String defined by
     Type  Date    Auth      "example.com" naming authority
+--++-----+ +---------+ +-----------------------------+
|  ||     | |         | |                             |

iqn.1992-01.com.example:storage:diskarrays-sn-a8675309
iqn.1992-01.com.example
iqn.1992-01.com.example:storage.tape1.sys1.xyz
iqn.1992-01.com.example:storage.disk2.sys1.xyz[3]
```

Implementaciones iSCSI en GNU/Linux

Desde el punto de vista del initiator, del cliente iSCSI que inicia las conexiones al target o servidor, hay una clara solución dominante y es la proporcionada por el proyecto open-iscsi. Soportada actualmente por las principales distribuciones está ampliamente aceptada como el referente en cuanto a implementaciones de initiators iSCSI en el mundo GNU/Linux.

En cuanto a targets, las cosas son un poco más complicadas, al menos cuatro implementaciones FLOSS (Free/Libre Open Source Software) están disponibles para el almacenamiento sobre TCP/IP:

- **IET**, the iSCSI Enterprise Target, es una implementación en espacio kernel disponible como módulo pero no incluida aún en los fuentes oficiales del kernel de Linux.

  IET tiene su origen en un fork GPL de la implementación iSCSI de la empresa holandesa Ardis Technologies y es usada ampliamente por vendedores de soluciones iSCSI de bajo coste. Varias distribuciones populares, incluyendo Debian, Ubuntu o SUSE, incluyen IET en sus repositorios oficiales, en Ubuntu es incluso el target por defecto (paquete `iscsitarget`). Su desarrollo continúa de forma activa siendo Scott Walker y Arne Redlich los principales desarrolladores del proyecto.

- **STGT**, the Linux SCSI target framework. Con STGT, el ex-desarrollador del proyecto IET, Fujita Tomonori, intentó escribir un reemplazo multiprotocolo y en espacio de usuario de IET.

  Solo una porción muy pequeña del código de STGT se ejecuta en espacio de kernel y fue incluida en la versión 2.6.20, el resto del target se ejecuta en espacio de usuario. Red Hat adoptó rápidamente STGT como su target iSCSI por defecto durante el lanzamiento de Red Hat Enterprise Linux 5 (RHEL 5) y continúa actualmente su soporte en RHEL 6. SUSE también adoptó esta implementación para su SUSE Linux Enterprise Server (SLES) 11, oras distribuciones como Debian o Ubuntu también lo incorporan en sus repositorios oficiales. Sin embargo, el activo desarrollo de STGT parece que se ha detenido y actualmente el proyecto se encuentra en un modo de solo mantenimiento.

- **SCST**, Generic SCSI Target Subsystem for Linux. SCST, mantenido principalmente por Vladislav Bolkhovitin y Bart van Assche es otro fork de IET con la meta principal de solucionar "todos los problemas, casos límite y violaciones del estándar que IET tiene", tal como indican en su página web. SCST tiene un gran número de devotos seguidores y usuarios, que alaban los beneficios del mayor rendimiento de SCST en comparación con el resto de implementaciones de targets.

  Contrariamente a STGT, pero de la misma forma que IET, SCST realiza gran parte de su trabajo en espacio de kernel, y sus desarrolladores han solicitado repetidamente su inclusión en la rama principal del kernel de Linux. De momento y hasta ahora, dichos esfuerzos no han tenido recompensa, parcialmente por la llegada de un nuevo cuarto target: LIO.

- **LIO**, linux-iscsi.org, toma su nombre del dominio al que el proyecto pertenece, su principal desarrollador es Nicholas Bellinger. LIO es una implementación genérica de target que se ejecuta en espacio de kernel, de la cual, iSCSI es solamente uno de sus muchos frontends. Es el único que no tiene nada que ver con ninguno de los otros tres y éste no es únicamente su rasgo más destacado. LIO utiliza una aproximación muy inusual basada en ConfigFS para su configuración, la cual produjo varias polémicas en las listas de desarrollo del kernel. A pesar de esto, a principio del 2011, LIO derrotó a SCST en la batalla de ser el sustituto de STGT como subsistema de target preferido en la rama principal del kernel de Linux. Por esta razón muchas distribuciones ya han incluido a LIO en sus repositorios, como Ubuntu, pero otras aún no.

Cualquiera de estas cuatro implementaciones sirven perfectamente para el cometido de construir una solución de almacenamiento estable, simple y completamente software libre basada en iSCSI.

Instalación y configuración de nova-volume

Para la instalación y configuración del servicio seguiremos los siguientes pasos:

1. Instalaremos todo el software necesario en el controlador:

```
root@jupiter:~# apt-get install lvm2 nova-volume
```

2. Configuramos LVM con un grupo de volúmenes denominado `nova-volumes`, para ello:

   a. Creamos una partición (de unos 60 GiB o más) etiquetada del tipo "Linux LVM" (`0x8e`), supongamos que sea la partición `/dev/sda4`.

   b. Creamos el grupo de volúmenes (VG):

```
root@jupiter:~# pvcreate /dev/sda4
root@jupiter:~# vgcreate nova-volumes /dev/sda4
```

3. Instalación y configuración de iSCSI: recordamos que todos los nodos de computación actúan como *iSCSI initiators* mientras que el nodo controlador actúa como *iSCSI target*. Esto implica que los nodos de computación deben poder establecer conexiones con el controlador en el puerto TCP `3260`.

   Instalamos el software iSCSI en todos los nodos de computación:

```
root@io:~# apt-get install open-iscsi
```

   Como paso optativo podemos modificar el nombre del initiator, para ello modificamos el fichero `/etc/iscsi/initiatorname.iscsi` dejando el siguiente contenido:

```
## DO NOT EDIT OR REMOVE THIS FILE!
## If you remove this file, the iSCSI daemon will not start.
## If you change the InitiatorName, existing access control lists
## may reject this initiator.  The InitiatorName must be unique
## for each iSCSI initiator.  Do NOT duplicate iSCSI InitiatorNames.
InitiatorName=iqn.2012-08.net.iescierva:initiator:io
```

   Tan solo hay que cambiar la última parte con el nombre del nodo, tenemos que recordar que el nombre del initiator iSCSI debe ser único en toda nuestra instalación.

4. Instalación y configuración iSCSI: instalamos el target en el controlador, como implementación elegimos STGT (Linux SCSI target framework). La implementación que aparece en la documentación de OpenStack, concretamente en el manual de administración es IET (the iSCSI Enterprise Target), pero el paquete de instalación `nova-volume` lleva a STGT (paquete `tgt`) como dependencia. Así que optamos por ésta última. El paquete se habrá instalado al instalar nova-volume, si no, lo hacemos a través de este comando:

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

Warning

La documentación oficial de OpenStack instala el paquete `iscsitarget` y su módulo del kernel correspondiente, `iscsitarget-dkms`. Esto es un error ya que deja el sistema con dos implementaciones distintas de targets iSCSI, con el correspondiente conflicto de configuración, puertos, etc. Solo hay que instalar uno, por lo que nos decantamos por TGT.

¡¡No hay que instalar ni el paquete iscsitarget ni el paquete iscsitarget-dkms!!

Reiniciamos el servicio en el controlador:

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

Iniciamos también los servicios iSCSI en los nodos de computación:

```
root@io:~# service open-iscsi restart
```

5. Configuramos el servicio nova-volume modificando las siguientes directivas del fichero `/etc/nova/nova.conf` en el **controlador**:

```
volume_group=nova-volumes
volume_name_template=volume-%08x
iscsi_helper=tgtadm
iscsi_ip_prefix=172.20.251
iscsi_ip_address=172.20.251.190
iscsi_target_prefix=iqn.2012-08.net.iescierva:jupiter:
```

6. Iniciamos el servicio:

```
root@jupiter:~# service nova-volume stop
```

```
root@jupiter:~# service nova-volume start
```

Comprobamos que el servicio está activo a través del comando:

```
root@jupiter:~# nova-manage service list
Binary           Host                                 Zone
 Status     State Updated_At
nova-scheduler   jupiter                              nova
 enabled    :-)   2012-08-06 08:13:10
nova-consoleauth jupiter                              nova
 enabled    :-)   2012-08-06 08:13:09
nova-cert        jupiter                              nova
 enabled    :-)   2012-08-06 08:13:10
nova-compute     jupiter                              nova
 enabled    :-)   2012-08-06 08:13:12
nova-volume      jupiter                              nova
 enabled    :-)   2012-08-06 08:13:11
nova-network     jupiter                              nova
 enabled    :-)   2012-08-06 08:13:13
nova-compute     io                                   nova
 enabled    :-)   2012-08-06 08:13:13
```

7. Creamos y conectamos un volumen.

   Para crear un volumen necesitamos asignarle un nombre y un tamaño en GiB, podemos hacerlo a través del siguiente comando:

```
root@jupiter:~# nova volume-create --display_name volumen00 5
```

   Podemos ver el volumen recién creado y su estado a través del comando:

```
root@jupiter:~# nova volume-list
+----+-----------+--------------+------+-------------+-------------+
| ID |   Status  | Display Name | Size | Volume Type | Attached to |
+----+-----------+--------------+------+-------------+-------------+
| 1  | available | volumen00    | 5    | None        |             |
+----+-----------+--------------+------+-------------+-------------+
```

   Una vez que el estado es 'disponible' (available), el volumen ya puede conectarse a una instancia. Al crear el volumen, podemos ver que se ha creado un nuevo volumen lógico, en este caso sobre `/dev/nova-volumes/volume-00000001`:

```
root@jupiter:~# lvs
  LV              VG           Attr   LSize   Origin Snap%  Move Log Copy%
 Convert
  volume-00000001 nova-volumes -wi-ao   5,00g
root@jupiter:~# lvdisplay
  --- Logical volume ---
  LV Name                /dev/nova-volumes/volume-00000001
  VG Name                nova-volumes
  LV UUID                MHDJ7R-eeZP-dfCm-Tt81-gacq-4EV7-ZddL8s
  LV Write Access        read/write
  LV Status              available
  # open                 1
  LV Size                5,00 GiB
  Current LE             1280
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0
```

   Y que a nivel de iSCSI, el volumen también está disponible:

```
root@jupiter:~# tgtadm --lld iscsi --op show --mode target
Target 1: iqn.2012-08.net.iescierva:jupiter:volume-00000001
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
    LUN information:
        LUN: 0
            Type: controller
            SCSI ID: IET     00010000
            SCSI SN: beaf10
            Size: 0 MB, Block size: 1
            Online: Yes
```

```
            Removable media: No
            Readonly: No
    Backing store type: null
    Backing store path: None
            Backing store flags:
       LUN: 1
            Type: disk
            SCSI ID: IET     00010001
            SCSI SN: beaf11
            Size: 5369 MB, Block size: 512
            Online: Yes
            Removable media: No
            Readonly: No
            Backing store type: rdwr
            Backing store path: /dev/nova-volumes/volume-00000001
            Backing store flags:
    Account information:
    ACL information:
        ALL
```

Podemos conectar el volumen a una instancia a través del subcomando **volume-attach** del comando **nova**, para ello necesitamos:

- El UID de la instancia, lo obtenemos a partir de los comandos **nova list** y **nova show**, este últimos solo si queremos obtener más detalles de la instancia y asegurarnos de ella.

- El identificador del volumen, lo obtenemos a través del comando **nova volume-list**.

- El dispositivo que verá la instancia que se ejecuta en el nodo de computación. Normalmente un dispositivo de la forma `/dev/vdX`. KVM nombra a los dispositivos de esta forma, distinguiendo los discos de las máquinas virtuales (invitados), `/dev/vdX`, de los dispositivos del anfitrión, `/dev/sdX`.

El comando queda así:

```
root@jupiter:~# nova volume-attach de701111-5b42-456a-b3ea-1609d49ebc2f 1 /
dev/vdb
```

Tras la ejecución podemos ver comprobar varias cosas:

- El estado del volumen ha cambiado:

```
root@jupiter:~# nova volume-list
+----+--------+--------------+------+-------------
+--------------------------------------+
| ID | Status | Display Name | Size | Volume Type |             Attached
 to              |
+----+--------+--------------+------+-------------
+--------------------------------------+
| 1  | in-use | volumen00    | 5    | None        | de701111-5b42-456a-
b3ea-1609d49ebc2f |
+----+--------+--------------+------+-------------
+--------------------------------------+
```

- Hay una sesión iSCSI iniciada en el nodo controlador:

```
root@jupiter:~# tgtadm --lld iscsi --op show --mode target
Target 1: iqn.2012-08.net.iescierva:jupiter:volume-00000001
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
        I_T nexus: 1
        Initiator: iqn.2012-08.net.iescierva:initiator:io
        Connection: 0
            IP Address: 172.20.251.191
    LUN information:
        LUN: 0
            Type: controller
            SCSI ID: IET     00010000
            SCSI SN: beaf10
            Size: 0 MB, Block size: 1
            Online: Yes
            Removable media: No
            Readonly: No
     Backing store type: null
     Backing store path: None
            Backing store flags:
        LUN: 1
            Type: disk
            SCSI ID: IET     00010001
            SCSI SN: beaf11
            Size: 5369 MB, Block size: 512
            Online: Yes
            Removable media: No
            Readonly: No
            Backing store type: rdwr
            Backing store path: /dev/nova-volumes/volume-00000001
            Backing store flags:
    Account information:
    ACL information:
        ALL
```

También podemos verlo en el nodo de computación, ya que tendrá una sesión iSCSI abierta y un nuevo disco disponible:

```
root@io:~# iscsiadm -m session
tcp: [1] 172.20.251.190:3260,1 iqn.2012-08.net.
iescierva:jupiter:volume-00000001
root@io:~# lsscsi
[4:2:0:0]    disk    SMC      SMC2108          2.12  /dev/sda
[7:0:0:0]    cd/dvd  KVM      vmDisk-CD        0.01  /dev/sr0
[8:0:0:0]    disk    KVM      vmDisk           0.01  /dev/sdb
[9:0:0:0]    storage IET      Controller       0001  -
[9:0:0:1]    disk    IET      VIRTUAL-DISK     0001  /dev/sdc
root@io:~# fdisk -l /dev/sdc

Disk /dev/sdc: 5368 MB, 5368709120 bytes
166 heads, 62 sectors/track, 1018 cylinders, total 10485760 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: 0x00000000

Disk /dev/sdc doesn't contain a valid partition table
```

El comando **lsscsi** muestra como tipo "disk IET" los dispositivos importados a través de iSCSI.

- A la máquina virtual, se le presenta un nuevo dispositivo:

```
root@server00:~# fdisk -l /dev/vdb

Disco /dev/vdb: 5368 MB, 5368709120 bytes
16 cabezas, 63 sectores/pista, 10402 cilindros, 10485760 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0x00000000

El disco /dev/vdb no contiene una tabla de particiones válida
```

... y podemos hacer uso de él a través de los siguientes comandos, una vez particionado el disco correctamente:

```
root@server00:~# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=256    agcount=4, agsize=327616
 blks
         =                       sectsz=512   attr=2, projid32bit=0
data     =                       bsize=4096   blocks=1310464, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =registro interno       bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =ninguno                extsz=4096   blocks=0, rtextents=0
root@server00:~# mkdir /doc
root@server00:~# mount /dev/vdb1 /doc
root@server00:~# df -h
S.ficheros     Tamaño Usado  Disp Uso% Montado en
/dev/vda1        5,0G  1,1G  3,7G  23% /
udev             112M  8,0K  112M   1% /dev
tmpfs             48M  252K   48M   1% /run
none             5,0M     0  5,0M   0% /run/lock
none             120M     0  120M   0% /run/shm
/dev/vdb1        5,0G   33M  5,0G   1% /doc
```

Note

Utilizando KVM como hipervisor, la máquina virtual no verá el disco a no ser que se reinicie. Se puede evitar esto si cargamos en la máquina virtual el módulo `acpiphp`. Este módulo proporciona el driver ACPI para conexión en caliente de dispositivos PCI. Podemos hacer el cambio permanente a través del comando:

```
root@server00:~# echo acpiphp >> /etc/modules
```

Comandos Nova para la gestión de volúmenes

Para la configuración y trabajo con volúmenes tendremos a nuestra disposición los siguientes comandos:

- **nova volume-create**

  Añade un nuevo volumen.

```
nova volume-create [--snapshot_id <snapshot_id>]
                   [--display_name <display_name>]
                   [--display_description <display_description>]
                   [--volume_type <volume_type>]
                   <size>
```

  Normalmente indicaremos un nombre y un tamaño en GiB. Ejemplos:

```
root@jupiter:~# nova volume-create --display_name volumen01 5
root@jupiter:~# nova volume-create --display_name volumen02 5
```

  El nombre no es necesario que sea único ya que internamente OpenStack le asigna otro identificador que sí lo es.

- **nova volume-attach**

  Conecta un volumen a una instancia (máquina virtual).

```
nova volume-attach <server> <volume> <device>
```

  Es necesario indicar:

  - La instancia a la que conectaremos el volumen, podemos obtener su ID a través del comando **nova list**.
  - El volumen a conectar. Hay que indicar su identificador, podemos obtenerlo a través del comando **nova volume-list.**
  - El dispositivo que verá la máquina virtual. Los volúmenes se exportan por iSCSI, por lo que el nodo de computación que contiene la máquina virtual, verá un nuevo disco con su fichero de dispositivo asociado de la forma `/dev/sdX`, normalmente `/dev/sdb`. La máquina virtual lo verá a través de KVM, a través de un fichero de dispositivo de la forma /dev/vdX, normalmente `/dev/vdb`. Hay que tener muy en cuenta a la hora de indicar este parámetro cuál es el siguiente libre en la máquina virtual. El comando no

funcionará si el dispositivo ya existe, si lo hará aunque asigne otra letra si el dispositivo que se pasa como parámetro no es el siguiente a asignar en la máquina virtual.

Ejemplo:

```
root@jupiter:~# nova volume-attach de701111-5b42-456a-b3ea-1609d49ebc2f 1 /
dev/vdc
```

Este comando conecta el volumen 1 con la instancia indicada por el UUID. En la instancia el volumen aparecerá como /dev/vdc, siempre y cuando vda y vdb estén ocupados y vdc libre.

- **nova volume-detach**

  Desconecta un volumen de una instancia (máquina virtual).

  ```
  nova volume-detach <server> <volume>
  ```

  Es necesario indicar:

  - La instancia a la que desconectaremos el volumen, podemos obtener su ID a través del comando **nova list**.
  - El volumen a desconectar. Hay que indicar su identificador, podemos obtenerlo a través del comando **nova volume-list.**

  Ejemplo:

  ```
  root@jupiter:~#  nova volume-detach de701111-5b42-456a-b3ea-1609d49ebc2f 1
  ```

- **nova volume-delete**

  Elimina un volumen.

  ```
  nova volume-delete <volume>
  ```

  Es necesario indicar:

  - El volumen a eliminar. Hay que indicar su identificador, podemos obtenerlo a través del comando **nova volume-list.**

  Ejemplos:

  ```
  root@jupiter:~#  nova volume-delete 1
  ```

  El borrado es una operación costosa ya que, por razones de seguridad, nova-volume sobrescribe con ceros todos los bloques de datos físicos que conforman el volumen. Podemos acelerar el borrado sobrescribiendo únicamente el primer gigabyte,

eliminando el MBR y el principio del sistema de ficheros, tal como se explica en: https://dijks.wordpress.com Basta realizar el siguiente cambio:

- Editamos el fichero `/usr/lib/python2.7/dist-packages/nova/volume/driver.py` modificando la línea 131 con este cambio:

```
      self._copy_volume('/dev/zero', self.local_path(volume), 1)
```

- Podemos automatizar esto a través del siguiente script:

```
#!/bin/bash
# https://dijks.wordpress.com/2012/07/09/how-to-speedup-removal-of-a-
volume/
# Function for easy string replacement
function sedeasy {
    sed -i "s/$(echo $1 | sed -e 's/\([[\/.*]\|\]\)/\\&/g')/$(echo $2 |
 sed -e 's/[\/&]/\\&/g')/g" $3
}

# Backup original file
cp /usr/lib/python2.7/dist-packages/nova/volume/driver.py\
/usr/lib/python2.7/dist-packages/nova/volume/driver.py.bak

# Change writing zero's to the whole file with only the first GB
sedeasy "self._copy_volume('/dev/zero', self.local_path(volume),
 size_in_g)"\
"self._copy_volume('/dev/zero', self.local_path(volume), 1)"\
/usr/lib/python2.7/dist-packages/nova/volume/driver.py
```

- **nova volume-list**

  Muestra un listado con todos los volúmenes e información sobre ellos.

```
usage: nova volume-list
```

- **nova volume-show**

  Muestra información detallada sobre un volumen.

```
usage: nova volume-show <volume>
```

  Es necesario indicar:

  - El volumen a mostrar la información. Hay que indicar su identificador, podemos obtenerlo a través del comando **nova volume-list.**

  Ejemplos:

```
root@jupiter:~#  nova volume-show 1
```

Resolución de problemas

Si por cualquier motivo, no se consigue la conexión entre los volúmenes y las instancias, podemos realizar ciertas comprobaciones para chequear que todo esté correctamente configurado. La fuente de información más importante son los ficheros de logs, concretamente:

- `/var/log/nova/nova-volume.log`, en el controlador.
- `/var/log/nova/nova-compute.log`, en los nodos de computación.

Los errores más comunes pueden ser, entre otros:

- *Error: "Stderr: 'iscsiadm: No portal found.".* Es un error de iSCSI, producido porque el nodo de computación no puede conectarse al target, normalmente por un error en la inicialización del subsistema iSCSI en el nodo controlador. También puede aparecer este error si se tienen instalados, por error, los dos paquetes que proporcionan una implementación target iSCSI: iscsitarget y tgt. Este último como dependencia del paquete nova-volume.

  Para solucionarlo nos tenemos que asegurar de que el subsistema iSCSI funciona correctamente. Nos tenemos que asegurar también de que solo uno de los paquetes anteriormente mencionados esté instalado.

  También se puede producir este error si todo está bien configurado pero el target del volumen en concreto no se está exportando. En ese caso debemos borrar las referencias al volumen de la tabla `block_device_mapping` de la base de datos de Nova.

- *Error: "Stderr: 'iscsiadm: Cannot resolve host...".*Este error se produce cuando el nodo de computación no puede encontrar el nombre del servidor que proporciona los servicios nova-volume.

  Para solucionarlo nos tenemos que asegurar del correcto funcionamiento de la resolución de nombres, bien vía DNS o bien vía `/etc/hosts`.

- *Error: "No route to host".* Este error puede estar causado por multitud de motivos, pero significa solo una cosa, el proceso *iscsid* (open-iscsi) no puede contactar con el servidor nova-volume (y por tanto con el target iSCSI). Desde el nodo de computación podemos comprobar la ejecución de este comando:

  ```
  root@io:~# telnet 172.20.254.190 3260
  ```

  Siendo `172.20.254.190` la dirección IP del nodo y `3260` el puerto usado en iSCSI. Si el comando finaliza con un timeout, hay un problema de conectividad, debemos entonces chequear cortafuegos, comprobar conectividad a través del comando **ping**, **traceroute**, etc. Nos puede ayudar también comprobar si una operación *discovery* de iSCSI desde los nodos de computación tiene éxito:

  ```
  root@io:~# iscsiadm --mode discovery --type sendtargets --portal 172.20.251.
  190
  ```

- *Pérdida de conexión entre nova-volume y nova-compute. Cómo recuperar a un estado limpio.*

  Las desconexiones de red, por desgracia, se producen. Desde el punto de vista iSCSI, una pérdida de conexión implica la extracción física de un disco en el servidor (el cliente iSCSI). Si la instancia estaba utilizando el volumen cuando se produjo la desconexión, nova-volumen no podrá desconectar el volumen (`detach`)

  Para solucionar este problema y que todo vuelva a la "normalidad" seguimos estos pasos:

  1. Desde el nodo de computación eliminamos todas las sesiones iSCSI activas (stalled), obtenemos el listado de todas ellas:

     ```
     root@calisto:~# iscsiadm -m session
     ```

     ... y eliminamos las problemáticas:

     ```
     root@calisto:~# iscsiadm -m session-r <ID>> -u
     ```

     ... siendo ID el identificador de la sesión a cerrar. Ejemplo: podemos cerrar la sesión iSCSI 10 de esta salida:

     ```
     tcp: [10] 172.20.251.190:3260,1 iqn.2012-08.net.
     iescierva:jupiter:volume-00000004
     ```

     A través de este comando:

     ```
     root@calisto:~# iscsiadm -m session -r 10 -u
     Logging out of session [sid: 10, target: iqn.2012-08.net.
     iescierva:jupiter:volume-00000004, portal: 172.20.251.190,3260]
     Logout of [sid: 10, target: iqn.2012-08.net.
     iescierva:jupiter:volume-00000004, portal: 172.20.251.190,3260]:
      successful
     ```

  2. A pesar de cerrar la sesión, desde el punto de vista de nova-volume, el volumen sigue "en uso" tal como se muestra en la salida de nova volume-list. Para delvolver el volumen al estado "disponible" ejecutamos las siguientes sentencias MySQL sobre la base de datos de Nova:

     a. Obtenemos el identificador del volumen y nos aseguramos de la información:

        ```
        mysql> select id, status, attach_status, instance_id, mountpoint from
         volumes;
        ```

     b. Reseteamos el volumen:

        ```
        mysql> update volumes set status="available", attach_status="detached",
         instance_id=NULL, mountpoint=NULL where id=5;
        ```

  3. Ya podemos volver a conectar el volumen.

- *Reinicio de nova-volume.*

  Si por cualquier motivo deseamos volver a restaurar el estado inicial de nova-volume debemos seguir los siguientes pasos:

  1. Nos aseguramos de que ninguna instancia esté usando ningún volumen. Podemos desconectar los volúmenes a través del comando **nova nova-detach**. Este paso no es obligatorio, pero sí recomendable.

  2. Detenemos el servicio iSCSI en los nodos de computación:

     ```
     root@io:~# service open-iscsi stop
     ```

  3. Detenemos el servicio nova-volume y el servicio iSCSI en el nodo controlador:

     ```
     root@jupiter:~# service nova-volume stop
     root@jupiter:~# service tgt stop
     ```

  4. No debe quedar ningún target en el nodo controlador (ni siquiera con el servicio `tgt` activo:

     Si quedara alguno, lo eliminamos:

     ```
     root@jupiter:~# tgtadm --lld iscsi --op show --mode target
     root@jupiter:~# tgtadm --lld iscsi --op delete --mode target --tid=ID
     ```

     ... siendo ID el identificador del target a eliminar.

  5. Eliminamos todos los volúmenes lógicos LVM creados:

     ```
     root@jupiter:~# lvremove /dev/nova-volumes/volume-00000001
     root@jupiter:~# lvremove /dev/nova-volumes/volume-00000002
     ...
     ```

  6. Eliminamos las siguientes entradas de la base de datos de Nova:

     ```
     root@jupiter:~# mysql-u root -p
     mysql> delete from iscsi_targets;
     mysql> delete from volume_metadata;
     mysql> delete from volume_type_extra_specs;
     mysql> delete from volume_types;
     mysql> delete from block_device_mapping;
     mysql> delete from volumes;
     ```

  7. Volvemos a iniciar el servicio iSCSI en los nodos de computación:

     ```
     root@io:~# service open-iscsi start
     ```

  8. Iniciamos el servicio iSCSI y el servicio nova-volume en el nodo controlador:

```
root@jupiter:~# service tgt start
root@jupiter:~# service nova-volume start
```

Controladores de Volúmenes (Volume Drivers)

El comportamiento por defecto de nova-volume se puede alterar usando diferentes drivers para nova-volume. Estos drivers ya están incluidos en el código base de Nova, tan solo hay que seleccionarlos a través de la directiva `volume_driver` del fichero `nova.conf`. El valor por defecto es el siguiente:

```
volume_driver=nova.volume.driver.ISCSIDriver
```

Pero Nova pone a disposición drivers para los siguientes backends de almacenamiento:

- Ceph Rados block device (RBD).

  Si se usa KVM/QEMU como hipervisor, se puede configurar Nova para que use RBD para la gestión de volúmenes. Tan solo hay que añadir las siguientes directivas en los nodos que ejecuten `nova-volume`:

  ```
  volume_driver=nova.volume.driver.RBDDriver
  rbd_pool=nova
  ```

- Nexenta.

  El Sistema Operativo NexentaStor es una plataforma software NAS/SAN para la construcción de arrays de almacenamiento en red rápido y confiable, al estar basado en el sistema operativo OpenSolaris y usar ZFS como sistema de ficheros nativo. NexentaStor puede actuar como nodo de almacenamiento en una infraestructura OpenStack y proporcionar volúmenes a nivel de bloque para las instancias en ejecución a través del protocolo iSCSI.

  El driver de Nexenta permite utilizar Nexenta SA para almacenar volúmenes de Nova. Cada volumen Nova está representado a través de un único `zvol` en un volumen predefinido de Nexenta. Para cada nuevo volumen, el driver crea un target iSCSI y un grupo de targets que se utilizarán para el acceso por las instancias.

  Para el uso de Nexenta por Nova, hay que configurar los siguientes parámetros:

  - `volume_driver=nova.volume.nexenta.volume.NexentaDriver`
  - `nexenta_host`: nombre del host o la dirección IP del servidor NexentaStor.
  - `nexenta_user` y `nexenta_password`: usename y password del usuario que cuente con todos los privilegios necesarios en el servidor, incluyendo acceso a la API REST.
  - `nexenta_volume`: nombre del volumen Nexenta a utilizar, por defecto se utilizará `nova`.

  Otros parámetros interesantes son los siguientes:

- `nexenta_target_prefix`: prefijo añadido al identificador de volumen para formar el nombre del target que Nexenta ofrecerá.
- `nexenta_target_group_prefix`: prefijo para el grupo de targets.
- `nexenta_blocksize`: tamaño del bloque de datos a utilizar. Ejemplo: 8K
- `nexenta_sparse`: valor booleano que indica si se usarán zvols libres para ahorrar espacio.

Los siguientes parámetros se pueden fijar en sus valores por defecto:

- `nexenta_rest_port`: puerto donde Nexenta escucha las peticiones REST.
- `nexenta_rest_protocol`: puede ser `http` o `https`, el valor por defecto es `auto`.
- `nexenta_iscsi_target_portal_port`: puerto donde Nexenta escucha las peticiones iSCSI.

- Xen Storage Manager

Iniciando una instancia desde un volumen

POR HACER.

8. Administración de OpenStack

Aunque gran parte de tareas de administración de OpenStack se pueden realizar de forma intuitiva desde la interfaz gráfica web Horizon(dashboard), aquí se presentan los comandos más usados para la administración desde la consola

Gestión de instancias

Una vez que se tenga alguna imagen dada de alta en nuestro sistema se podrán arrancar instancias de ésta. Estas instancias se ejecutarán en el o los nodos de computación que estén activos.

Podemos listar las imágenes que tengamos en glance:

```
$ glance index
```

```
root@neo:~# glance index
ID                                   Name                           Disk
 Format
Container Format     Size
------------------------------------ ------------------------------
-------------------- -------------------- --------------
86119171-9ed6-435c-9620-573e0b440e58 Windows 7 general              vdi
bare                    14463094784
2710174c-9388-4d55-8ece-ace0250817de Ubuntu 12.04 Server            qcow2
bare                      233177088
838d547b-4a6c-4177-9e6a-fb6695b67cd4 Cirros (test) 0.3.0 64 bits    qcow2
bare                        9761280
root@neo:~#
```

Como vemos tenemos actualmente tres imágenes disponibles para ser instanciadas. Una imagen de Windows 7 en formato vdi (Virtualbox) y dos imagenes en qcow2, una de Ubuntu 12.04 Server y otra Cirros.

Para instanciar una de las imágenes es necesario saber qué plantillas de ejecución (flavor) tenemos disponibles en nuestro sistema.

Si no las conocemos, listamos las plantillas disponibles en nuestro sistema:

```
$ nova manage flavor list
```

```
m1.medium: Memory: 4096MB, VCPUS: 2, Root: 40GB, Ephemeral: 0Gb, FlavorID: 3,
Swap: 0MB, RXTX Factor: 1.0
m1.large: Memory: 8192MB, VCPUS: 4, Root: 80GB, Ephemeral: 0Gb, FlavorID: 4,
Swap: 0MB, RXTX Factor: 1.0
m1.tiny: Memory: 512MB, VCPUS: 1, Root: 0GB, Ephemeral: 0Gb, FlavorID: 1,
 Swap:
0MB, RXTX Factor: 1.0
m1.xlarge: Memory: 16384MB, VCPUS: 8, Root: 160GB, Ephemeral: 0Gb, FlavorID:
 5,
Swap: 0MB, RXTX Factor: 1.0
```

```
m1.small: Memory: 2048MB, VCPUS: 1, Root: 20GB, Ephemeral: 0Gb, FlavorID: 2,
Swap: 0MB, RXTX Factor: 1.0
```

Vemos que por defecto el sistema viene con 5 plantillas de ejecución

En nuestro ejemplo vamos a instanciar la imagen de Ubuntu con la plantilla *m1.small*

```
$ nova boot --image "Ubuntu 12.04 Server" --flavor m1.small ubuntuserver1
```

```
+-------------------------------------+--------------------------------------+
|               Property              |                Value                 |
+-------------------------------------+--------------------------------------+
| OS-DCF:diskConfig                   | MANUAL                               |
| OS-EXT-SRV-ATTR:host                | None                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                 |
| OS-EXT-SRV-ATTR:instance_name       | instance-00000035                    |
| OS-EXT-STS:power_state              | 0                                    |
| OS-EXT-STS:task_state               | scheduling                           |
| OS-EXT-STS:vm_state                 | building                             |
| accessIPv4                          |                                      |
| accessIPv6                          |                                      |
| adminPass                           | hvpgLFGuF9ne                         |
| config_drive                        |                                      |
| created                             | 2012-09-19T12:07:42Z                 |
| flavor                              | m1.small                             |
| hostId                              |                                      |
| id                                  | f7cb19ef-06a4-433e-806c-ab0db55e30a2 |
| image                               | Ubuntu 12.04 Server                  |
| key_name                            |                                      |
| metadata                            | {}                                   |
| name                                | ubuntuserver1                        |
| progress                            | 0                                    |
| status                              | BUILD                                |
| tenant_id                           | a70373bf297a42eb9203645d84476c23     |
| updated                             | 2012-09-19T12:07:42Z                 |
| user_id                             | 4c249654613e41208b2d03169ef29580     |
+-------------------------------------+--------------------------------------+
```

Podemos ver que el sistema nos ha devuelto todos los datos de la instancia que se está construyendo.

Seguidamente se puede listar en qué estado está nuestras instancias con:

```
$ nova list
```

```
+--------------------------------------+---------------+--------
+------------------+
|                  ID                  |      Name     | Status |     Networks
|
+--------------------------------------+---------------+--------
+------------------+
| f7cb19ef-06a4-433e-806c-ab0db55e30a2 | ubuntuserver1 | BUILD  | private=10.
0.0.2 |
+--------------------------------------+---------------+--------
+------------------+
```

Como vemos la instancia está aún construyéndose

Si todo ha ido bien al cabo de un tiempo deberá aparecer la instancia como activa

```
$ nova list
+--------------------------------------+---------------+--------
+------------------+
|                  ID                  |      Name     | Status |     Networks
|
+--------------------------------------+---------------+--------
+------------------+
| f7cb19ef-06a4-433e-806c-ab0db55e30a2 | ubuntuserver1 | ACTIVE | private=10.
0.0.2 |
+--------------------------------------+---------------+--------
+------------------+
```

Una vez creadas varias instancias podemos ver su estado:

```
$ nova list
```

```
+--------------------------------------+---------------+--------
+------------------+
|                  ID                  |      Name     | Status |     Networks
|
+--------------------------------------+---------------+--------
+------------------+
| 9273c28b-7db9-41aa-916a-b4589420af58 | ubuntuserver4 | ACTIVE |
private=10.0.0.5 |
| cd5e6c35-9e62-49d7-815f-c3ea09540806 | ubuntuserver2 | ACTIVE |
private=10.0.0.3 |
| dbefde80-48e1-48b0-a8b0-84f2ddc7a1f7 | ubuntuserver3 | ACTIVE |
private=10.0.0.4 |
| f7cb19ef-06a4-433e-806c-ab0db55e30a2 | ubuntuserver1 | ACTIVE |
private=10.0.0.2 |
+--------------------------------------+---------------+--------
+------------------+
```

Las demás acciones que se pueden realizar con una instancia se listan a continuación:

- **Lanzar instancia en un nodo:**si se desea lanzar una instancia en un nodo en concreto se tiene que forzar con los flags *–hint force_hosts=*

```
$ nova boot --image "Ubuntu 12.04 Server" --flavor m1.small --hint
 force_hosts=trinity ubuntuserver5
```

```
+-------------------------------------
+------------------------------------------------------------+
|               Property               |                          Value
|
+-------------------------------------
+------------------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL
|
| OS-EXT-SRV-ATTR:host                 | trinity
|
```

```
| OS-EXT-SRV-ATTR:hypervisor_hostname | None
|
| OS-EXT-SRV-ATTR:instance_name       | instance-0000003c
|
| OS-EXT-STS:power_state              | 1
|
| OS-EXT-STS:task_state               | None
|
| OS-EXT-STS:vm_state                 | active
|
| accessIPv4                          |
|
| accessIPv6                          |
|
| config_drive                        |
|
| created                             | 2012-09-19T18:30:48Z
|
| flavor                              | m1.small
|
| hostId                              |
c4cfd12e74d001411d189ab076248cd268b2b7cfda5017cf67376682 |
| id                                  | d21be20f-3a03-4979-8d5c-6a32dab71a43
|
| image                               | Ubuntu 12.04 Server
|
| key_name                            |
|
| metadata                            | {}
|
| name                                | ubuntuserver5
|
| private network                     | 10.0.0.6
|
| progress                            | 0
|
| status                              | ACTIVE
|
| tenant_id                           | a70373bf297a42eb9203645d84476c23
|
| updated                             | 2012-09-19T18:30:29Z
|
| user_id                             | 4c249654613e41208b2d03169ef29580
|
+-------------------------------------
+------------------------------------------------------------+
```

Note

Atención la funcionalidad de forzar nodo de computación solo funciona con ciertos tipos de planificadores (scheduler). No funciona con*SimpleScheduler* pero si con *Filter Scheduler*

- **Pausar instancia:**

```
$  nova pause $id_de_instancia
```

- **Reanudar instancia pausada:**

```
$  nova unpause $id_de_instancia
```

- **Suspender una instancia:**

```
$  nova suspend $id_de_instancia
```

- **Reanudar una instancia suspendida:**

```
$  nova resume $id_de_instancia
```

- **Borrar una instancia (debe estar activa):**

```
$  nova delete $id_de_instancia
```

Gestión de plantillas

Las plantillas o flavors describen la CPU, memoria y tamaño de almacenamiento que van a poder ocupar las distintas instancias en nova.

Así cada instancia va a poder ocupar el máximo de estos recursos una vez levantada.

En nova se van a poder utilizar las plantillas que vienen por defecto y también se pueden crear nuevas plantillas en caso de que sea necesario.

Las siguientes acciones se pueden realizar con las plantillas:

- **Listar las distintas plantillas existentes:**

```
$nova-manage flavor list
```

```
m1.medium: Memory: 4096MB, VCPUS: 2, Root: 10GB, Ephemeral: 40Gb, FlavorID:
 3, Swap: 0MB, RXTX Factor: 1.0
m1.asir: Memory: 256MB, VCPUS: 1, Root: 5GB, Ephemeral: 0Gb, FlavorID: 6,
 Swap: 0MB, RXTX Factor: 1.0
m1.small: Memory: 2048MB, VCPUS: 1, Root: 10GB, Ephemeral: 20Gb, FlavorID:
 2, Swap: 0MB, RXTX Factor: 1.0
m1.large: Memory: 8192MB, VCPUS: 4, Root: 10GB, Ephemeral: 80Gb, FlavorID:
 4,Swap: 0MB, RXTX Factor: 1.0
m1.tiny: Memory: 512MB, VCPUS: 1, Root: 0GB, Ephemeral: 0Gb, FlavorID: 1,
 Swap: 0MB, RXTX Factor: 1.0
m1.xlarge: Memory: 16384MB, VCPUS: 8, Root: 10GB, Ephemeral: 160Gb,
 FlavorID: 5 Swap: 0MB, RXTX Factor: 1.0
```

- **Crear una nueva plantilla**

```
$nova-manage flavor create --name=m1.xxlarge
    --memory=16384 --cpu=16 --local_gb=20 --flavor=7 --swap=0 --rxtx_quota=0
    --rxtxcap=0
```

- **Borrar una plantilla existente**

```
$nova-manage flavor delete --name=m1.xxlarge
```

Gestión de usuarios

La gestión de usuarios para esta versión de Openstack se puede realizar de forma ágil desde la interfaz gráfica de Horizon. Sin embargo si se desea utilizar la interfaz de comandos deberemos utilizar **keystone** para la creación y administración de usuarios.

Atención: en versiones anteriores se podía crear usuarios a través de **nova-manage**, método a extinguir desde que existe keystone.

- Listar los usuarios existentes

```
$keystone user-list
```

```
+----------------------------------+---------+-------------------+--------+
|                id                | enabled |       email       |  name  |
+----------------------------------+---------+-------------------+--------+
| 05743001bbf14700bcdf2ecc43edbf9b | True    | alex@iescierva.net | admin  |
| 246ba4e3d81c4ae8bdde8ec5784e74d3 | True    | alex@iescierva.net | swift  |
| 291c58f7258747758d109ecee2eb4a69 | True    | alex@iescierva.net | glance |
| 404dafc53b364a7e9f1476aa98082966 | True    | alex@iescierva.net | nova   |
+----------------------------------+---------+-------------------+--------+
```

- Crear un usuario nuevo asociado a un proyecto (sabiendo el id de proyecto-tenant)

```
$ keystone user-create --name alumno
--tenant_id e7b1868b24a742318f1d73f612ecfe1d --pass curso12-13 --email
 correo@gmail.com
```

```
+----------
+-----------------------------------------------------------------------------------------
+
| Property |                                                                  Value
|
+----------
+-----------------------------------------------------------------------------------------
+
| email    | correo@gmail.com
|
| enabled  | True
|
| id       | d0bfed18cce3460892ae99ee812d8aa3
|
| name     | alumno
|
| password |
$6$rounds=40000$8Q0MqKruQVZXvYZN
$MeAsef3ihVpBWJrVUyBACF5TuiOwVSHtLU5HvkZBXrqXEmU0dvqjgNhfBc6eK6HZbatJ.
CQDgQvkUNVrIBAjY1
|
| tenantId | e7b1868b24a742318f1d73f612ecfe1d
|
+----------
+-----------------------------------------------------------------------------------------
+
$ keystone user-list
+----------------------------------+---------+-------------------+--------+
```

```
|                id                | enabled |       email        |  name  |
+----------------------------------+---------+--------------------+--------+
| 05743001bbf14700bcdf2ecc43edbf9b | True    | alex@iescierva.net | admin  |
| 246ba4e3d81c4ae8bdde8ec5784e74d3 | True    | alex@iescierva.net | swift  |
| 291c58f7258747758d109ecee2eb4a69 | True    | alex@iescierva.net | glance |
| 404dafc53b364a7e9f1476aa98082966 | True    | alex@iescierva.net | nova   |
| d0bfed18cce3460892ae99ee812d8aa3 | True    | correo@gmail.com   | alumno |
+----------------------------------+---------+--------------------+--------+
```

- Borrar un usuario sabiendo su id (ejemplo borrar usuario alumno)

```
$ keystone user-delete d0bfed18cce3460892ae99ee812d8aa3
$ keystone user-list
```

```
+----------------------------------+---------+--------------------+--------+
|                id                | enabled |       email        |  name  |
+----------------------------------+---------+--------------------+--------+
| 05743001bbf14700bcdf2ecc43edbf9b | True    | alex@iescierva.net | admin  |
| 246ba4e3d81c4ae8bdde8ec5784e74d3 | True    | alex@iescierva.net | swift  |
| 291c58f7258747758d109ecee2eb4a69 | True    | alex@iescierva.net | glance |
| 404dafc53b364a7e9f1476aa98082966 | True    | alex@iescierva.net | nova   |
+----------------------------------+---------+--------------------+--------+
```

- Obtener información detallada de un usuario

```
$ keystone user-get 05743001bbf14700bcdf2ecc43edbf9b
```

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| email    | alex@iescierva.net               |
| enabled  | True                             |
| id       | 05743001bbf14700bcdf2ecc43edbf9b |
| name     | admin                            |
| tenantId | None                             |
+----------+----------------------------------+
```

- Cambiar password de usuario

```
$keystone user-password-update --pass nuevopass
 05743001bbf14700bcdf2ecc43edbf9b
```

Gestión de proyectos

Al igual que con los usuarios la administración de proyectos se puede realizar de forma gráfica a través de Horizon. Para realizarla en modo comando se debe utilizar **keystone.**

- Listar proyectos (tenant)

```
$ keystone tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 634675c752634b53879656c81da70a83 | service | True    |
| e7b1868b24a742318f1d73f612ecfe1d | admin   | True    |
+----------------------------------+---------+---------+
```

- Crear un proyecto

```
$keystone tenant-create --name asir1 --description "1 curso asir"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | 1 curso asir                     |
| enabled     | True                             |
| id          | 8c6d42b3e339448fb3068ec512df7802 |
| name        | asir1                            |
+-------------+----------------------------------+
```

- Listar información detallada de un proyecto

```
$keystone tenant-get 8c6d42b3e339448fb3068ec512df7802
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | 1 curso asir                     |
| enabled     | True                             |
| id          | 8c6d42b3e339448fb3068ec512df7802 |
| name        | asir1                            |
+-------------+----------------------------------+
```

- Borrar un proyecto

```
$keystone tenant-delete 8c6d42b3e339448fb3068ec512df7802
```

Gestión de roles y asignación a usuarios

A los usuarios una vez asignados de forma predeterminada a un proyecto se les debe asignar uno o varios roles

La gestión de los roles se realiza a través de **keystone**

- **Listar roles existentes**

```
$keystone role-list
```

```
+----------------------------------+--------+
|                id                |  name  |
+----------------------------------+--------+
| 2a716d669ce349eb88e44049723e0cb7 | admin  |
| 622c78f77bbe4a298452783f25cb4635 | Member |
+----------------------------------+--------+
```

Por defecto al instalar el sistema se crearon dos roles: **admin** y **Member**.

- **Asignación de rol a usuario**

```
$keystone user-role-add
 --user=4ca77dc4d8a448219b1c6ae7a208ff47 --tenant=
e7b1868b24a742318f1d73f612ecfe1d
 --role=622c78f77bbe4a298452783f25cb4635
```

```
```

Se le ha asignado el rol **Member**al usuario **alumno** en el proyecto **asir1**

Para verificarlo se puede ver con:

```
$keystone role-list
--user=4ca77dc4d8a448219b1c6ae7a208ff47
--tenant=e7b1868b24a742318f1d73f612ecfe1d
```

```
+----------------------------------+--------+
|                id                |  name  |
+----------------------------------+--------+
| 622c78f77bbe4a298452783f25cb4635 | Member |
+----------------------------------+--------+
```

- **Desasignación de rol a usuario**

```
$keystone user-role-remove --user=4ca77dc4d8a448219b1c6ae7a208ff47
--tenant=e7b1868b24a742318f1d73f612ecfe1d
--role=622c78f77bbe4a298452783f25cb4635
```

```
$keystone role-list
--user=4ca77dc4d8a448219b1c6ae7a208ff47
--tenant=e7b1868b24a742318f1d73f612ecfe1d
```

```
+----+------+
| id | name |
+----+------+
+----+------+
```

Gestión de cuotas

La gestión de cuotas se realiza para cada proyecto concreto. Se pueden fijar cuotas para parámetros como CPU's(cores), memoria, volúmenes, instancias,numero de ip's, tamaño de disco, grupos de seguridad y ficheros injectados(injected files)

Se puede realizar de forma gráfica con Horizon y de modo comando a través de **nova-manage**

- **Obtener cuotas actuales de proyecto**

```
$nova-manage project quota
--project=e7b1868b24a742318f1d73f612ecfe1d
```

```
metadata_items: 128
volumes: 10
gigabytes: 1000
ram: 51200
security_group_rules: 20
instances: 10
security_groups: 10
injected_file_content_bytes: 10240
floating_ips: 10
injected_files: 5
cores: 20
```

- **Modificar parámetro de cuota de proyecto (ej:ampliar num de cores a 50)**

```
$nova-manage project quota
--project=e7b1868b24a742318f1d73f612ecfe1d --key=cores --value=50
metadata_items: 128
volumes: 40
gigabytes: 1000
ram: 51200
security_group_rules: 20
instances: 40
security_groups: 10
injected_file_content_bytes: 10240
floating_ips: 10
injected_files: 5
cores: 50
```

Monitorización de instancias y nodos

La monitorización de instancias y nodos se puede realizar a través de comandos de **nova**.

- **Listar instancias existentes en nova**

```
$nova list
```

```
+--------------------------------------+---------------+--------
+------------------+
|                  ID                  |      Name     | Status |
 Networks
|
+--------------------------------------+---------------+--------
+------------------+
| 0167eee1-9cbe-4bf2-a1b1-7d88b8259423 | UbuntuServer  | ACTIVE | private=
10.0.0.17 |
| 0cf6b815-d7d5-4a0d-af3f-c45468eab6b5 | UbuntuServer1 | ACTIVE | private=
10.0.0.9  |
| 11dada1d-90a5-46d5-b855-bbb447e48e65 | UbuntuServer2 | ACTIVE | private=
10.0.0.30 |
| 1a8bba78-65fd-4ac5-96bb-8538d39f4825 | UbuntuServer3 | ACTIVE | private=
10.0.0.32 |
| 1a8d14fb-39b1-4c9c-8501-fd81d0d59f01 | UbuntuServer4 | ACTIVE | private=
10.0.0.21 |
| 32d9aa7c-3e94-4d73-afd8-f441ab7a00b7 | UbuntuServer5 | ACTIVE | private=
10.0.0.24 |
| 3a4f4e81-48be-486a-82e8-ddec95aa6c40 | UbuntuServer6 | ACTIVE | private=
10.0.0.22 |
--------------------------------------------------------------------------------
```

- Mostrar información de una instancia determinada

```
$nova show
 0167eee1-9cbe-4bf2-a1b1-7d88b8259423
```

```
+-------------------------------------
+-----------------------------------------------------+
|               Property              |                          Value
|
```

```
+-------------------------------------
+-----------------------------------------------------------+
| OS-DCF:diskConfig                   | MANUAL
|
| OS-EXT-SRV-ATTR:host                | trinity
|
| OS-EXT-SRV-ATTR:hypervisor_hostname | None
|
| OS-EXT-SRV-ATTR:instance_name       | instance-0000004b
|
| OS-EXT-STS:power_state              | 1
|
| OS-EXT-STS:task_state               | None
|
| OS-EXT-STS:vm_state                 | active
|
| accessIPv4                          |
|
| accessIPv6                          |
|
| config_drive                        |
|
| created                             | 2012-09-24T09:06:40Z
|
| flavor                              | m1.medium
|
| hostId                              |
bcc6c02ce3eeaf89b869048bdf5039817b7c989be5e6ef7498dd09f9 |
| id                                  | 0167eee1-9cbe-4bf2-a1b1-7d88b8259423
|
| image                               | windows7plantilla
|
| key_name                            |
|
| metadata                            | {}
|
| name                                | UbuntuServer
|
| private network                     | 10.0.0.17
|
| progress                            | 0
|
| status                              | ACTIVE
|
| tenant_id                           | a70373bf297a42eb9203645d84476c23
|
| updated                             | 2012-09-24T09:06:38Z
|
| user_id                             | 4c249654613e41208b2d03169ef29580
|
+-------------------------------------
+-----------------------------------------------------------+
```

- Mostrar estado de los servicios de nova

```
$nova-manage service list
```

```
Binary           Host                                 Zone
 Status
State Updated_At
nova-cert        neo                                  nova
 enabled :-)   2012-10-28 17:26:22
nova-network     neo                                  nova
 enabled :-)   2012-10-28 17:26:22
nova-scheduler   neo                                  nova
 enabled :-)   2012-10-28 17:26:22
nova-consoleauth neo                                  nova
 enabled :-)   2012-10-28 17:26:22
nova-compute     trinity                              nova
 enabled XXX   2012-10-28 17:25:11
nova-compute     cifra                                nova
 enabled XXX   2012-09-28 15:12:00
nova-compute     tanque                               nova
 enabled XXX   2012-09-28 15:12:27
nova-compute     dozer                                nova
 enabled XXX   2012-09-28 15:12:50
```

- Mostrar estado de recursos en un determinado nodo

```
$nova-manage service describe_resource trinity
```

```
HOST                              PROJECT     cpu mem(mb)     hdd
trinity         (total)                         24   64414     824
trinity         (used_now)                      15    5995     101
trinity         (used_max)                      15   30720     300
trinity                  a70373bf297a42eb9203645d84476c23      15   30720
 300
```

Servicios VNC y VNCProxy

VNC Proxy

VNC Proxy es un componente de OpenStack que permite a los usuarios de los servicios de Computación acceder a sus instancias a través clientes VNC. En Essex y versiones posteriores viene soportado tanto por libvirt como por XenServer utilizando tanto clientes Java como clientes de websocket.

La conexión VNC a través de consola funciona como sigue:

1. El usuario se conecta a la API y recupera un acceso_url del tipo http://ip:port/?token=xyz

2. El usuario copia la URL en un navegador o como un parámetro de cliente

3. El navegador o cliente se conecta al proxy

4. El Proxy habla con nova-consoleauth para autorizar el token de usuario, entonces asocia el token al host privado y puerto de una instancia de VNC Server. En el nodo de computación se especifica la dirección que debe utilizar el proxy conectándose a través de la opción vncserver_proxy_client_address del fichero nova.conf. Así el vnc proxy trabaja como un puente entre la red pública y la red privada del host.

5. El proxy inicia una conexión con VNC Server que durará hasta que la sesión finalice.

El proxy también puede realizar la función de tunelizar el protocolo VNC a través de Websockets para que el cliente noVNC tenga una forma de hablar VNC. En general VNC-proxy desarrolla varias funciones

- Puente entre la red pública (dónde están los clientes) y la red privada (dónde están los servidores VNC)
- Facilita la authenticación vía token
- Provee una conexión uniforme con el hypervisor específico para homogeneizar la experiencia del usuario

Despliegue típico de VNC Proxy

Un despliegue típico de VNC proxy consta de:

- Un proceso **nova-consoleauth**. Normalmente se instala en el nodo controlador.
- Uno o varios servicios **nova-novncproxy**. Para dar soporte a los clientes novnc via navegador. En despliegues simples estos servicios pueden correr en el mismo host que **nova-api**. Estos servicios actuan de proxies entre la red pública y la red privada de los hosts de computación.
- Uno o varios servicios **nova-xvpvncproxy**. Estos servicios dan soporte a clientes Java especiales descritos más adelante. En despliegues simples estos servicios también pueden correr en el mismo host que **nova-api**. También actúan de proxies entre la red pública y la red privada de los hosts de computación.
- Uno o más hosts de computación. Estos hosts deberán tener correctamente configuradas las opciones para su correcto funcionamiento.

Obteniendo una URL de acceso

Nova ofrece la posibilidad de crear URL's de acceso (access_urls) a través de utilidades de consola

Se realiza a través de novaclient:

```
$nova get-vnc-console [server_id] [novnc|xvpnc]
```

Si se especifíca "novnc" se obtiene una URL válida para utilizarla en un navegador web. Por el contrario si se especifíca "xvpnc" se obtiene una URL para utilizarla en un cliente Java.

Opciones importantes de nova-compute

Para activar vncproxy en el cloud, se tiene que tener instalado el servicio de nova-consoleauth (normalmente en el controlador) y se deben configurar las siguientes opciones en nova.conf en los nodos de computación

- [no]vnc_enabled - Por defecto viene activado. Si se desactiva las instancias arrancaran sin soporte VNC.
- vncserver_listen - Por defecto viene configurado a la 127.0.0.1. Es la dirección a la que se asocian los vncservers y podría ser cambiada en entornos de producción a una dirección privada. Se aplica solo a entornos libvirt. Para despliegues multi-host libvirt debería

cambiarse por una dirección IP de gestión del nodo de computación. Dicha dirección IP estará en la misma red que los proxies

- vncserver_proxyclient_address - Por defecto viene configurado a la 127.0.0.1. Esta es la dirección del nodo de computación que nova le pasa al proxy cuando se va a realizar una conexión a las instancias de vncservers. Para despliegues libvirt multi-host debe cambiarse por una dirección IP de gestión del nodo de computación. Dicha dirección IP estará en la misma red que los proxies.

- novncproxy_base_url=[url base para los clientes] - Es la URL base a la que los clientes se conectarán. A esta URL se le añadira "?token=abc" para propósito de autenticación. Un despliegue típico tendrá la siguiente URL base "http://$SERVICE_HOST:6080/vnc_auto.html" dónde SERVICE_HOST es el nombre público del host

- xvpvncproxy_base_url=[url base para los clientes] - Es la URL base a la que los clientes se conectarán. A esta URL se le añadira "?token=abc" para propósito de autenticación. Un despliegue típico tendrá la siguiente URL base "http://$SERVICE_HOST:6081/console" dónde SERVICE_HOST es el nombre público del host.

Acceso a consolas VNC desde cliente Java

Para activar el soporte de Openstack a los clientes Java se dispone del servicios **nova-xvpvncproxy** que debe configurarse como sigue:

- xvpvncproxy_port=[port] - puerto de escucha (por defecto 6081)

- xvpvncproxy_host=[host] - ip del host (por defecto 0.0.0.0)

Como cliente se necesita un cliente especial Java que es una versión especial de TightVNC modificada para soportar los token de autenticación.

Para obtener ese cliente se puede hacer desde github

```
$git clone https://github.com/cloudbuilders/nova-xvpvncviewer
$cd nova-xvpvncviewer
$make
```

Una vez que tengamos el cliente, para crear una conexión, debemos primero obtener la URL de acceso a través de novaclient, como se ha comentado anteriormente, y luego realizar la conexión a través del cliente

Para obtener la URL de acceso:

```
$nova get-vnc-console [server_id] xvpvnc
```

Para realizar la conexión al host desde el cliente:

```
$java -jar VncViewer.jar [access_url]
```

Acceso VNC a través de navegador web

Al igual que para el cliente Java, se debe obtener la URL de acceso:

```
$nova get-vnc-console [server_id] novnc
```

Una vez obtenida se le pasa al navegador web para realizar la conexión