UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Creación de sistema Cloud con OpenStack

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alejandro Carlos Osuna Fontan

Tutor: Federico Silla

Julio 2016

Resumen

En este trabajo hemos realizado la implementación completa un sistema de Cloud Computing con la tecnología OpenStack. Hemos realizado un pequeño estudio de las diferentes opciones libres de sistemas cloud. Se ha realizado un estudio cronológico del origen de OpenStack, qué empresas lo utilizan y con qué fin. Por otro lado hemos analizado la arquitectura del sistema y los diferentes módulos que dispone. Además hemos realizando una primera implantación básica del sistema con el objetivo de ofrecer una guía de implantación y una segunda implantación con módulos avanzados, donde hemos analizado las diferentes funciones que nos ofrece un sistema cloud. Finalmente hemos analizado las sobrecargas que aparecen a causa de la virtualización de los sistemas principalmente, sobrecarga en Escritura/Lectura de disco, de transferencia de red y ejecución de aplicaciones.

Palabras clave: openstack, cloud, computing, computacion, nube.

Abstract

In this work we have carried at a complete implementation of a cloud computing system by leveraging the OpenStack technology. We have performed a small study of the different available options of cloud systems. A chronological study of OpenStack's, which companies use it and for witch propose. On the other hand, we have analysed the architecture of the system and the different modules that it comprises. We have performed a first basic installation of the system with the aim of offering a guide of installation and a second implantation with advanced modules, where we have analysed the different functions that a cloud system offers us. Finally, we have analysed the overloads that appear because of the virtualization of systems. We have made this analysis studying, it overloads in W/R on disk, network transfer and execution of applications.

Keywords: openstack, cloud, computing.

Tabla de contenidos

1. Introducción

La utilización de sistemas informáticos en las empresas es algo habitual desde hace ya años. Estos sistemas informáticos se emplean para agilizar diversos procesos como la contabilidad, la creación de nóminas, la utilización de bases de datos de la empresa, etc. En numerosos casos estos sistemas informáticos también se pueden emplear para automatizar parte del trabajo, llevando a cabo cómputos más o menos complejos. Hasta hace unos años, lo más habitual para las empresas era adquirir estos sistemas informáticos, los cuales se instalaban en las propias dependencias de las empresas. No obstante, algunos de esos equipos, principalmente los servidores, conllevan una gran inversión económica, tanto de adquisición como de mantenimiento. En este sentido, al estar alojados en las dependencias de la empresa, ocupan un espacio físico que de otra forma la empresa no tendría por qué pagar. Además, el consumo de electricidad de estos servidores puede llegar a ser importante, encareciendo en algunos casos la factura eléctrica. Por otra parte, hace falta realizar el mantenimiento de estos equipos, coste que de nuevo debe asumir la empresa contratando, por ejemplo, personal especializado para llevarlo a cabo.

Durante las últimas décadas, se ha producido la evolución de diversas tecnologías que, de forma conjunta, ha propiciado un cambio en el modelo descrito en el párrafo anterior. Estas tecnologías son la potencia de los equipos informáticos, la evolución de las infraestructuras y las tecnologías de red. En primer lugar, la capacidad computacional de los equipos informáticos ha aumentado hasta extremos insospechados hace unas décadas. La evolución en las tecnologías de integración VLSI ha propiciado la comercialización a precios populares de procesadores multi-núcleo a los cuales se les añade una cantidades de memoria RAM muy grandes. Tal es la potencia computacional alcanzada que a menudo ocurre que un único usuario no es capaz de consumir dicha potencia en su totalidad. Esto ha motivado una evolución en las infraestructuras de cómputo (segunda tecnología mencionada anteriormente). En este sentido, dado que los equipos informáticos han alcanzado unas potencias computacionales mayores de las que un único usuario puede consumir, se ha hecho habitual el uso de máquinas virtuales en los centros de datos de las empresas e instituciones. Una máquina virtual es un software el cual nos permite simular un equipo completo dentro de nuestro propio sistema. Este tipo de sistemas nos permiten la ejecución de aplicaciones o servicios como si de un equipo real se tratase, ofreciendo en un mismo equipo real varios sistemas virtuales simultáneamente. Gracias a esto podemos aprovechar al máximo los recursos del equipo, con el consecuente ahorro económico y de espacio. Otra de las ventajas que ofrecen las máquinas virtuales es la facilidad con la que podemos recuperar un sistema completo, ya que al fin y al cabo, los sistemas virtualizados son ficheros en nuestro equipo. Finalmente, la evolución en la potencia computacional de los equipos y la evolución en las infraestructuras ha confluido con la generalización del acceso a Internet de banda ancha. Esta confluencia ha motivado un cambio en los hábitos de numerosas empresas e instituciones, donde para substituir a sus servidores propios aparece un nuevo modelo de trabajo al que se denomina Computación en la nube (*Cloud Computing*).

El *Cloud Computing* es un paradigma de computación que consiste en ofrecer como servicio los diferentes requisitos de un sistema informático convencional (hardware y software) a través de Internet. La forma más sencilla de pensar en cloud computing es considerar un centro de datos que proporciona a sus usuarios remotos computadores completos basados en el uso de máquinas virtuales. Esto es, cada usuario dispone de una o varias máquinas virtuales que se ejecutan concurrentemente con las máquinas virtuales de otros usuarios. Esta ejecución concurrente de diversas máquinas virtuales en el mismo servidor hace que la utilización del hardware aumente de forma considerable, ayudando de esta manera rentabilizar la inversión en infraestructura.

Aunque el modelo descrito en el párrafo anterior podría verse como la forma más sencilla de cloud computing, en realidad existen tres modelos de servicio en la nube: *Infraestructure as a Service*, *Platform as a Service* y *Software as a Service*.

- **Infrastructure as a Service (IaaS)**: consiste en ofrecer al usuario todo el sistema físico de manera virtualizada, compuesto de servidores virtuales, conexiones de red, ancho de banda, direccionamiento y balanceadores de carga. Todo esto es ofrecido mediante servidores físicos que ceden una parte de su potencia de procesamiento a los clientes. Este es el modelo descrito en el párrafo anterior. Las ventajas de este tipo de servicios son escalabilidad del sistema (si el cliente necesita más potencia, simplemente debe solicitarlo), evita la inversión por pate del cliente en un sistema informático completo (incluyendo adquisición, mantenimiento,...), permite la movilidad ya que es un acceso normalmente vía web y seguridad gracias a la redundancia de equipos.

- **Platform as a Service (PaaS)**: consiste en ofrecer al cliente un entorno para desarrollar y ejecutar sus aplicaciones al que puede acceder mediante un navegador web. Estos entornos suelen estar compuestos de un sistema operativo, un entorno de desarrollo, sistema de gestión de base de datos, etc. Aunque, como todo en la nube, se suele adaptar a las necesidades del usuario. La principal diferencia entre IaaS y PaaS, es que en PaaS el proveedor te ofrece un acceso a una máquina ya configurada, es decir, dispone de un sistema operativo y el cliente no tiene que gestionar sus maquinas virtuales, ya que esa tarea la realiza el proveedor. Los beneficios de contratar este tipo de servicios son, además de que no es necesario invertir en un infraestructura física igual que en IaaS, que en PaaS los sistemas son más accesibles para usuarios no expertos, ofrecen una gran flexibilidad a la hora de crear su plataforma adaptada totalmente a sus necesidades, adaptabilidad en caso de que las necesidades del cliente cambien, al igual que el resto de modalidades ofrecen movilidad ya que con son accesibles desde un navegador (permitiendo a empresas con diferentes sedes trabajar en un mismo servidor desde cualquier lugar) y una gran seguridad tanto de acceso como de redundancia de los datos.

- **Software as a Service (SaaS)**: consiste en ofrecer aplicaciones como servicio. Existen numerosos ejemplos actualmente (Google, Twitter, Facebook, Office 365, Google Drive, etc). Es muy similar a lo que hasta ahora existía en cuanto a software, con algunas particularidades como que no se adquiere el software, sino que se “alquila” (es decir, que requiere de una suscripción, en el

caso de que sea de pago). Algunas de las múltiples ventajas que ofrecen estos servicios son que se evitan costes de infraestructura ya que la capacidad de cómputo la ofrece el proveedor, no tiene costes de alta (como se ha comentado, el software se "alquila"), es escalable según las necesidades del usuario, el software siempre está actualizado ya que el proveedor se encarga de ello, es accesible desde cualquier dispositivo y desde cualquier lugar y las aplicaciones pueden personalizarse para cada usuario.

Independientemente de cual de los tres modelos mencionados se use, no hay que olvidar que un sistema *Cloud* al final es un software que se implanta sobre un servidor, por lo que como cualquier software, existen soluciones de tipo libre y propietarias. Analizando ambos tipos de soluciones, observamos que cada vertiente tiene sus ventajas y desventajas:

- Soluciones Cloud propietarias: por lo general son sistemas que están muy testeados. Además el vendedor se encarga de toda la implantación, las actualizaciones. El mantenimiento depende únicamente del proveedor. Por otro lado, son de pago (mensual, trimestral, anual, etc) y suelen tener limitaciones en cuanto a flexibilidad, ya que si el cliente necesita alguna modificación, el proveedor es el único que puede hacerla.

- Soluciones Cloud libres: son sistemas muy flexibles. Una vez implantado no se depende de ningún proveedor (en el caso de que se opte por una implantación mediante proveedor) y no conllevan costes de adquisición y uso. En contra de este tipo de soluciones podemos decir que son más complicados de gestionar.

En este trabajo fin de grado se explora el uso de un sistema cloud. Para ello nos decantamos por las soluciones libres. Haciendo un pequeño estudio sobre las principales plataformas Cloud que existen, entre las que se encuentran OpenStack, OpenNebula, Eucalyptus y CloudStack, nos decantamos por OpenStack. Los motivos por los que nos decantamos por esta solución son que tiene gran cantidad de documentación y que dispone de una comunidad bastante considerable y muy activa. A nivel técnico también es importante mencionar que al ser tan modular tiene una capacidad de escalado muy interesante ya que puede trabajar tanto con un único servidor como con miles de ellos en un centro de datos de gran tamaño

OpenStack es un proyecto de software libre y de código abierto de computación en la nube. Es distribuido bajo licencia Apache 2.0. Es la alternativa libre de Amazon Web Services. Este proyecto surgió a mediados del 2010 a causa de las limitaciones del sistema *Eucalyptus* desarrollado por Hewlett-Packard. La NASA hasta esa fecha utilizaba el sistema de *Hewlett Packard*, pero con el tiempo las necesidades de cómputo aumentaban exponencialmente, por lo que se necesitaba un sistema altamente escalable (del orden de un millón de servidores). Además *Eucalyptus* estaba orientado principalmente a la Nube Privada (Private Cloud). Para que el sistema cubriese las necesidades de la agencia espacial, necesitaban poder modificar el código del núcleo, cosa que no era posible con Eucalyptus. Por todo esto la NASA y Rackspace anunciaron que estaban desarrollando su propio software de IaaS. En el proyecto OpenStack actualmente están involucradas más de 200 empresas (entre las que se

encuentran AMD, Cisco, Google, IBM, Red Hat, etc). Para comprender la importancia que tiene OpenStack debemos observar qué empresas lo utilizan y con qué objetivo:

- Computación de alto rendimiento: para este tipo de objetivo se necesita proveer de almacenamiento, computación y red para realizar gran cantidad de cálculos en base a información procesada. El departamento de IT (Information Technology) del CERN lo utiliza en sus experimentos usando la información del Gran Colisionador de Hadrones. Por otro lado, como resulta obvio, la NASA y Rackspace utilizan OpenStack con este mismo fin.

- Hosting Web: sistemas cloud dedicados a ofrecer servicios de hosting web. La empresa GoDaddy utiliza OpenStack.

- Servicios Web/eComerce: servicios de venta online a través de páginas web. Empresas como eBay y Best Buy utilizan OpenStack.

- Procesamiento de vídeo y distribución de contenidos: servicios de video bajo demanda. Comcast (más conocida por su filial NBC) emplean OpenStack.

- Cloud Privado: dedicado a ofrecer servicios cloud de manera interna. Empresas como Paypal utilizan OpenStack para ofrecer sus servicios de pagos online.

Otras empresas que también utilizan OpenStack son Redhat como cloud público, CloudScaling lo usa como plataforma para implementar cloud privados para sus clientes y VMWare, la cual utiliza el cloud con su sistema privativo de virtualización.

Teniendo en cuenta la cantidad de empresas importantes del sector que utilizan y desarrollan esta tecnología, se puede observa la demanda de especialistas en computación en la nube. Además, es de esperar que esta demanda vaya en aumento con el paso del tiempo. Por otro lado, también hay que tener en cuenta que las pequeñas empresas sean las próximas en migrarse a estos sistemas cloud por el ahorro que supone en costes informáticos. En este sentido, parece claro que complementar la formación recibida durante los cuatro años de estudio del Grado en Ingeniería Informática con cierta formación en sistemas cloud puede resultar interesante para el futuro profesional de un graduado en Informática por esta universidad. Es por ello que el objetivo principal de este trabajo fin de grado es estudiar cómo funciona OpenStack, cómo realizar una primera implantación funcional.

El objetivo principal del trabajo fin de grado se desglosa en los siguientes objetivos específicos:

1. Instalar OpenStack en un equipo.
2. Realizar una primera configuración.
3. Lanzar instancias de maquinas virtuales CirrOS, Ubuntu y Windows 7.
4. Realizar pruebas con Benchmarks para analizar la sobrecarga en lectura/escritura en disco, en transferencia de red y tiempo de ejecución de una aplicación.
5. Ampliar instalación añadiendo nuevos módulos que puedan ser útiles para implantaciones de OpenStack mas complejas.

6. Probar el sistema con los nuevos módulos.

Nótese que este trabajo fin de grado complementa la formación recibida y está directamente relacionado con el plan de estudios del Grado de Ingeniería Informática con las siguientes asignaturas: Fundamentos de sistemas operativos, Redes de computadores, Interfaces persona computador, Concurrencia y sistemas distribuidos, Bases de datos y sistemas de información, Computación paralela, Tecnologías de sistemas de información en la red, Administración de sistemas, Desarrollo web, Diseño y configuración de redes de área local, Diseño, configuración y evaluación de los sistemas informáticos, Tecnología de bases de datos, Desarrollo centrado en el usuario, Integración de aplicaciones, Redes corporativas, Seguridad en redes y sistemas informáticos, Sistemas y servicios en red.

Finalmente, la presente memoria se organiza la siguiente manera:

Capítulo 1. Introducción: ha descrito motivación y objetivos de este trabajo.

Capítulo 2. OpenStack: se realiza un estudio de la arquitectura OpenStack y su funcionamiento

Capítulo 3. Instalación y configuración básica de OpenStack: se describe el modo de instalación mediante método manual y mediante Devstack

Capítulo 4. Funcionamiento básico de OpenStack: se muestra las diferentes opciones y menús que ofrece la interfaz gráfica de OpenStack.

Capítulo 5. OpenStack Avanzado: se describe la instalación de instalación de OpenStack con algunos módulos extra, se realizan pruebas de rendimiento y análisis de resultados

Capítulo 6. Conclusiones: se comentan los contratiempos encontrados y las soluciones aplicadas y pequeño análisis de los resultados del trabajo fin de grado.

2. OpenStack

Como se ha mencionado en el apartado anterior, OpenStack es un sistema que ofrece una infraestructura como servicio (IaaS), en la que los usuarios podrán albergar sus sistemas virtualizados. Una de las principales características del sistema es su modularidad, que otorga una gran flexibilidad y escalabilidad. Gracias a esto, se puede distribuir sus componentes en los servidores de un centro de datos de la manera que más convenga.

Para la intercomunicación entre los diferentes módulos, OpenStack utiliza mensajería basada en el estándar *Advances Message Queuing Protocol* (AMQP), que como su nombre indica es un protocolo de nivel de aplicación según el modelo de interconexión de sistemas abiertos (comúnmente conocido como modelo OSI). Esta mensajería es gestionada mediante RabbitMQ, que es un servicio de negociación de mensajería para aplicaciones (aunque acepta otros gestores).

OpenStack tienen sus módulos englobados en dos grupos: los principales (entre los que se encuentran los que realizan las funciones de intercomunicación con el usuario, los que realizan las tareas de computación, los que realizan las tareas de red y los de almacenamiento) y los módulos complementarios (denominados *Shared Services*).

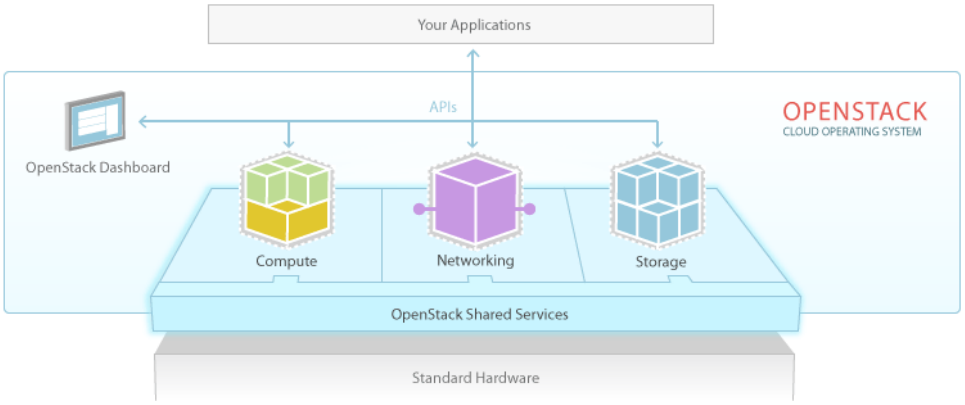


Figura 1: Estructura de OpenStack

A continuación pasamos a describir los módulos principales:

- **Compute:** es el bloque principal del IaaS. Es el encargado de asignar los recursos a las diferentes instancias de máquinas virtuales que se crean en el entorno e identificar a los diferentes usuarios y módulos que intentan interactuar con el sistema. Los módulos que lo integran son Nova (Computación) y Keystone (Autenticación).

- **Storage:** este bloque se dedica a gestionar el almacenamiento del sistema. Por un lado tenemos el almacenamiento de los usuarios (imágenes, documentos, …) y por otro el de las imágenes de las máquinas virtuales instaladas en el sistema (máquinas ya instaladas y listas para ser lanzadas en el sistema). Existen dos niveles de gestión de la información de los usuarios:

  - Nivel de objeto: se refiere a la gestión del almacenamiento a nivel de objetos concretos, por ejemplo un fichero, que es replicado en los diferentes servidores que componen el sistema.

  - Nivel de bloque: es la parte encargada de ofrecer la persistencia de la información al sistema. Gestiona la creación, agregación y desagregación de los dispositivos de bloques a los servidores.

  Los módulos que lo componen son Glance (Servicio de imágenes de las máquinas virtuales), Cinder (Almacenamiento a nivel de bloque) y Swift (Almacenamiento de objetos).

- **Networking:** es la parte que separa el sistema cloud del exterior. Nos permite crear redes virtuales, con todos sus componentes (switch, router, etc). Esto nos permite crear una arquitectura completa de red, brindándonos una seguridad extendida.

  El modulo principal que lo gestiona es Neutron, aunque hay que destacar que el modulo Nova (Computación), incluye internamente virtualización de red aunque a un nivel básico (solo permite la interconexión entre instancias lanzadas).

- **Dashboard:** es el bloque que permite a los usuarios y al administrador interactuar con el sistema. Tiene tres modos de interacción:
  - Línea de comandos
  - Web (si instalamos Horizon)
  - RESTful API

  Dashboard tiene como módulo principal OpenStack Client (Línea de Comandos), aunque en este mismo bloque también podemos incluir el módulo Horizon (Interfaz Web), aunque según OpenStack está incluido en los *Shared Services*.

Respecto a los módulos complementarios (Shared Services), se trata de módulos que están preparados para la integración con el sistema. Dependiendo del objetivo del sistema, se puede necesitar estos módulos. Este bloque está compuesto por:

- **Ceilometer (Telemetría):** este módulo permite monitorizar el sistema, tiene funciones de benchmarking y de estadísticas de rendimiento.
- **Heat (Orquestación):** permite implantar aplicaciones desarrolladas para la ejecución en la nube utilizando plantillas para su despliegue.
- Trove (Base de datos): este módulo permite ofrecer bases de datos relacionales y no relacionales como servicio.
- **Designate (Servicio DNS):** ofrece servicios de resolución de nombres integrado en el sistema cloud (SaaS).
- **Zaqar (Servicio de Mensajería):** es un servicio de mensajería para desarrolladores web. Combina las ideas del servicio pionero Amazon SQS con soporte adicional para eventos de broadcast.
- **Barbican (Gestión de claves):** módulo dedicado al almacenamiento y gestión de secretos tales como contraseñas, claves de encriptación y certificados X.509. Está orientado para ser una potente herramienta para todos los entornos, incluido los sistemas cloud.
- **Congress (Gobierno):** permite ofrecer Políticas de Sistema como servicio. Se encarga de automatizar la definición y aplicación de políticas en sistemas cloud, debido a que este tipo de sistemas son dinámicos.
- **Sahara (Elastic MapReduce):** este módulo nos permite crear un cluster de tipo Hadoop o Spark, mediante unos sencillos pasos.
- **Ironic (Bare-Metal):** este módulo brinda otro modo de ofrecer los servicios cloud. En vez de virtualizar los sistemas, lo que hace es ofrecer una parte del sistema sin virtualizar, con la ventaja de no sufrir la sobrecarga que nos da el HyperVisor.
- **Manila (Sistemas de Ficheros Compartidos):** permite ofrecer almacenamiento compartido. Este almacenamiento compartido puede utilizarse desde las diferentes instancias que están en funcionamiento en el cloud o también podría ofrecerse como servicio.
- **Magnum (Contenedores):** este módulo permite hacer despliegues automatizados de sistemas con aplicaciones distribuidas mediante un contenedor (actualmente permite utilizar Docker Swarm y Kubernetes).
- **Murano (Catalogo de Aplicaciones):** permite a los desarrolladores diseñar aplicaciones preparadas para su ejecución en entornos cloud.

Como se ha mencionado anteriormente, dependiendo del objetivo que tenga el sistema que se quiera implementar, necesitaremos unos módulos u otros. Vamos a revisar algunas de las configuraciones que utilizan las grandes organizaciones:

- Computación de alto rendimiento: como se comentaba en el capítulo anterior, el CERN utiliza este sistema para realizar los cálculos para el colisionador de hadrones. Para este uso utilizan Nova (Computación), Glance (Servicio de imágenes de las máquinas virtuales), Cinder (Almacenamiento a nivel de bloque), Keystone (Autenticación), Ceilometer (Telemetría), Heat (Orquestación) y Horizon (Interfaz Web).
- Web Hosting: este servicio consiste en ofrecer a los clientes servicios para publicar sus páginas web. Los módulos serían Nova (Computación), Neutron (Red), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación), Ceilometer (Telemetría) y Horizon (Interfaz Web).
- Clouds públicos: en este tipo de servicio se ofrece una infraestructura (IaaS) para que los usuarios puedan instalar sus servicios. Los módulos para este tipo de configuraciones serían Nova (Computación), Cinder (Almacenamiento a

nivel de bloque), Swift (Almacenamiento de objetos), Neutron (Red), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación) y Designate (Servicio DNS).

- Servicios Web y Comercio electrónico (eComerce): dedicado a dar servicios y a la venta online. Para este objetivo utilizaríamos Nova (Computación), Cinder (Almacenamiento a nivel de bloque), Neutron (Red), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación), Horizon (Interfaz Web) y Trove (Base de Datos).
- Big Data: otro de los servicios que, junto con la Computación en la Nube, son de los que más destaca actualmente. El *Big Data* consiste en la gestión, el procesamiento y almacenamiento de grandes volúmenes de información. Para esta finalidad los módulos indicados serían Nova (Computación), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación), Neutron (Red), Horizon (Interfaz Web), Sahara (Elastic MapReduce) e Ironic (Bare-Metal).
- DBaaS: Base de datos como servicio (*Database as a Service)*, como su nombre indica consiste en ofrecer un servicio de base de datos. Para este fin usaríamos Nova (Computación), Cinder (Almacenamiento a nivel de bloque), Neutron (Red), Glance (Servicio de imágenes de las máquinas virtuales), Keystone (Autenticación), Swift (Almacenamiento de objetos), Horizon (Interfaz Web), Ironic (Bare-Metal), Trove (Base de Datos) y Designate (Servicio DNS).
- Procesamiento de vídeo y distribución de contenidos: este tipo de servicios está actualmente muy extendido. Dedicado a ofrecer vídeos bajo demanda, los cuales deben ser procesados y entregados al cliente para su visualización. Para este tipo de servicios se usarían los módulos Nova (Computación), Neutron (Red), Keystone (Autenticación) y Swift (Almacenamiento de objetos).

Cabe destacar que muchas empresas no utilizan OpenStack con sus módulos oficiales, sino que realizan una combinación entre los módulos y sus propias tecnologías, como es el caso de VMWare, que utiliza toda la plataforma de OpenStack con su sistema de virtualización.

Por último, mencionar que OpenStack dispone de APIs para interactuar con el sistema. Cada módulo dispone de su propia API. Los principales métodos de comunicación mediante API son:

- cURL: este tipo de API funciona como una línea de comandos que permite enviar peticiones HTTP y recibir respuestas mediante HTTP.
- Línea de comandos OpenStack: como se ha comentado anteriormente, es el método principal para intercomunicarse con el sistema.
- Clientes REST: OpenStack dispone de una API RESTful, por lo que con cualquier cliente REST podemos interactuar con el sistema.
- *Software Development Kit*: los SDK nos permiten escribir programas para crear y gestionar los recursos en nuestro sistema cloud. Dispone SDK para los lenguajes Java, Node.Js, Python, Ruby, .NET y PHP.

3. Instalación y configuración básica de OpenStack

Una vez estudiada en profundidad la estructura y las formas de interactuar con OpenStack, vamos a realizar una primea instalación básica. Nuestra instalación inicial se realizará en un único equipo ya que no disponemos en local de servidores para realizar una instalación más grande. Por otro lado, se podría pensar en disponer de varios servidores localizados físicamente en alguno de los laboratorios de la Escuela Técnica Superior de Ingeniería Informática. No obstante, revisando una experiencia del trabajo de otro alumno[14], al tener que acceder en remoto y realizar configuraciones de red, es bastante tedioso trabajar en remoto.

OpenStack da soporte para la instalación en distribuciones Linux Ubuntu, RedHat, CentOS, openSUSE y SUSE. Esto no quiere decir que no pueda ser instalado en otros sistemas, pero para dichas distribuciones ofrecen guías de instalación muy detalladas. Nuestra instalación la realizaremos sobre una Ubuntu 14.04 (LTS) Server, ya que es la distribución con la que más familiarizado está el autor del presente trabajo final de grado y además porque esta distribución tiene una de las comunidades más grandes y activas.

Existen dos métodos de instalación para OpenStack en Ubuntu: la instalación manual y la instalación con Devstack. La principal diferencia que existe entre ambos métodos es que Devstack está orientado para un entorno de desarrollo y pruebas, por lo que no es recomendable realizar un despliegue final. Algunas de las particularidades de este método es que no instala todo el sistema como servicio, por lo que no se inicia automáticamente con el sistema, realiza una instalación por defecto (aunque puede ser modificada mediante un fichero de configuración, que comentaremos posteriormente), no permite realizar algunas tareas de seguridad (como copias de seguridad), permite editar componentes sin tener permisos de *root*, entre otros. Por otro lado, la instalación manual, aunque es más laboriosa, permite crear un sistema totalmente a medida.

Cabe destacar otro método de instalación, que aunque únicamente es compatible con sistemas RedHat Enterprise Linux, CentOS y derivados, nos permite implantar un sistema OpenStack completamente funcional, mediante un script llamado Packstack. Este método está orientado a la implantación de OpenStack en un solo nodo.

A pesar de que para este trabajo se ha elegido el sistema Ubuntu 14.04 y el método Devstack, antes vamos a realizar un análisis de cómo se realiza la instalación manual básica para realizar un despliegue de OpenStack.

Primero se debe tener en cuenta que para un entorno empresarial, lo habitual es disponer de varios servidores donde se realizaría la instalación del sistema. Para este tipo de entorno, la arquitectura mínima sería tener un nodo que actuaría de controlador y otro que realizaría las tareas de computación.

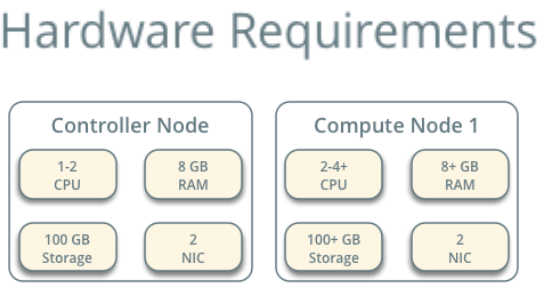


Figura 2: Arquitectura mínima de un implantación operativa en un entorno empresarial

Como podemos ver en la figura, los requisitos hardware para poder trabajar con OpenStack son, en el caso del nodo controlador, una o dos CPUs, ocho gigabytes de RAM, cien gigabytes de almacenamiento y dos tarjetas de red. Cabe destacar que cuanto mayor sea el número de nodos que compongan el sistema, más carga tendrá que soportar el nodo controlador. En el caso del nodo de computación, se observa que son superiores con respecto al nodo controlador. En este caso necesita de dos a cuatro CPUs, ocho o más gigabytes de RAM, cien gigabytes o más de almacenamiento y dos tarjetas de red.

El nodo controlador ejecuta el servicio de Identidad (Keystone), el cual permite dar seguridad a las comunicaciones entre los diferentes módulos, el servicio de Imágenes (Glance), que nos permite gestionar las imágenes para lanzar instancias de máquinas virtuales, realiza operaciones de balanceo de carga, enviando las instancias a un nodo u otro (en caso de disponer de varios nodos de computación) dependiendo de la carga, gestiona la red y ejecuta el servicio de Dashboard (Horizon), para interactuar con el sistema. También incluye servicios internos como base de datos SQL, para almacenar datos del sistema (datos de seguridad, nodos activos, carga de nodos, etc), cola de mensajes, la cual permite la intercomunicación entre los nodos, y servicios de *Network Time Protocol* (NTP).

Por otro lado, los nodos de computación ejecutan el *hypervisor*, que permite ofrecer una parte del nodo para lanzar las instancias de las máquinas virtuales. Por defecto utiliza *KVM Hypervisor*. También ejecuta un agente de red, que permite intercomunicar las instancias con las redes virtuales y provee seguridad mediante servicios de cortafuegos.

Partiendo de una instalación limpia de Ubuntu Server 14.04 LTS, vamos a realizar la instalación del nodo controlador. Lo primero que nos recomiendan es que configuremos manualmente las tarjetas de red, ya que las asignaciones por Dynamic Host Configuration Protocol (DHCP), protocolo que nos da una configuración IP dinámicamente para poder navegar, dan problemas a la hora de interconectar los nodos. Esto es aplicable tanto al nodo controlador como al nodo de computación. Otra de las recomendaciones que nos dan es que instalemos *Chrony*, que es el servidor Network Time Protocol (NTP), protocolo que permite la sincronización de los relojes de los equipos que se están comunicando, para que los servicios se sincronicen correctamente.

1.- Para realizar las configuraciones estáticas debemos editar el fichero /etc/network/interfaces con las siguientes líneas:

```
auto INTERFACE_NAME
iface  INTERFACE_NAME inet manual
address 10.0.0.11
netmask 255.255.255.0
Gateway 10.0.0.1
```

Reemplazamos INTERFACE_NAME por el nombre de la interfaz (similar a *etho*). Las direcciones IP son las que utilizaremos en este trabajo. En otras instalaciones no tiene por que utilizarse las mismas direcciones IP.

2.- Como no disponemos actualmente de servidor DNS, es recomendable añadir las siguientes líneas al fichero de host, ubicando en /etc/hosts , para poder utilizar nombres de host en vez de direcciones IP:

```
# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1
```

Procedemos a la instalación y configuración del servicio NTP mediante el paquete Chrony.

1.- Ejecutamos el siguiente comando:

```
# apt-get install chrony
```

2.- Una vez finalice la instalación, editamos el fichero de configuración que se encuentra en la ruta /etc/chrony/chrony.conf y añadimos la siguiente línea:

```
server NTP_SERVER iburst
```

Substituyendo NTP_SERVER por el nombre de host del equipo o la dirección IP de un servidor con mayor precisión. Puede haber más de una entrada *server* en el fichero de configuración. Por defecto se suele poner varios servidores públicos para la sincronización, pero se puede utilizar uno que disponga la empresa.

3.- Para que los cambios surtan efecto, reiniciamos el servicio:

```
# service chrony restart
```

Una vez hemos configurado el servidor NTP en nuestro controlador, vamos a añadir los repositorios oficiales de OpenStack:

1.- Añadimos los repositorios:

```
# apt-get install software-properties-common
# add-apt-repository cloud-archive:mitaka
```

2.- Actualizamos los repositorios y el sistema:

```
# apt-get update && apt-get dist-upgrade
```

3.- Instalamos el cliente OpenStack:

```
# apt-get install python-openstackclient
```

La mayoría de servicios que componen OpenStack utilizan base de datos SQL. Para nuestra distribución nos recomiendan utilizar MariaDB.

1.- Instalamos el paquete:

```
# apt-get install mariadb-server python-pymysql
```

2.- Elegimos una contraseña para la cuenta *root* de la base de datos.

3.- Creamos un archivo de configuración en /etc/mysql/conf.d/openstack.cnf. En la sección *[mysqld]*, añadimos una línea para permitir acceso desde otros nodos mediante la red de gestión:

```
[mysqld]
...
bind-address = 10.0.0.11
```

La dirección IP es la del nodo controlador.

Añadimos también otras líneas en la misma sección que son de gran ayuda:

```
[mysqld]
...
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
character-set-server = utf8
```

En estas líneas realizamos la definición del tipo de motor de base de datos que utilizaremos (que es *innodb*), y utilizaremos el formato de codificación UTF8.

Para que los cambios surtan efecto debemos reiniciar el servicio:

```
# service mysql restart
```

Nos recomiendan que ejecutemos el siguiente script para hacer segura la base de datos:

```
# mysql_secure_installation
```

Vamos a instalar y configurar el gestor de colas que utiliza OpenStack para la intercomunicación de los módulos, que es RabbitMQ (aunque también se puede utilizar Qpid o ZeroMQ).

1.- Instalamos el paquete RabbitMQ:

```
# apt-get install rabbitmq-server
```

2.- Añadimos el usuario *openstack*:

```
# rabbitmqctl add_user openstack RABBIT_PASS
Creating user "openstack" ...
...done.
```

Cambiamos RABBIT_PASS por la contraseña que queramos utilizar

3.- Damos permisos de lectura y escritura al usuario *openstack*:

```
# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
...done.
```

Por último, vamos a instalar Memcached, el cual es utilizado por el servicio de Identidad para almacenar en caché los *tokens* de autenticación:

1.- Instalamos el paquete:

```
# apt-get install memcached python-memcache
```

2.- Editamos el fichero situado en /etc/memcached.conf para que sea accesible desde otros nodos mediante la red de gestión (similar a lo que hemos realizado en el apartado de MariaDB):

```
-l 10.0.0.11
```

3.- Reiniciamos el servicio para aplicar los cambios:

```
# service memcached restart
```

Hasta aquí hemos dejado en el nodo controlador el entorno preparado para satisfacer las necesidades de OpenStack. Vamos a proceder a instalar los servicios propios de OpenStack. Empezamos con el servicio de Identidad (Keystone).

1.- Nos conectamos a nuestra base de datos, creamos una base de datos llamada *keystone* que será utilizada por el módulo para almacenar toda la información que utiliza (claves de acceso, autorización de módulos, etc) y aplicaremos los permisos pertinentes:

```
$ mysql -u root -p

CREATE DATABASE keystone;

GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
```

Substituimos KEYSTONE_DBPASS por la contraseña que deseemos.

2.- Generamos un *token* aleatorio para el administrador que utilizaremos durante la configuración inicial:

```
$ openssl rand -hex 10
```

3.- Deshabilitamos el inicio automático del servicio *Keystone* para después de la instalación:

```
# echo "manual" > /etc/init/keystone.override
```

4.- Instalamos *Keystone, Apache y el Mod WSGI para Apache*:

```
# apt-get install keystone apache2 libapache2-mod-wsgi
```

5.- Editamos el fichero de configuración de *Keystone* ubicado en /etc/keystone/keystone.conf:

```
[DEFAULT]
...
admin_token = ADMIN_TOKEN
```

Substituimos ADMIN_TOKEN por el número aleatorio que hemos generado antes.

```
[database]
...
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

Substituimos KEYSTONE_DBPASS por la contraseña que hemos elegido anteriormente

```
[token]
...
provider = fernet
```

6.- Procedemos a poblar la base de datos de Keystone, que se encarga de crear toda la estructura de la base de datos y el acceso para el módulo:

```
# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

7.- Inicializamos las claves de Fernet:

```
# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
```

Fernet es un formato de mensajería que ofrece la seguridad a los mensajes que se envían entre los módulos mediante encriptación. Es un formato ligero y los tokens no son persistentes (se van generando nuevos invalidando los antiguos), brindando así al sistema de más seguridad.

Hasta aquí hemos realizado las configuraciones propias del servicio Keystone. Ahora vamos a proceder a configurar el servicio Apache el cual se utiliza para recibir las peticiones (mediante REST).

1.- Editamos el fichero de configuración ubicado en /etc/apache2/apache2.conf para referenciar el nombre del servidor al nodo controlador:

```
ServerName controller
```

2.- Creamos el fichero /etc/apache2/sites-available/wsgi-keystone.conf con el contenido siguiente:

```
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone group=keystone
 display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone group=keystone
display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>
```

En este fichero de configuración lo que estamos definiendo son permisos de acceso a la base de datos de Keystone, mediante peticiones REST.

3.-Habilitamos host virtuales en Keystone, para que el servicio pueda recibir las peticiones REST:

```
# ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled
```

4.- Para aplicar los cambios, reiniciamos el servicio:

```
# service apache2 restart
```

5.- [OPCIONAL] Ubuntu crea por defecto una base de datos SQLite. Como nosotros utilizamos nuestro propio servidor de base de datos, podemos eliminarla:

```
# rm -f /var/lib/keystone/keystone.db
```

Ya tenemos el servicio de Identidad funcionando, vamos a proceder ahora con el servicio de Imágenes (Glance).

1.- En primer lugar, como en el caso de Keystone, debemos añadir una base de datos para Glance, la cual contendrá toda la información relacionada con las imágenes para las instancias de máquinas virtuales (tamaños, formato, arquitectura, etc):

```
$ mysql -u root -p
```

```
CREATE DATABASE glance;

GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
```

Substituimos GLANCE_DBPASS por la contraseña que más nos convenga.

2.- Referenciamos las credenciales de *admin* para tener acceso a los comandos que son solo para administradores:

```
$ . admin-openrc
```

Este fichero contiene los datos para definir las variables de entorno que necesitan algunos módulos de Openstack para trabajar. Este fichero tienen un formato similar al siguiente:

```
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controller:35357/v2.0
```

En este caso estaríamos definiendo el usuario admin con la contraseña ADMIN_PASS, que pertenece al proyecto admin y el servicio de identidad se encuentra en controller.

3.- Creamos las credenciales para el servicio *Glance*:

Creamos el usuario *glance*

```
$ openstack user create --domain default --password-prompt glance
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | e38230eeff474607805b596c91fa15d9 |
| name      | glance                           |
+-----------+----------------------------------+
```

Añadimos el rol de Admin a *glance*

```
$ openstack role add --project service --user glance admin
```

Creamos la entidad del servicio *Glance*

```
$ openstack service create --name glance \
  --description "OpenStack Image" image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

4.- Keystone gestiona un catálogo de API endpoints, que son unas entradas en su base de datos que se utilizan para obtener información de otros módulos del sistema (como la disponibilidad, en que servidor se encuentra el módulo, etc) . Creamos los API endpoints de Glance:

```
$ openstack endpoint create --region RegionOne \
  image public http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 340be3625e9b4239a6415d034e98aace |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  image internal http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | a6e4b153c2ae4c919eccfdbb7dceb5d2 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  image admin http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 0c37ed58103f4300a84ff125a539032d |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

5.- Ya podemos realizar la instalación del servicio Glance:

```
# apt-get install glance
```

6.- Editamos el fichero /etc/glance/glance-api.conf:

```
[database]
...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

Con esta línea damos acceso a la base de datos. Substituimos GLANCE_DBPASS por la contraseña que hemos definido en la base de datos.

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = GLANCE_PASS

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

Mediante estas modificaciones, damos al servicio Glance los datos de acceso al servicio de Keystone.

Substituimos GLANCE_PASS por la que hemos introducido en el servicio Keystone

```
[glance_store]
...
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Aquí definimos de donde provienen los ficheros de imágenes y donde los vamos a almacenar.

7.- Editamos el fichero localizado en /etc/glance/glance-registry.conf:

```
[database]
...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

Substituimos GLANCE_DBPASS por la contraseña que hemos elegido para la base de datos. Con esta línea damos acceso a la base de datos.

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = GLANCE_PASS

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

Substituimos GLANCE_PASS por la contraseña que hemos definido en el servicio Keystone.

8.- Poblamos la base de datos:

```
# su -s /bin/sh -c "glance-manage db_sync" glance
```

9.- Por ultimo, reiniciamos los servicios:

```
# service glance-registry restart
# service glance-api restart
```

Ya hemos finalizado la instalación de Glance. A continuación procederemos a la instalación de la parte que gestiona los nodos de computación en el controlador, llamado Nova.

1.-Como en los módulos anteriores, debemos crear una nueva base de datos y aplicarle los correspondientes permisos:

```
$ mysql -u root -p


CREATE DATABASE nova_api;
CREATE DATABASE nova;
```

```
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

Cambiamos NOVA_DBPASS por la contraseña que mejor nos convenga.

2.- Referenciamos las credenciales de *admin* para tener acceso a los comandos que son solo para administradores:

```
$ . admin-openrc
```

3.- Procedemos a crear las credenciales del servicio:

Creamos el usuario *nova*

```
$ openstack user create --domain default \
  --password-prompt nova
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | 8c46e4760902464b889293a74a0c90a8 |
| name      | nova                             |
+-----------+----------------------------------+
```

Añadimos el rol de *admin*

```
$ openstack role add --project service --user nova admin
```

Creamos el servicio de indentidad de *nova*

```
$ openstack service create --name nova \
  --description "OpenStack Compute" compute
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 060d59eac51b4594815603d75a00aba2 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

4.- Creamos los API endpoints de Nova:

```
$ openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 3c1caa473bfe4390a11e7177894bcc7b        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | e3c918de680746a586eac1f2d9bc10ab        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 38f7af91666a47cfb97b4dc790b94424        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

5.- Ya podemos realizar la instalación del módulo:

```
# apt-get install nova-api nova-conductor nova-consoleauth \
  nova-novncproxy nova-scheduler
```

6.- Editamos el fichero de configuración ubicado en /etc/nova/nova.conf

```
[DEFAULT]
...
enabled_apis = osapi_compute,metadata
```

```
[api_database]
...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
```

Substituimos NOVA_DBPASS por la contraseña que escogimos en la base de datos.

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Substituimos RABBIT_PASS por la contraseña que asignamos al usuario *openstack* en RabbitMQ.

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

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS
```

Reemplazamos NOVA_PASS por la contraseña que asignamos al usuario *nova* en el servicio de Identidad.

Añadimos en la sección default la siguiente línea:

```
[DEFAULT]
...
my_ip = 10.0.0.11
```

La dirección IP corresponde a la red de gestión, al igual que en las anteriores configuraciones.

```
[vnc]
...
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip

[glance]
...
api_servers = http://controller:9292

[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

7.- Una vez realizado las modificaciones en el fichero de configuración, pasamos a poblar la base de datos, para definir la estructura de la base de datos:

```
# su -s /bin/sh -c "nova-manage api_db sync" nova
# su -s /bin/sh -c "nova-manage db sync" nova
```

8.- Por último, reiniciamos el servicio Nova y sus componentes para que los cambios surtan efecto:

```
# service nova-api restart
# service nova-consoleauth restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

Ya tenemos Nova funcionando en nuestro nodo controlador. Procedemos a la instalación y configuración del módulo de red Neutron.

1.- Como en los módulos anteriores, procedemos a añadir una nueva base de datos para Neutron:

```
$ mysql -u root -p

CREATE DATABASE neutron;

GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
```

Reemplazamos NEUTRON_DBPASS por la contraseña que mejor nos convenga.

2.- Referenciamos las credenciales de *admin* para tener acceso a los comandos que son solo para administradores:

```
$ . admin-openrc
```

3.-Creamos las credenciales del servicio, añadimos el rol de admin y creamos el servicio de identidad para Neutron:

```
$ openstack user create --domain default --password-prompt neutron
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | b20a6692f77b4258926881bf831eb683 |
| name      | neutron                          |
+-----------+----------------------------------+
```

```
$ openstack role add --project service --user neutron admin


$ openstack service create --name neutron \
  --description "OpenStack Networking" network
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | f71529314dab4a4d8eca427e701d209e |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+
```

4.- Creamos los API endpoints de Neutron:

```
$ openstack endpoint create --region RegionOne \
  network public http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 85d80a6d02fc4b7683f611d7fc1493a3 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  network internal http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 09753b537ac74422a68d2d791cf3714f |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  network admin http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 1ee14289c9374dffb5db92a5c112fc4e |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Una vez hemos preparado el entorno para instalar el módulo Neutron, tenemos dos opciones de instalación y configuración del mismo. Por un lado tenemos la opción que permite a las instancias de las máquinas virtuales comunicarse con el exterior, es decir, nos permite utilizar la conexión a internet que utilizaríamos normalmente. La segunda opción, a parte de ofrecer conexión a las instancias con la línea del proveedor, también nos brinda el nivel tres del modelo OSI de manera virtual, es decir, nos permite instalar una red virtual (con VPN, routers virtuales, switches virtuales, etc), permitiéndonos diseñar una red sin necesidad de adquirir nuevos equipos. Para la instalación básica utilizaremos la primera opción ya que más adelante realizaremos una configuración con la segunda opción. Cabe decir que la diferencia entre configuraciones no es muy grande (dista de un par de líneas de configuración).

5.- Instalamos el modulo Neutron y sus componentes:

```
# apt-get install neutron-server neutron-plugin-ml2 \
  neutron-linuxbridge-agent neutron-dhcp-agent \
  neutron-metadata-agent
```

6.- Editamos el fichero de configuración ubicado en /etc/neutron/neutron.conf

```
[database]
...
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
```

Reemplazamos NEUTRON_DBPASS por la contraseña que introducimos en la base de datos

```
[DEFAULT]
...
core_plugin = ml2
service_plugins =


[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Reemplazamos RABBIT_PASS por la contraseña del usuario *openstack* de RabbitMQ

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

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

Reemplazamos NEUTRON_PASS por la contraseña que introducimos para el usuario *neutron* en el servicio de identidad.

```
[DEFAULT]
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True

[nova]
...
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

Reemplazamos NOVA_PASS por la contraseña que hemos utilizado para el usuario *nova* en el servicio de identidad.

7.- Editamos el fichero ubicado en /etc/neutron/plugins/ml2/ml2_conf.ini

```
[ml2]
...
type_drivers = flat,vlan
```

```
[ml2]
...
tenant_network_types =

[ml2]
...
mechanism_drivers = linuxbridge

[ml2]
...
extension_drivers = port_security

[ml2_type_flat]
...
flat_networks = provider

[securitygroup]
...
enable_ipset = True
```

En esta configuración hemos definido los tipos de redes que tenemos (redes con VLAN y sin VLAN o Flat), se habilita el linuxbridge que nos permite intercomunicarnos con la red del proveedor y habilitamos el ipset para incrementar la eficiencia de las reglas de los grupos de seguridad.

8.- Editamos el fichero de configuración que se encuentra en /etc/neutron/plugins/ml2/linuxbridge_agent.ini:

```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```


Esta línea define qué interfaz nos da acceso a la red del proveedor. Reemplazamos PROVIDER_INTERFACE_NAME por el nombre de la interfaz que nos da acceso a internet (recordar que nuestro nodo dispone de dos tarjetas de red, una para acceso a internet y otra para gestión interna, su nombre es similar a *etho*).

```
[vxlan]
enable_vxlan = False


[securitygroup]
...
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
```

9.- Realizamos la configuración del agente DHCP para la red virtual:

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
```

10.- Procedemos a configurar el agente de metadatos, que es utilizado para dar credenciales a las instancias:

```
[DEFAULT]
...
nova_metadata_ip = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

Reemplazar METADATA_SECRET por una clave para el proxy de metadatos.

11.- Configuramos Nova para que utilice Neutron (se había comentado anteriormente que Nova incluye un módulo de red, pero es bastante más limitado):

```
[neutron]
...
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
```

Reemplazamos NEUTRON_PASS por la contraseña del usuario *neutron* del servicio de identidad y METADATA_SECRET por la clave que hemos introducido para el proxy de metadatos.

12.- Poblamos la base de datos, definiendo la estructura de la base de datos y la configuración que hemos definido en los ficheros anteriores:

```
# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \
  --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

13.- Reiniciamos los servicios para aplicar los cambios:

```
# service nova-api restart


# service neutron-server restart
# service neutron-linuxbridge-agent restart
# service neutron-dhcp-agent restart
# service neutron-metadata-agent restart
```

Para finalizar la instalación y configuración del nodo controlador procedemos a implantar el módulo Dashboard.

1.- Instalamos el módulo Dashboard:

```
# apt-get install openstack-dashboard
```

2.- Editamos el fichero de configuración ubicado en /etc/openstack-dashboard/local_settings.py:

Definimos el nodo que ejecutará Dashboard, en nuestro caso el nodo controlador:

```
OPENSTACK_HOST = "controller"
```

Permitimos el acceso a todos los host que accedan a Dashboard:

```
ALLOWED_HOSTS = ['*', ]
```

Configuramos el servicio de almacenamiento de sesiones (memcached):

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
    'default': {
         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
         'LOCATION': 'controller:11211',
    }
}
```

Habilitamos la API de identidad v3:

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

Habilitamos el soporte para dominios:

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

Configuramos las versiones de las API:

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}
```

Definimos el dominio por defecto para los usuarios que creemos mediante Dashboard:

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
```

Definimos el rol por defecto para los usuarios que creemos mediante Dashboard:

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

Deshabilitamos el nivel tres (nivel de red) de los servicios de red:

```
OPENSTACK_NEUTRON_NETWORK = {
    ...
    'enable_router': False,
    'enable_quotas': False,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_lb': False,
    'enable_firewall': False,
    'enable_vpn': False,
    'enable_fip_topology_check': False,
}
```

3.- Reiniciamos el servicio apache para aplicar los cambios:

```
# service apache2 reload
```

Ya tenemos nuestro nodo controlador totalmente configurado y listo para funcionar. Vamos a realizar ahora la instalación y configuración del nodo de computación, en el que se ejecutarán las instancias de máquinas virtuales. Primero debemos instalar y configurar el servicio NTP y habilitar el repositorio de OpenStack. Como en el caso anterior, partimos de una instalación de Ubuntu Server 14.04 LTS limpia. Procedemos a realizar loas configuraciones previas de red.

1.- Para realizar las configuraciones estáticas para la red debemos editar el fichero /etc/network/interfaces, con las siguientes líneas:

```
# The provider network interface
auto INTERFACE_NAME
iface  INTERFACE_NAME inet manual
address 10.0.0.31
netmask 255.255.255.0
Gateway 10.0.0.1
```

Reemplazamos INTERFACE_NAME por el nombre de la interfaz (similar a *etho*). Las direcciones IP son las que utilizaremos en este trabajo. En otras instalaciones no tienen por qué utilizarse las mismas direcciones IP.

2.- Como no disponemos actualmente de servidor DNS, es recomendable añadir las siguientes líneas al fichero de host para poder utilizar nombres de host en vez de direcciones IP:

```
# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1
```

Procedemos a la instalación y configuración del servicio NTP.

1.- Instalamos el paquete Chrony ejecutando el siguiente comando:

```
# apt-get install chrony
```

2.- Editamos el fichero de configuración ubicado en /etc/chrony/chrony.conf, borrando todas las líneas que empiezen con la palabra *server* y añadimos esta:

```
server controller iburst
```

El host *controller* es el que hace las funciones de nodo controlador.

3.- Reiniciamos el servicio para aplicar los cambios:

```
# service chrony restart
```

Habilitamos el repositorio oficial de OpenStack donde se encuentran todos los módulos que lo componen:

1.- Ejecutamos el comando siguiente:

```
# apt-get install software-properties-common
# add-apt-repository cloud-archive:mitaka
```


2.- Actualizamos los repositorios y actualizamos el sistema:

```
# apt-get update && apt-get dist-upgrade
```

3.- Instalamos el cliente de OpenStack:

```
# apt-get install python-openstackclient
```

Procedemos a la instalación del módulo Nova, de modo que se ejecutarán las instancias.

1.- Ejecutamos el siguiente comando para instalar el paquete Nova:

```
# apt-get install nova-compute
```

2.- Editamos el fichero de configuración ubicado en /etc/nova/nova.conf:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Reemplazamos RABBIT_PASS por la contraseña que le asignamos al ususario *openstack* del gestor de colas RabbitMQ que instalamos en el nodo controlador.

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

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS
```

Reemplazamos NOVA_PASS por la contraseña que se le asignó al usuario *nova* en el servicio de identidad, ubicado en el nodo controlador.

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Reemplazamos MANAGEMENT_INTERFACE_IP_ADDRESS por la dirección IP de la interfaz de red que utilizamos para gestión, en nuestro caso 10.0.0.31.

```
[DEFAULT]
...
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[vnc]
...
enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html

[glance]
...
api_servers = http://controller:9292
```

```
[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

3.- Para finalizar la configuración del módulo, debemos ejecutar el siguiente comando para comprobar qué tipo de aceleración hardware para máquinas virtuales soporta:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Si el comando devuelve un uno o un número mayor, quiere decir que nuestro nodo soporta la aceleración hardware estándar y no necesita más configuraciones.

4.- En caso de que el comando anterior devuelva un zero, debemos editar el fichero de configuración ubicado en /etc/nova/nova.conf y añadir la siguiente línea:

```
[libvirt]
...
virt_type = qemu
```

Aquí estamos definiendo que el tipo de virtualización será del tipo *qemu*, que se utiliza en equipos que no disponen soporte nativo para virtualización. Este tipo de virtualización nos brinda un rendimiento más óptimo en equipos sin soporte a cambio de tener menos soporte para arquitecturas minoritarias y soporte incompleto de hardware (tarjetas de video, sonido y dipositivos de entrada y salida), por lo que tiene una sobrecarga considerable en aplicaciones que utilicen esos componentes.

5.- Por último reiniciamos los servicios para aplicar los cambios:

```
# service nova-compute restart
```

Ya tenemos listo nuestro nodo de computación para albergar las instancias que necesitemos ejecutar en el sistema. Para finalizar la configuración, procedemos a instalar el último módulo que necesitaremos en el equipo, encargado de gestionar la red de las instancias (Nova).

1.- Ejecutamos el siguiente comando para instalar el paquete Nova:

```
# apt-get install neutron-linuxbridge-agent
```

2.- Editamos el fichero de configuración ubicado en /etc/neutrón/neutron.conf:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Reemplazamos RABBIT_PASS por la contraseña que hemos asignado al usuario *openstack* en el gestor de colas RabbitMQ, que se encuentra en el nodo controlador.

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

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

Como hemos visto anteriormente, este fichero define dónde se encuentra el servicio Keystone y cómo se comunica con él. Reemplazamos NEUTRON_PASS por la contraseña que hemos asignado al usuario *neutron* en el servicio de identidad del nodo controlador.

3.- Como en el caso del nodo controlador, elegimos la opción de únicamente interconectar las instancias con el exterior. Para ello debemos editar el fichero de configuración ubicado en /etc/neutron/plugins/ml2/linuxbridge_agent.ini

```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

Reemplazamos PROVIDER_INTERFACE_NAME por el nombre de la interfaz que nos da acceso a internet (recordar que nuestro nodo dispone de dos tarjetas de red, una para acceso a internet y otra para gestión interna, su nombre es similar a *etho*).

```
[vxlan]
enable_vxlan = False
```

```
[securitygroup]
...
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

4.- Editamos el fichero ubicado en /etc/nova/nova.conf para que utilice Neutron para acceder al exterior:

```
[neutron]
...
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

Reemplazamos NEUTRON_PASS por la contraseña que asignamos al usuario *neutron* en el servicio de identidad ubicado en el nodo controlador.

5.- Reiniciamos los servicios para aplicar los cambios:

```
# service nova-compute restart

# service neutron-linuxbridge-agent restart
```

Ya está completa nuestra instalación manual de OpenStack con una arquitectura compuesta por dos nodos, uno controlador y otro de computación. Una vez analizada esta instalación podemos observar que a pesar de ser más laboriosa, nos brinda una gran flexibilidad en cuanto configuraciones y la elección de algunas tecnologías, como tipo de base de datos, cola de mensajes, etc. A continuación procederemos a realizar la instalación mediante el método de Devstack, el cual instalará por defecto los módulos Keystone, Swift, Glance, Cinder, Nova (con su plugin de red), Horizon y Heat.

Para nuestro trabajo vamos a utilizar un único nodo con las siguientes características:

Processor Name:	Intel Core i7 CPU 4870HQ
Processor Model:	Haswell
Processor Speed:	2,5 GHz
Number of Processors:	1
Total Number of Cores:	4
Memory Size:	16 GB 1600 MHz DDR3
Graphics Model:	Intel Iris Pro 1536 MB AMD Radeon R9 M370X 2048 MB

Hemos optado por un único nodo ya que, según experiencia de otros usuarios, al realizar las configuraciones iniciales, se pierde la conexión, ya que modificamos configuraciones de red, y los equipos que disponemos para realizar pruebas deben ser accedidos de manera remota por VNC, por lo que es una opción poco eficiente de trabajar. Por otro lado no existen grandes diferencias entre implantar el sistema en único nodo y realizarlo en varios nodos (a groso modo la única diferencia se encuentra en el equipo donde se realizan las instalaciones, y eso se ha tratado en la explicación de la instalación manual).

Partiendo de una instalación limpia de Ubuntu Server 14.04 LTS, realizamos la instalación del paquete *git* el cual utilizaremos para la instalación de OpenStack.

1.- Instalamos *git* con el siguiente comando:

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

2.- Una vez finalice el comando, en nuestro directorio actual tendremos una nueva carpeta llamada *devstack*, en la que disponemos de todo lo necesario para realizar la instalación. Para realizar modificaciones en la instalación hay que crear un fichero de configuración llamado *local.conf*, el cual analizaremos y utilizaremos en el capítulo de instalación avanzada. Para instalar OpenStack debemos ejecutar el siguiente comando dentro del directorio *devstack*:

```
./stack.sh
```

La instalación puede durar en torno a la media hora, dependiendo del equipo en el que se trabaje. Lo único que nos solicitará será que definamos las contraseñas para la base de datos, para el servicio de identidad y para el gestor de colas. Una vez finalizado, podremos acceder al Dashboard mediante el navegador web, utilizando la dirección IP del equipo. Lo que realiza Devstack es bajarse el código de las últimas versiones liberadas por los equipos de desarrollo de los repositorios Git, los compila y finalmente los instala en nuestro equipo.

4. Funcionamiento básico de OpenStack

Ya tenemos instalado OpenStack en nuestro equipo. Vamos a realizar un análisis de OpenStack funcionando. Lo primero que debemos hacer para acceder a Dashboard es saber qué dirección IP tenemos asignada, en caso de que accedamos desde otro equipo (si accedemos desde el mismo, podemos acceder mediante la dirección de *loopback* 127.0.0.1).

Una vez sabemos la dirección a la que tenemos que acceder, abrimos el navegador e introducimos dicha IP. Nos aparecerá una pantalla de login como esta:

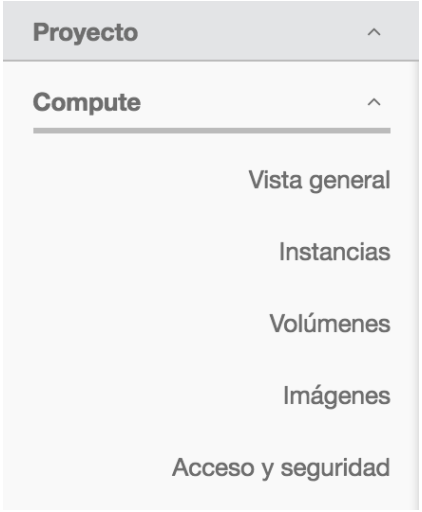
El usuario de acceso es *admin*, y la contraseña es la que hemos introducido para el servicio de identidad cuando Devstack la solicitó al inicio de su ejecución.

Una vez hemos validado el usuario y la contraseña, accederemos a la página principal del sistema:

En nuestro caso, como hay una instancia en ejecución, aparece que para el proyecto llamado *demo* se han asignado esos recursos (la tabla que aparece en el centro de la imagen). Los proyectos sirven para separar la asignación de recursos entre diferentes objetivos. En el lateral se encuentra la barra de navegación que incluye los siguientes apartados:

- Proyecto: este apartado es el principal, donde podremos lanzar instancias, asignar recursos, configurar una red virtual, acceder a las instancias lanzadas mediante VNC en versión web, etc. Aquí aparecerán los menús de configuración de los módulos que agreguemos.
- Administrador: aquí podremos subir las imágenes de máquinas virtuales para lanzar las instancias, crear espacios de almacenamiento, acceder a toda la información del sistema, etc.
- Identity: este submenú es donde podremos gestionar todo lo relacionado con accesos, como usuarios, grupos, roles, proyectos, etc.
- Desarrollador: este apartado está dedicado exclusivamente para revisar el tema que utiliza la interfaz web (usa la tecnología bootstrap).

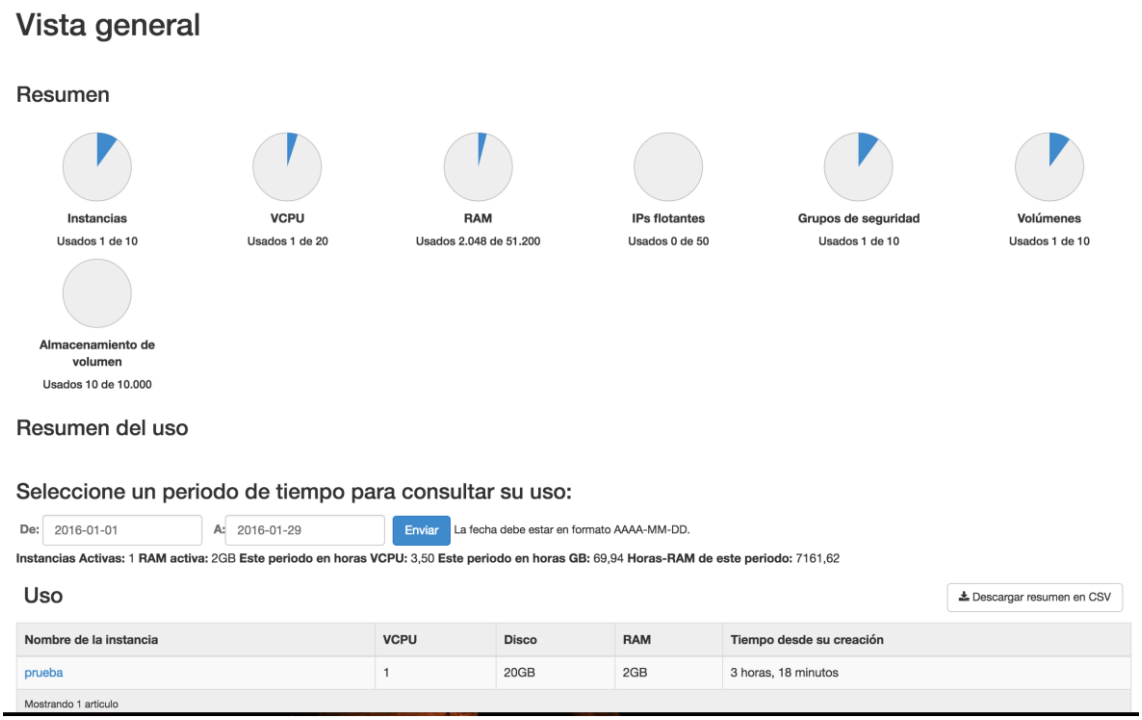
Vamos a analizar las opciones principales del menú del Dashboard. Empezamos por el submenú Proyecto. Al desplegarlo podemos ver las siguientes opciones:



Al realizar la instalación básica, solo disponemos del apartado *Compute* (es importante aclarar que aunque solo aparezca *Compute,* se han instalado más módulos, como

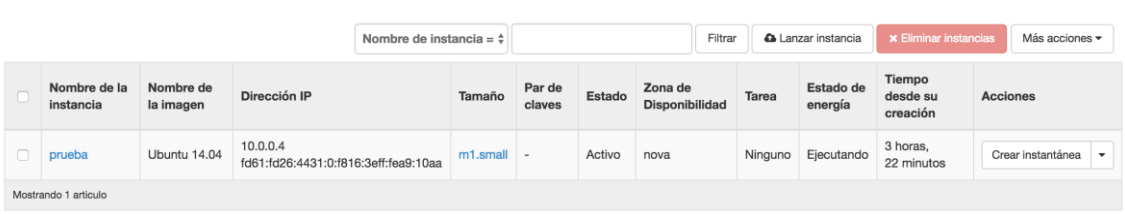
Glance, Switft, Cinder, pero son gestionados directamente desde este mismo apartado, ya que están directamente relacionados con Nova).

En vista general, podremos visualizar de manera gráfica los recursos que están en uso por el proyecto actual.

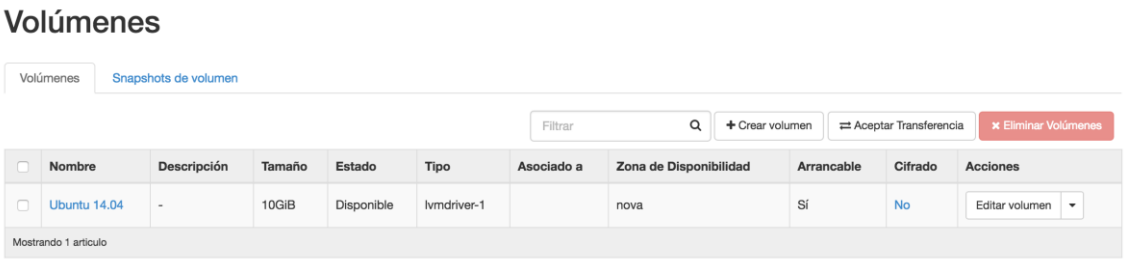


Las limitaciones de recursos son definidas en la configuración del proyecto que tienen plantillas asociadas, con las cuotas máximas.

El apartado instancias es donde podemos gestionar las instancias del proyecto. Es donde debemos dirigirnos si deseamos lanzar una instancia, pararla, eliminarla, hacer una instantánea, etc. Las instancias pueden ser lanzadas a partir de una imagen, de un volumen arrancable o desde una instantánea. Aquí no es donde podemos añadir nuevos tipos de instancias, esa opción la veremos más adelante.

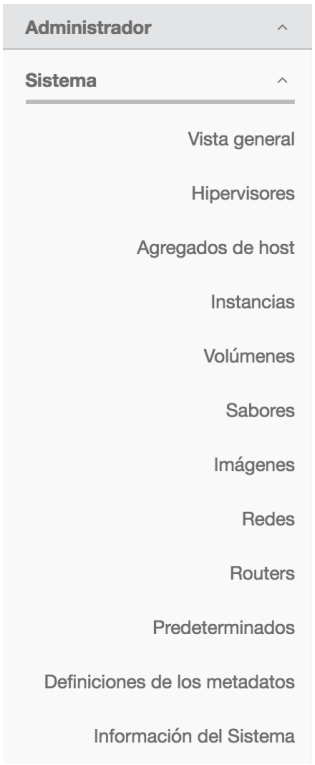


El menú de volúmenes nos permitirá gestionar los espacios de almacenamiento que posteriormente podremos asociar a una instancia. Cada volumen puede asociarse a una única instancia al mismo tiempo, pero puede desasociarse de la instancia y asociarse a una nueva instancia.

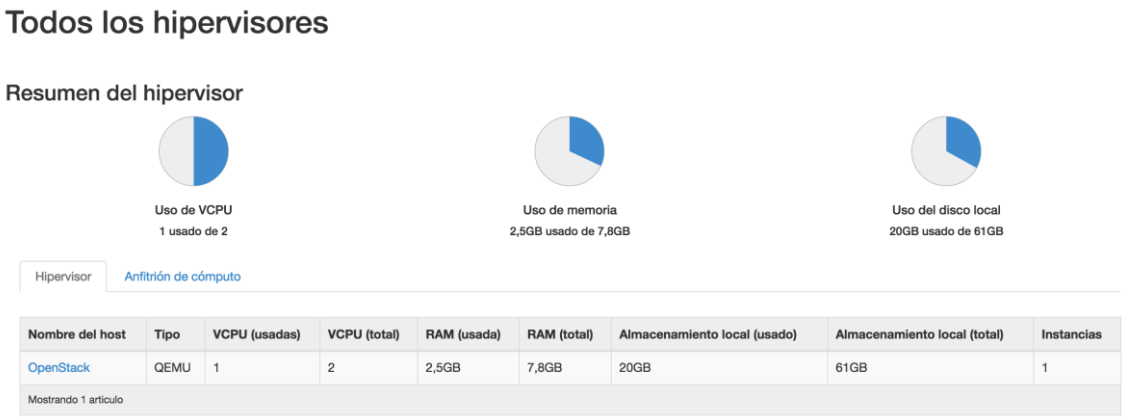


En el apartado de imágenes es donde podremos subir nuestras imágenes de nuevos sistemas que queramos lanzar. Por defecto vienen tres imágenes de prueba de un sistema llamado CirrOS. Este sistema se caracteriza por ser muy ligero, básicamente es para realizar pruebas de virtualización, y tiene unas funcionalidades muy limitadas.

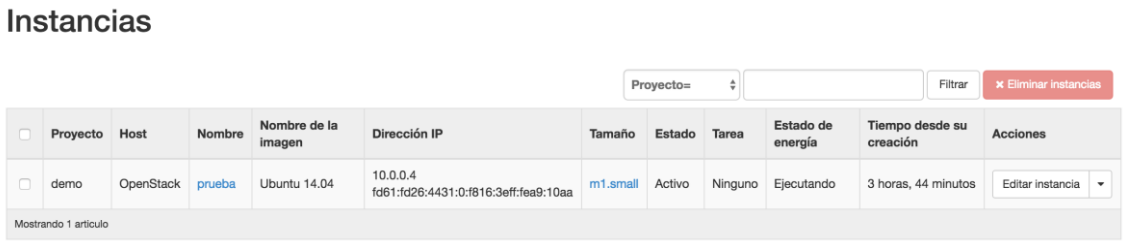
Procedemos a analizar el menú de Administración. Al desplegar veremos las siguientes opciones:

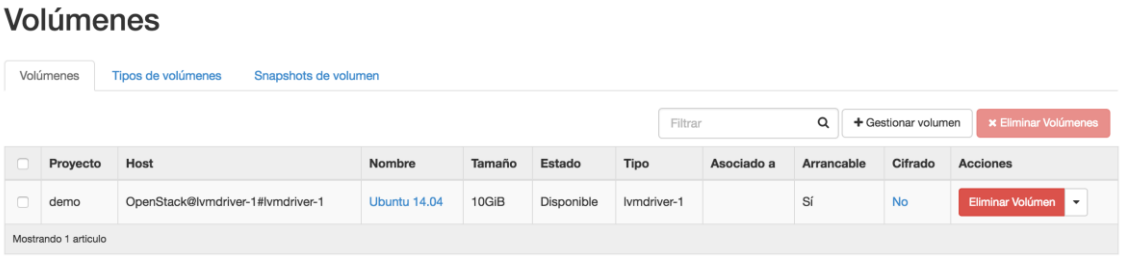


En la opción Hipervisores podemos ver el estado actual del Hipervisor del sistema, que es el encargado de realizar la virtualización de las instancias, proveyendo a las instancias los recursos que necesitan.

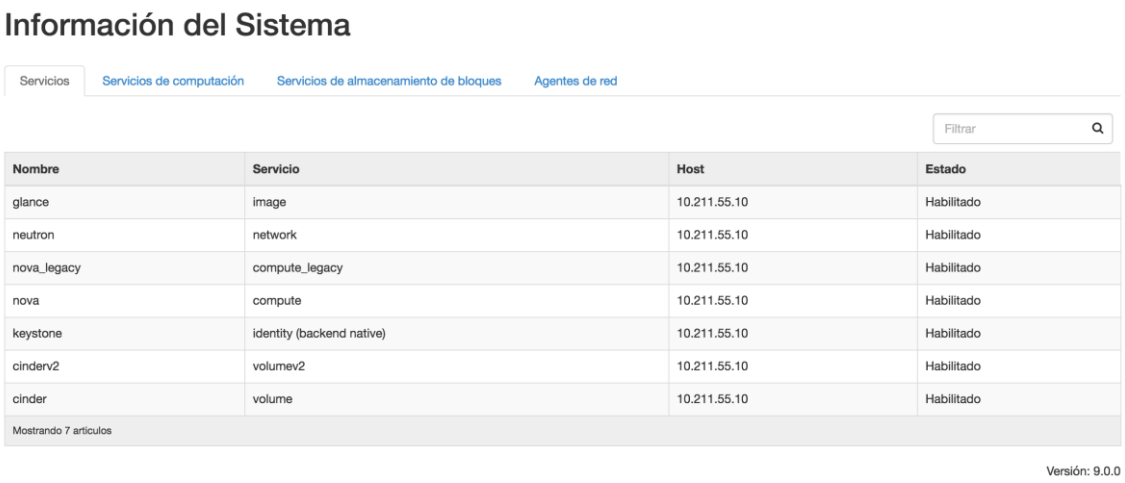


Las opciónes de instancia, volúmenes e imágenes nos permiten gestionar los recursos del sistema. Son similares a la opción del apartado de Computación, con la particularidad de que aquí podemos gestionar las de todo el sistema, independientemente del proyecto.

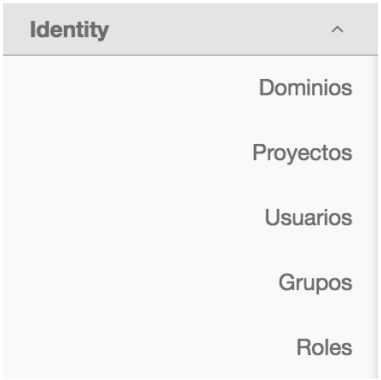




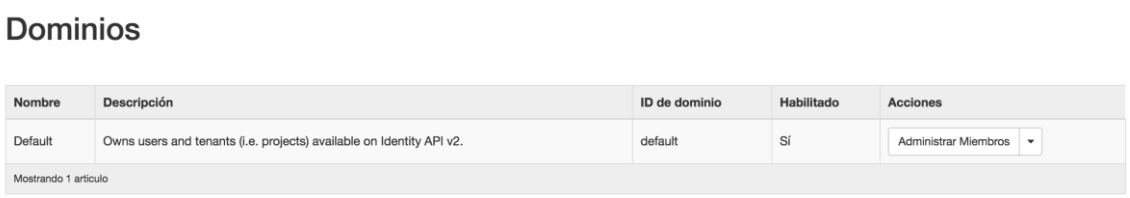
Por último, en la opción de Información del Sistema podremos visualizar los módulos que componen nuestro sistema.



Para finalizar, analizaremos las opciones del submenú Identity, el cual está compuesto por las siguientes opciones:



Aquí básicamente podremos gestionar los usuarios del sistema, agregarlos a grupos, asociarles roles y proyectos. Los dominios engloban tanto proyectos, usuarios, grupos y roles en el sistema, representando las diferentes organizaciones que pueden utilizar nuestro sistema cloud.



En proyectos podremos asignar una serie de recursos y los usuarios que trabajarán con dichos recursos.

Proyectos

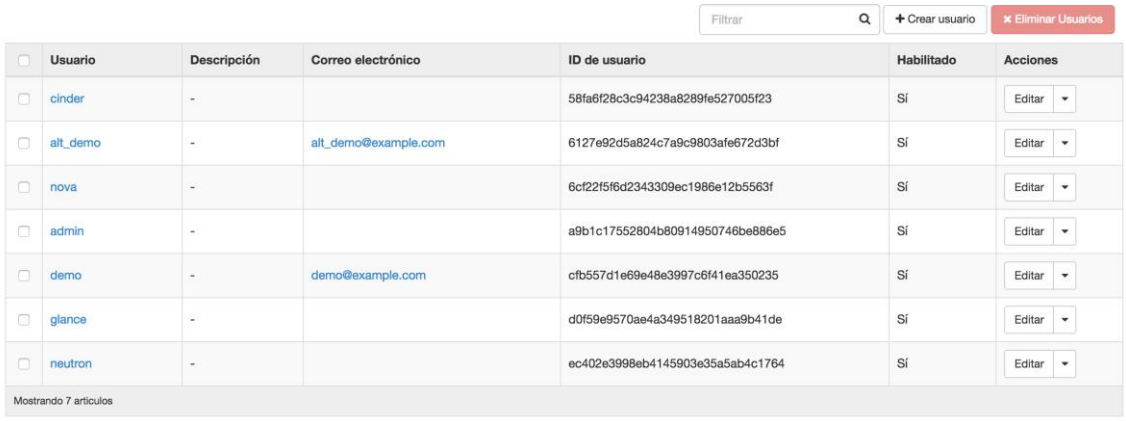
Filtrar | + Crear proyecto | × Eliminar Proyectos

Nombre	Descripción	ID del proyecto	Habilitado	Acciones
alt_demo		11785519db284b06bdc3be45332e877d	Sí	Administrar Miembros
demo		477d7eac65a0444ba2562104ccf9300a	Sí	Administrar Miembros
admin	Bootstrap project for initializing the cloud.	87da69522b374c4a87e0fad38ca331c4	Sí	Administrar Miembros
service		d9ad70c8288942668ea2662b7e299dbd	Sí	Administrar Miembros
invisible_to_admin		e9fd5487b5124391910c4593dab94bc0	Sí	Administrar Miembros

Mostrando 5 artículos

Los usuarios representan a las personas que trabajarán con los recursos de los diferentes proyectos a los que son asignados.

Usuarios



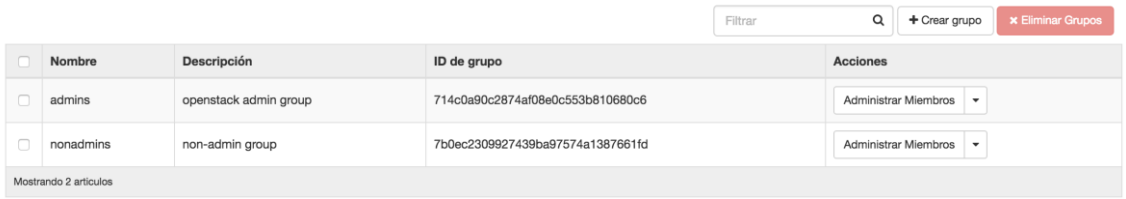
Filtrar | + Crear usuario | × Eliminar Usuarios

Usuario	Descripción	Correo electrónico	ID de usuario	Habilitado	Acciones
cinder	-		58fa6f28c3c94238a8289fe527005f23	Sí	Editar
alt_demo	-	alt_demo@example.com	6127e92d5a824c7a9c9803afe672d3bf	Sí	Editar
nova	-		6cf22f5f6d2343309ec1986e12b5563f	Sí	Editar
admin	-		a9b1c17552804b80914950746be886e5	Sí	Editar
demo	-	demo@example.com	cfb557d1e69e48e3997c6f41ea350235	Sí	Editar
glance	-		d0f59e9570ae4a349518201aaa9b41de	Sí	Editar
neutron	-		ec402e3998eb4145903e35a5ab4c1764	Sí	Editar

Mostrando 7 artículos

Los grupos nos permiten agrupar a los usuarios según características comunes.

Grupos

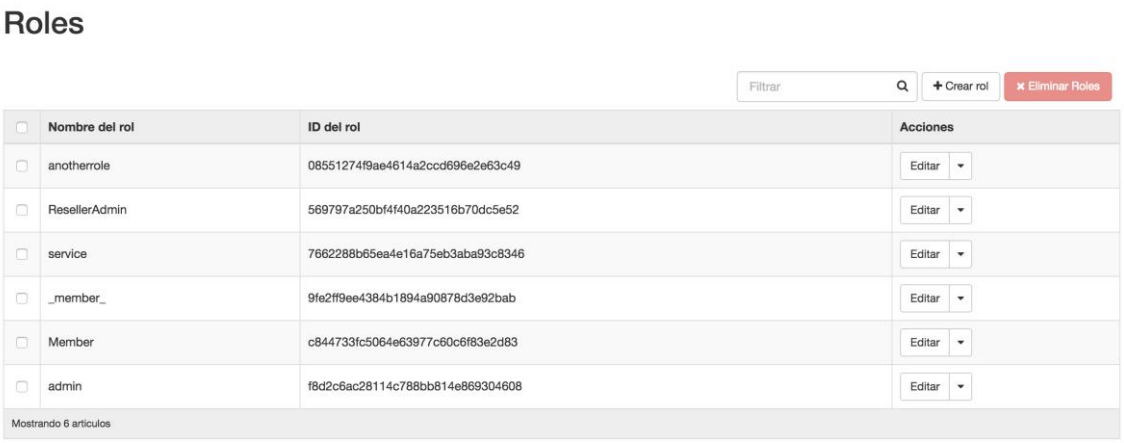


Filtrar | + Crear grupo | × Eliminar Grupos

Nombre	Descripción	ID de grupo	Acciones
admins	openstack admin group	714c0a90c2874af08e0c553b810680c6	Administrar Miembros
nonadmins	non-admin group	7b0ec2309927439ba97574a1387661fd	Administrar Miembros

Mostrando 2 artículos

Los roles permiten definir el objetivo de cada usuario del sistema. Por ejemplo, un usuario puede encargarse de la gestión de los recursos y otro de la creación de nuevas imágenes, por lo que no necesitarán los mismos permisos.



5. OpenStack Avanzado

En los capítulos anteriores hemos visto como es la arquitectura de OpenStack, como funciona y sus orígenes. Una vez tenemos una instalación básica totalmente funcional (que hemos obetenido mediante el capítulo anterior) es interesante explorar las capacidades complementarias del sistema OpenStack.

Vamos a realizar la instalación de los módulos Neutron y Ceilmeter mediante Devstack. Para ello debemos generar un fichero, llamado *local.conf*, en el que indicaremos las modificaciones de instalación que queremos aplicar. El fichero debe ubicarse en el directorio raíz de Devstack. El contenido del mismo será el siguiente:

```
[[local|localrc]]
# Deshabilita el plugin de red de Nova y habilita Neutron
disable_service n-net
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
# Habilita Ceilometer devstack plugin
enable_plugin ceilometer https://git.openstack.org/openstack/ceilometer.git
```

Como se describe en los comentarios, las seis primeras líneas deshabilitan el plugin de red del módulo Nova para que no haya conflictos y habilita Neutron como gestor de red. La última línea nos añade el modulo Ceilometer indicando el repostiorio Git donde se encuentra.

Una vez tenemos nuestro fichero *local.conf* ubicado en el directorio raíz de *devstack*, ejecutaremos la siguiente orden:

```
./stack.sh
```

Deberemos esperar a que finalice la ejecución del script, y una vez finalizado ya tendremos nuestro sistema funcionando con los nuevos módulos. Vamos a crear una nueva imagen de sistema para lanzar una instancia de Ubuntu 14.04 LTS.

El equipo de OpenStack nos recomienda realizar una instalación de máquina virtual con el programa *virt-manager*, que se trata de un programa para crear máquinas virtuales mediante línea de comandos. Al trabajar en un entorno servidor no se dispone de interfaz grafica para realizar la instalación, lo que conllevaría a la necesidad de instalar un gestor de ventanas, por lo que nosotros usaremos otro método que consiste

en crear una máquina virtual y transformarla en formato compatible con OpenStack. Para ello utilizaremos el programa *qemu-img*, que sorporta los siguientes formatos:

qemu-img format strings

Image format	Argument to qemu-img
QCOW2 (KVM, Xen)	`qcow2`
QED (KVM)	`qed`
raw	`raw`
VDI (VirtualBox)	`vdi`
VHD (Hyper-V)	`vpc`
VMDK (VMware)	`vmdk`

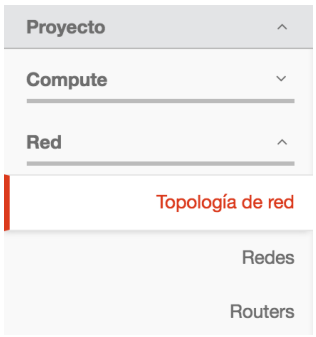
Como el autor del presente trabajo final de grado tiene mayor experiencia con VMware, nosotros utilizaremos el formato VMDK. Para ello debemos realizar una instalación normal del sistema que deseemos, en nuestro caso Ubuntu 14.04 LTS. Una vez instalado, nos dirigimos al directorio donde se almacenan las máquinas virtuales y buscamos el fichero con la extensión *.vmdk. Una vez localizado, debemos ejecutar el siguiente comando:

```
qemu-img convert -f vmdk -O qcow2 image.vmdk image.qcow2
```

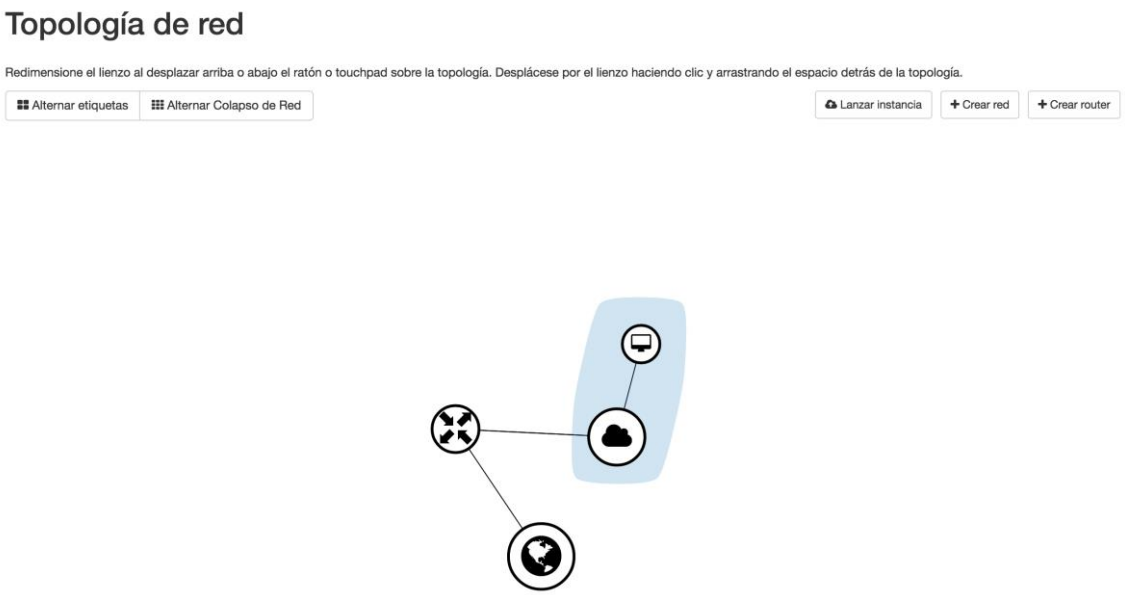
Una vez finalice la ejecución del comando, tendremos un nuevo fichero con extensión *.qcow2, el cual podremos subir a nuestro sistema mediante el menú de Imágenes en el apartado Compute.

Vamos a realizar un pequeño análisis de los módulos añadidos mediante el fichero *local.conf*.

El menú de configuración de Neutron se encuentra en el apartado proyecto, como podemos observar en la imagen:

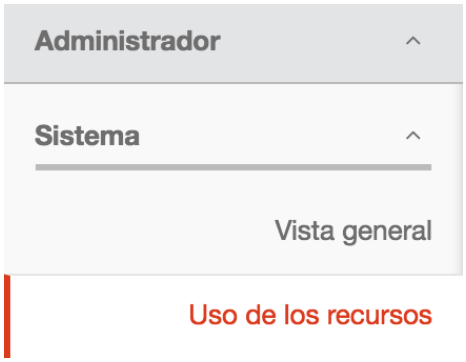


Aquí podemos realizar el diseño de nuestra arquitectura de red virtual, en la que interconectaremos las diferentes instancias que sean ejecutadas en el sistema. En la opción Topología de red tenemos una representación visual de la red actual del sistema. En nuestro caso disponemos del sistema OpenStack, representado por una nube, una instancia, que es representada por un ordenador, un router virtual, representado por el icono de un router (circulo con flechas en su interior) y por último la conexión del host anfitrión, representada por un globo terráqueo, que representa Internet. Pulsando en Topología de red, nos aparecerá una pantalla similar a esta:

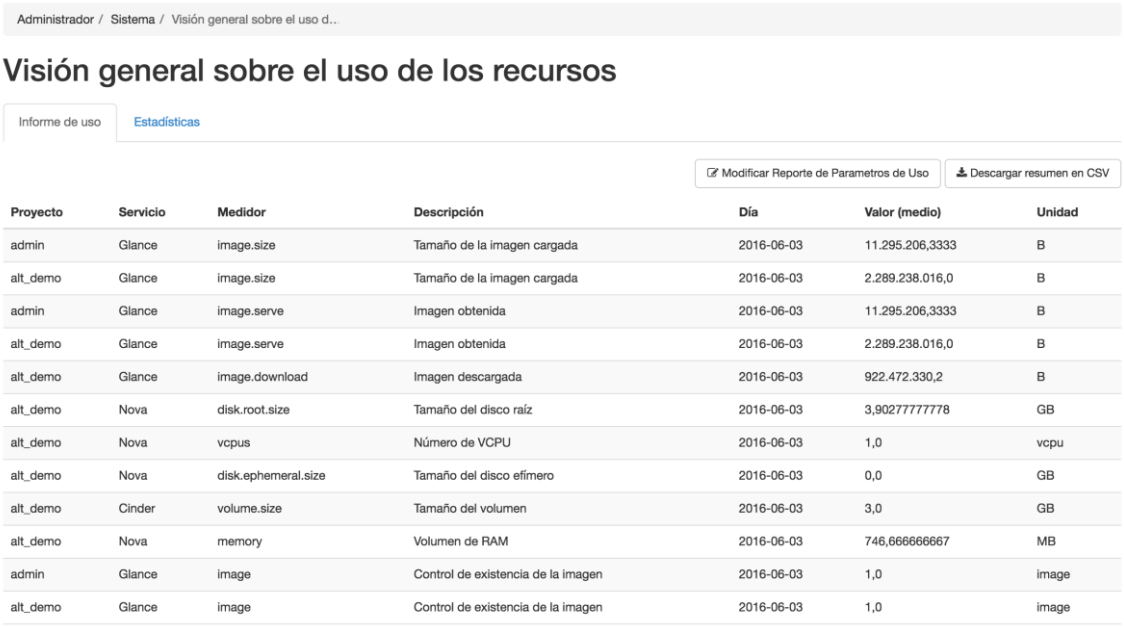


Si hacemos click en cualquiera de los elementos podremos acceder a su menú de configuración. En la parte superior podemos observar que se encuentran las opciones para agregar redes y routers virtuales. Cada nueva instancia que se lance en el sistema aparecerá adherido a la nube. Aquí también podremos crear nuevas VLANs, asociar las instancias a estas VLANs, interponer routers entre diferentes redes de nodos, etc.

Por otra parte dado que hemos instalado el módulo Ceilometer encargado de la medida de uso del sistema vamos a proceder a su análisis. Para acceder a la información de uso debemos acceder al siguiente menú:



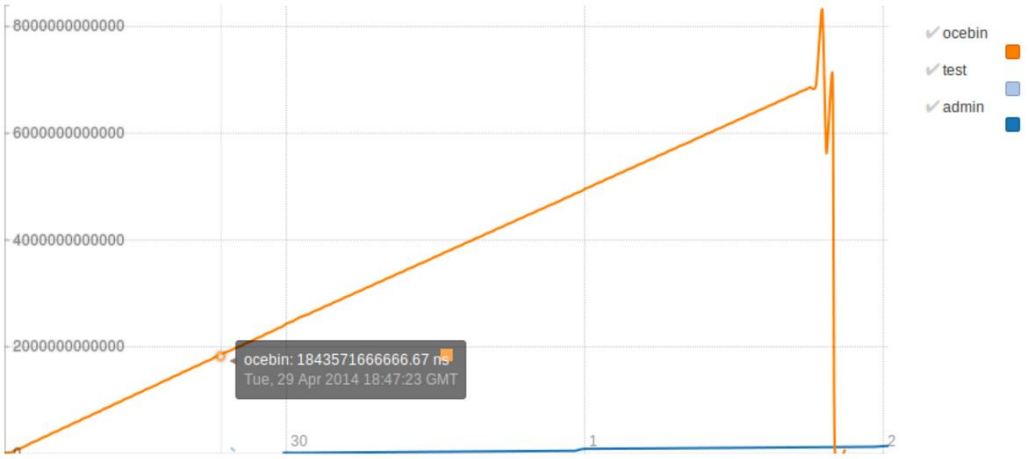
Una vez aquí, en la pestaña de Informe de uso nos aparecerán todos los elementos que han requerido recursos del sistema para su ejecución, el proyecto al que pertenecen, que tipo de recurso han consumido, etc. Uno de los ejemplos principales de uso son las instancias que lanzamos, ya que éstas necesitan una VCPU, memoria, almacenamiento, etc.



Proyecto	Servicio	Medidor	Descripción	Día	Valor (medio)	Unidad
admin	Glance	image.size	Tamaño de la imagen cargada	2016-06-03	11.295.206,3333	B
alt_demo	Glance	image.size	Tamaño de la imagen cargada	2016-06-03	2.289.238.016,0	B
admin	Glance	image.serve	Imagen obtenida	2016-06-03	11.295.206,3333	B
alt_demo	Glance	image.serve	Imagen obtenida	2016-06-03	2.289.238.016,0	B
alt_demo	Glance	image.download	Imagen descargada	2016-06-03	922.472.330,2	B
alt_demo	Nova	disk.root.size	Tamaño del disco raíz	2016-06-03	3,90277777778	GB
alt_demo	Nova	vcpus	Número de VCPU	2016-06-03	1,0	vcpu
alt_demo	Nova	disk.ephemeral.size	Tamaño del disco efímero	2016-06-03	0,0	GB
alt_demo	Cinder	volume.size	Tamaño del volumen	2016-06-03	3,0	GB
alt_demo	Nova	memory	Volumen de RAM	2016-06-03	746,666666667	MB
admin	Glance	image	Control de existencia de la imagen	2016-06-03	1,0	image
alt_demo	Glance	image	Control de existencia de la imagen	2016-06-03	1,0	image

Por otro lado, en la pestaña de Estadísticas podremos ver la información de manera gráfica. Nos permite seleccionar qué tipo de recurso queremos visualizar, el periodo de tiempo que queremos comprobar, etc.

En nuestro caso, al tener una utilización tan corta de los recursos, solo vemos una pequeña muestra de la información recogida. Veríamos algo similar a esto:



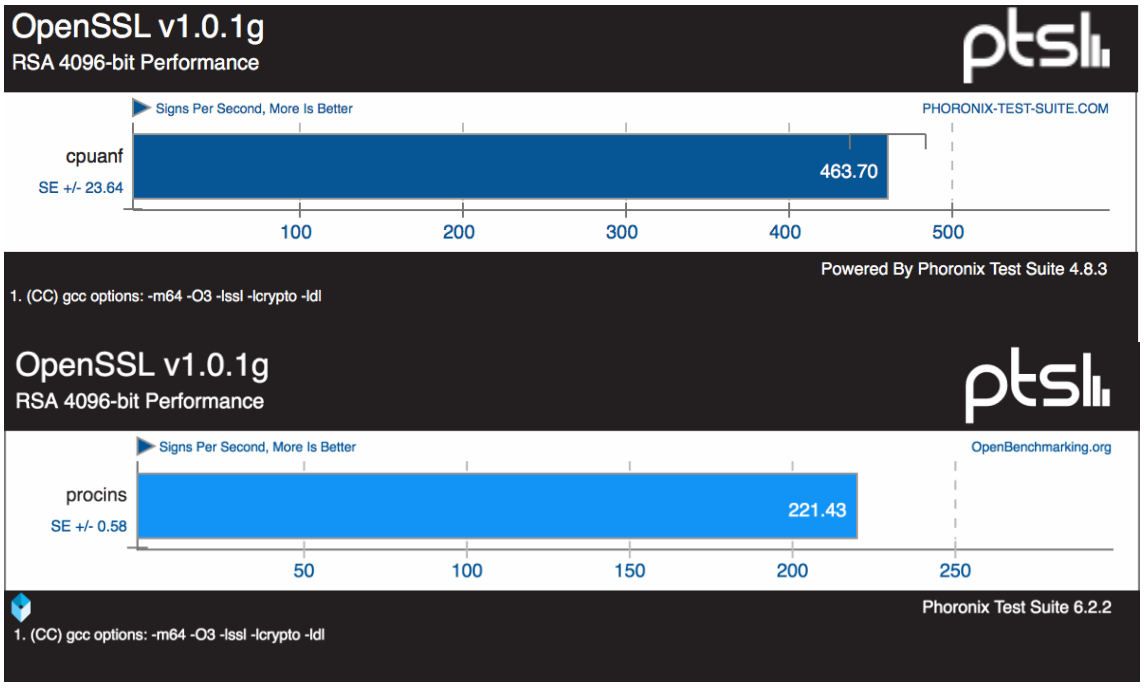
El eje X representa la línea de tiempo y el eje Y representa el consumo medio en nanosegundos de la CPU. Los nombres de la derecha representan los diferentes proyectos.

6. Análisis de rendimiento

Una vez que tenemos todo el sistema implantado y funcionando, es interesante conocer qué rendimiento nos ofrece el sistema virtualizado con respecto al equipo original, y la sobrecarga que conlleva la virtualización. Vamos a realizar una serie de pruebas de rendimiento de los componentes principales que son CPU, memoria RAM, red y disco.

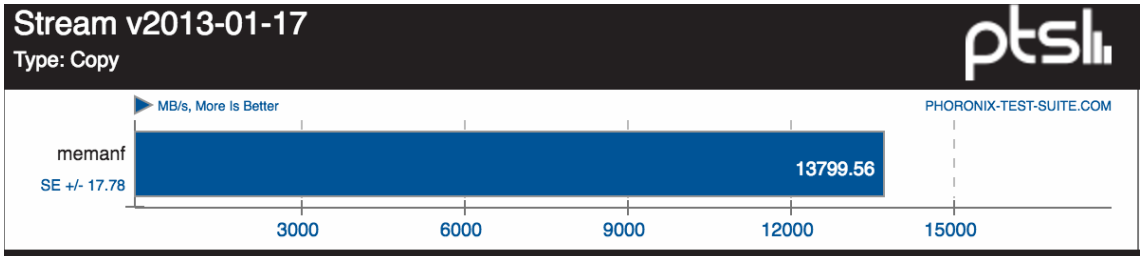
Tras realizar un pequeño estudio de los benchmark que hay disponibles para Ubuntu, hemos seleccionado la herramienta *Phoronix Test Suite*, ya que es muy completa y dispone de un gran numero de tests de todo tipo. Nosotros realizaremos tests en nuestra máquina anfitrión y en una instancia. El hardware de la máquina anfitrión ha sido comentado en el tercer capítulo de Instalación básica de OpenStack. Por otro lado, la instancia dispone de un procesador Intel(R) Core(TM) i7-4870HQ CPU @ 2.50GHz (solo tiene acceso a dos núcleos), ocho gigabytes de RAM, cuarenta gigabytes de disco duro SSD y una tarjeta de red ethernet a diez megabits/segundo.

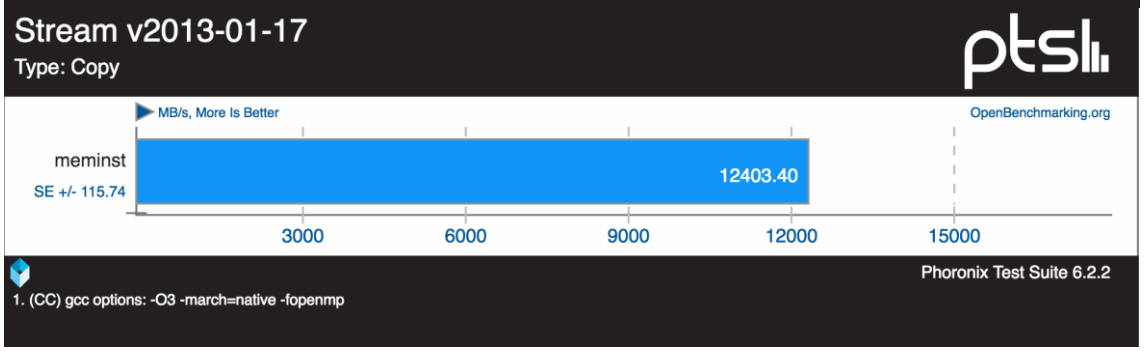
Primero ejecutamos el test de la CPU, el cual mediante el OpenSSL medirá el rendimiento de nuestro procesador tanto en el equipo anfitrión como en el instanciado. Los resultados obtenidos son estos:

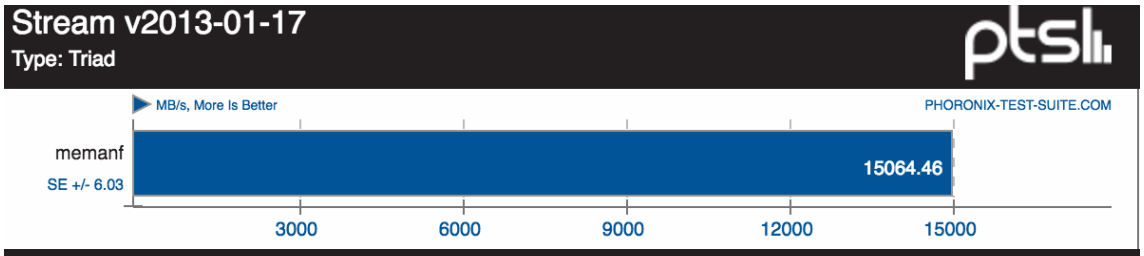


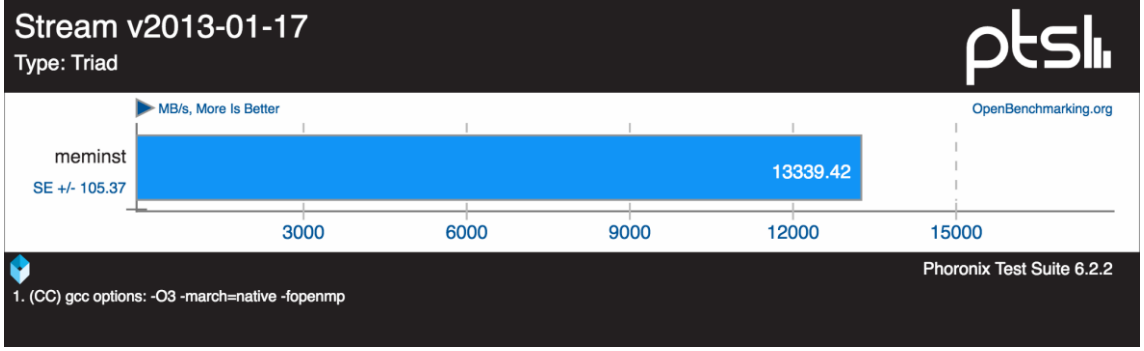
El test realizado a *cpuanf* (el del equipo anfitrión), muestra que el rendimiento del equipo sin virtualizar (*procins*), es muy superior en comparación a la instancia. Esto es debido a que el equipo instanciado solo dispone de dos núcleos mientras que el anfitrión dispone de cuatro, por lo que justifica la mitad del rendimiento.

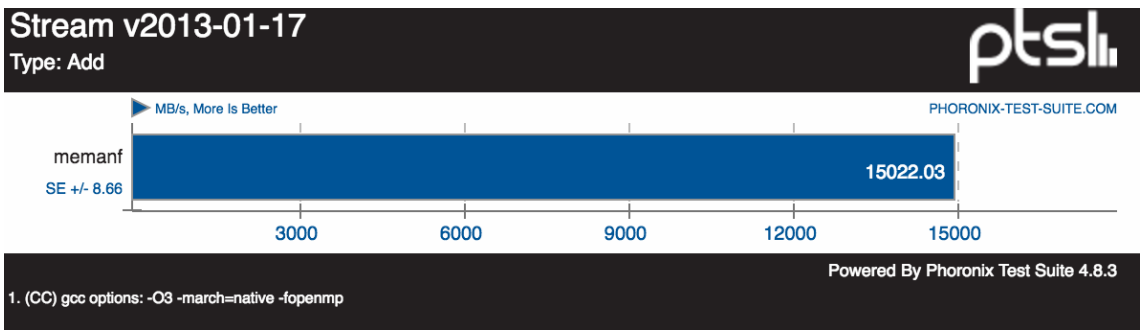
Realizamos una prueba de rendimiento a la memoria del sistema. La prueba consiste en la lectura, escritura y escalado. Hemos obtenidos los siguientes resultados:

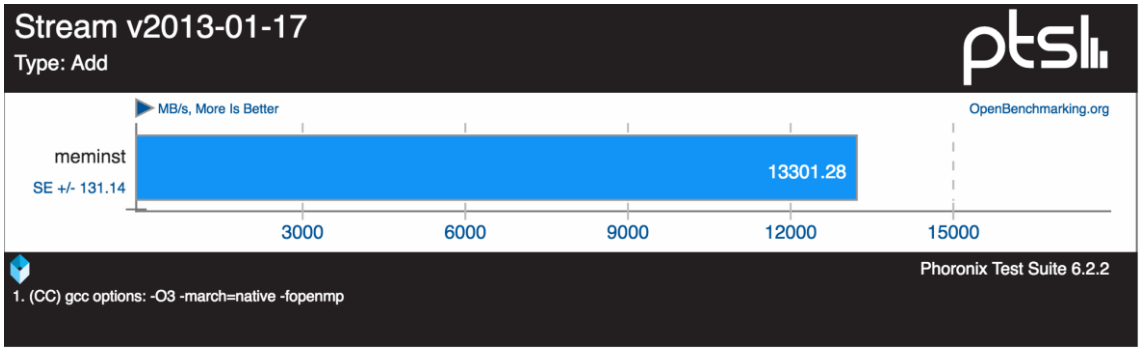

Stream v2013-01-17
Type: Copy
MB/s, More Is Better
PHORONIX-TEST-SUITE.COM
memanf
SE +/- 17.78
13799.56
3000
6000
9000
12000
15000
Powered By Phoronix Test Suite 4.8.3
1. (CC) gcc options: -O3 -march=native -fopenmp


Stream v2013-01-17
Type: Copy
MB/s, More Is Better
OpenBenchmarking.org
meminst
SE +/- 115.74
12403.40
3000
6000
9000
12000
15000
Phoronix Test Suite 6.2.2
1. (CC) gcc options: -O3 -march=native -fopenmp


Stream v2013-01-17
Type: Triad
MB/s, More Is Better
PHORONIX-TEST-SUITE.COM
memanf
SE +/- 6.03
15064.46
3000
6000
9000
12000
15000
Powered By Phoronix Test Suite 4.8.3
1. (CC) gcc options: -O3 -march=native -fopenmp

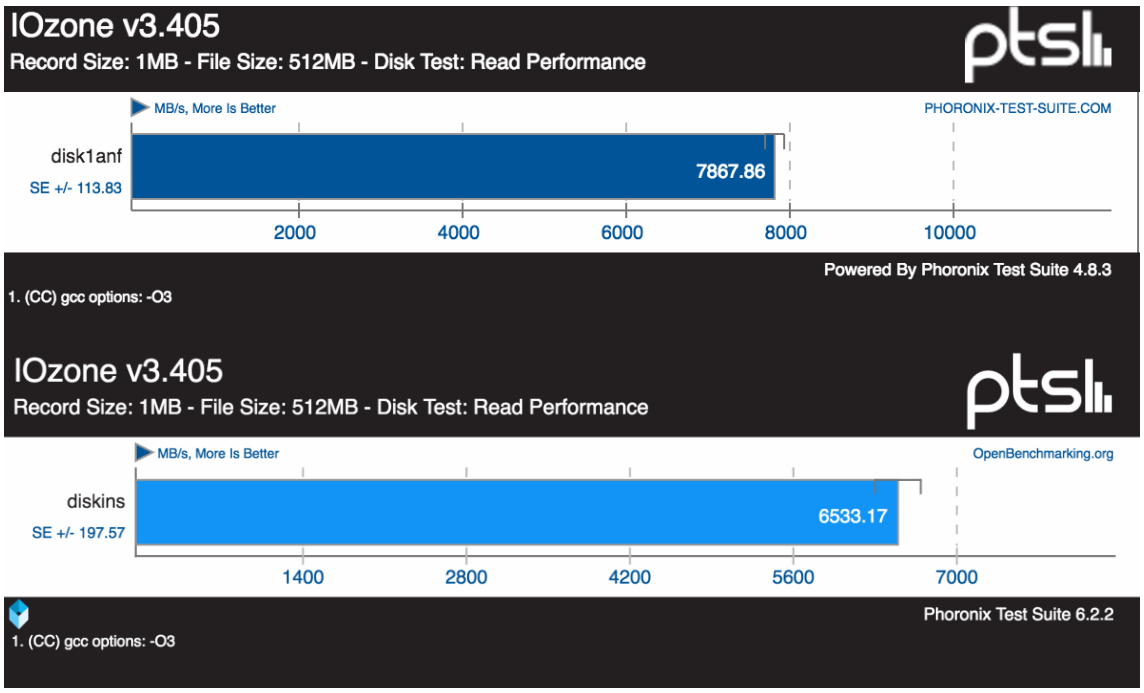

Stream v2013-01-17
Type: Triad
MB/s, More Is Better
OpenBenchmarking.org
meminst
SE +/- 105.37
13339.42
3000
6000
9000
12000
15000
Phoronix Test Suite 6.2.2
1. (CC) gcc options: -O3 -march=native -fopenmp

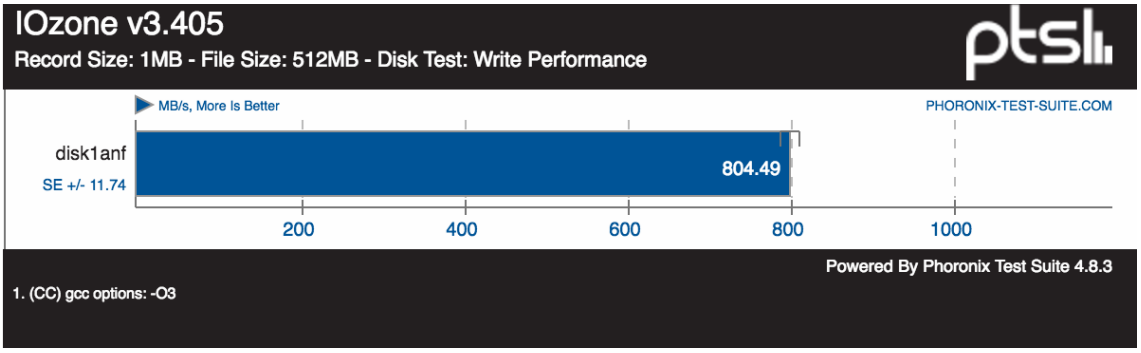

Stream v2013-01-17
Type: Add
MB/s, More Is Better
PHORONIX-TEST-SUITE.COM
memanf
SE +/- 8.66
15022.03
3000
6000
9000
12000
15000
Powered By Phoronix Test Suite 4.8.3
1. (CC) gcc options: -O3 -march=native -fopenmp

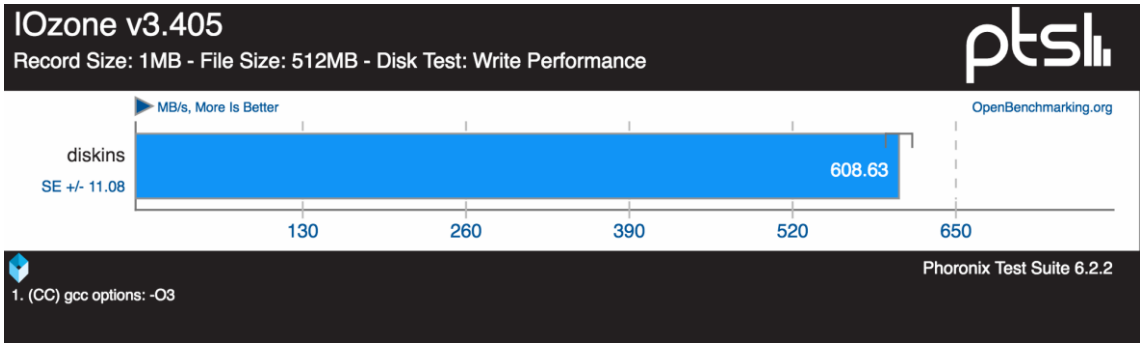


Como podemos observar en los resultados, la diferencia del test en el anfitrión (*memanf*) y en la instancia (*meminst*) no es tan elevada (en el caso anterior el rendimiento era de casi el doble de rápido en el anfitrión).

Realizamos una prueba de rendimiento en el disco, en la que realizamos pruebas de lectura y escritura. Los resultados obtenidos son los siguientes:

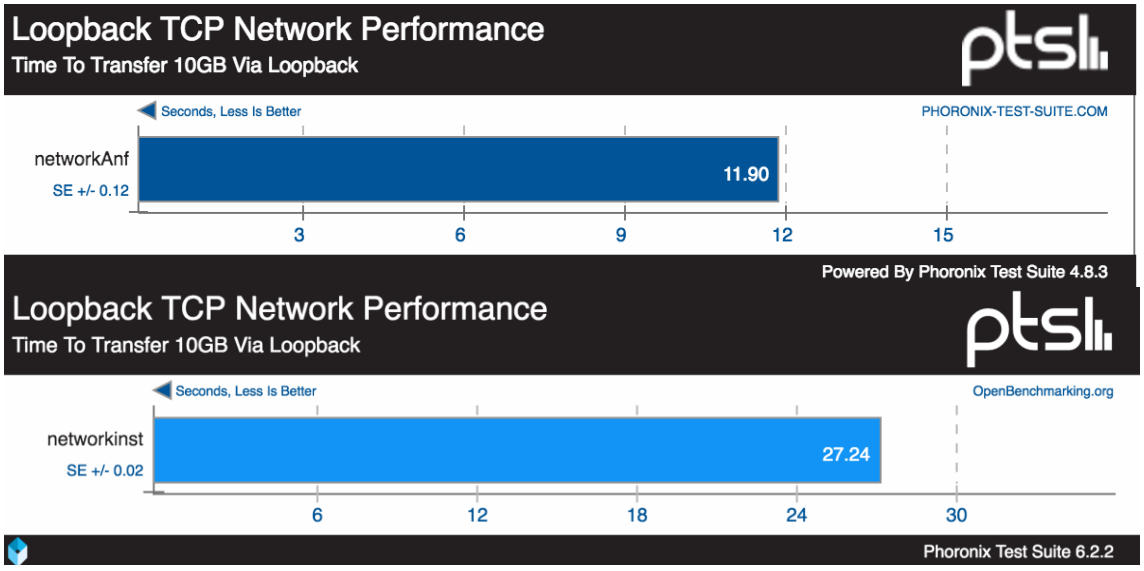






Como en el caso de la memoria, no existe una diferencia tan abultada como en el procesador. Como era de esperar, el rendimiento del anfitrión (*disk1anf*), es ligeramente superior comparado con el de la instancia (*diskins*).

Por último realizamos pruebas de red, que consiste en transferir un fichero de 10GB mediante loopback. Los resultados obtenidos son los siguientes:



En este caso, *networkinst* (el equipo instanciado), tiene un rendimiento bastante inferior al del anfitrión, ya que viendo los resultados, tarda casi tres veces más en realizar la misma transferencia de datos. Esto se debe a que la prueba se realiza sobre una interfaz virtual, la cual el sistema anfitrión debe realizar todas las operaciones que haría la electrónica de la tarjeta de red, por lo que al no ser un componente prioritario, la virtualización le da menos prioridad.

Analizando estos resultados podemos decir que aunque los equipos instanciados ofrecen un rendimiento óptimo, si que ofrecen una sobrecarga notable en el rendimiento de los componentes en comparación con el equipo anfitrión. Por otra parte hay que tener en cuenta que las instancias tienen un hardware acotado, es decir, no disponen de todo el hardware disponible en el equipo, aunque esto no sea tan relevante en términos de rendimiento, excepto en el caso del procesador y la tarjeta de red.

7. Conclusiones

Como resultado de este trabajo final de grado hemos obtenido un sistema cloud totalmente funcional OpenStack instalado mediante el script Devstack. Hemos profundizado en su funcionamiento, arquitectura y el potencial que tienen este tipo de sistemas. Se ha analizado el rendimiento, las funcionalidades básicas que ofrece y algunas de las avanzadas.

Vamos a realizar un pequeño resumen de los contratiempos que nos hemos encontrado durante el desarrollo de este trabajo y como los hemos conseguido solucionar. Finalmente comentaremos las consideraciones finales.

Al trabajar con Devstack, hemos sufrido varios cambios de versión de OpenStack (como se comento anteriormente, Devstack usa siempre la versión más reciente de OpenStack). Por lo que al realizar algunas tareas, surgían cambios en interfaz y en los menús. Por suerte, con la ayuda de la documentación de OpenStack, hemos podido solucionarlos. Cabe destacar, que los cambios de versión tan frecuentes solo se dan con la instalación mediante Devstack, ya que las instalaciones tradicionales utilizan siempre la ultima versión estable (como es el caso de muchos sistemas software, por ejemplo la distribución Linux Ubuntu, que dispone de versiones *Long Time Support* o LTS y versiones normales).

En las últimas versiones, el equipo de desarrollo de Devstack decidió eliminar un script llamado *rejoin_stack.sh*, con el que junto al script *unstack.sh*, nos brindaba persistencia después de reiniciar el equipo anfitrión, es decir, perdemos todas nuestras configuraciones, instancias, imágenes, etc. Actualmente el script *rejoin_stack.sh* no está disponible por problemas de consistencia del sistema (si daba error el sistema podía quedar corrompido). Como única solución a la pérdida de persistencia, tuvimos que realizar todas las pruebas seguidas, ya que si no debíamos empezar de cero (ejecutando el script *stack.sh*).

Por otro lado, en cuanto a las instancias que lanzábamos, no tenían acceso a Internet, aun teniendo correctamente configurada la parte de la red interna y externa. Tenían conexión entre instancias y con el anfitrión pero no podían conectar con el exterior. Realizando una larga búsqueda, encontramos que el problema reside en que el anfitrión tiene que redirigir el trafico de las instancias hacia el exterior, por lo que con esta entrada en las *iptables*, conseguimos solucionarlo.

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Por último tuvimos problemas al lanzar instancias en el sistema. El problema que nos mostraba OpenStack decía que no disponía de ningún host en el que albergar la instancia. Realizando una pequeña investigación, nos dimos cuenta de que no teníamos suficiente espacio en disco. Para solucionarlo tuvimos que redimensionar la partición

de Ubuntu en el ordenador anfitrión a cuarenta gigabytes (inicialmente tenia veinte gigabytes).

Como consideraciones finales cabe decir que OpenStack es una herramienta muy potente, en la que se pueden basar grandes sistemas. Por otro lado, no está más extendido debido a la complejidad de su implantación.

No hemos podido realizar pruebas en un entorno con varios servidores, por cuestiones de disponibilidad de recursos, ya que la universidad sí que podía ofrecer algunos nodos de sus servidores, pero al tener que trabajar en remoto, como se comentó anteriormente, es muy tedioso, ya que al realizar las configuraciones de red es posible perder la conexión y necesitar acceder directamente al equipo. Tampoco ha sido posible analizar módulos también muy interesantes como sería Ironic (Bare-metal privisioning), que consiste en realizar virtualización a más bajo nivel (directamente del hardware) por motivo de duración del trabajo. Por último, también habría sido interesante realizar un montaje con un objetivo explícito, es decir, montaje similar a los que se comentaban anteriormente, como el que usa eBay, PayPal o la NBC.

Finalmente destacar la gran salida laboral que ofrece este sector, ya que cada vez más las empresas, y nosotros mismos como usuarios, estamos utilizando más este tipo de plataformas, véase Dropbox, Youtube, Google Docs y un largo etcétera, por lo que la demanda de este tipo de especialistas seguirá creciendo durante los próximos años.

8. Bibliografía

1. UN VISTAZO RÁPIDO A LA HISTORIA DE OPENSTACK - http://nubeblog.com/2011/01/10/un-vistazo-rapido-a-la-historia-de-openstack/ - 03/02/2016

2. ¿Qué es PaaS? - http://www.interoute.es/what-paas - 03/02/2016

3. ¿Qué es IaaS? - http://www.interoute.es/what-iaas - 03/02/2016

4. ¿Qué es SaaS? - http://www.interoute.es/what-saas - 03/02/2016

5. Computación en la nube - https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube - 03/02/2016

6. Plataformas cloud: ¿open source o propietarias? - http://www.revistacloudcomputing.com/2013/06/plataformas-cloud-open-source-o-propietarias/ - 15/04/2016

7. Comparativa de las plataformas cloud abiertas: OpenStack, OpenNebula, Eucalyptus y CloudStack - http://www.revistacloudcomputing.com/2013/01/comparativa-de-las-plataformas-cloud-abiertas-openstack-opennebula-eucalyptus-y-cloudstack/ - 15/04/2016

8. 10 Empresas de Referencia que usan OpenStack - https://openwebinars.net/10-empresas-de-referencia-que-usan-openstack/ - 15/04/2016

9. Advanced Message Queuing Protocol - https://es.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol - 03/05/2016

10. RabbitMQ - https://www.rabbitmq.com/features.html - 03/05/2016

11. What is OpenStack? - http://www.openstack.org/software/ - 03/05/2016 (preguntar si hay que poner todas las paginas de los modulos)

12. Building a Hadoop cluster in an OpenStack environment with Sahara - https://www.youtube.com/watch?v=MAaSiMLE05A - 03/05/2016

13. Sample Configurations - http://www.openstack.org/software/sample-configs/ - 04/05/2016

14. Laboratorios virtuales: una solución con infraestructura de virtualización – Antoni Salmerón Bermúdez – 27/09/2013

15. OpenStack API Guide - http://developer.openstack.org/api-guide/quick-start/ - 05/05/2016

16. Development resources for OpenStack clouds - http://developer.openstack.org/ - 05/05/2016

17. How To Get Started With OpenStack - http://www.openstack.org/software/start/ - 05/05/2016

18. OpenStack Installation Guide for Ubuntu - http://docs.openstack.org/mitaka/install-guide-ubuntu/ - 05/05/2016

19. Devstack - http://docs.openstack.org/developer/devstack/ - 05/05/2016

20. OpenStack Installation Guide for Ubuntu - http://docs.openstack.org/mitaka/install-guide-ubuntu/ - 25/05/2016

21. All-In-One Single Machine - http://docs.openstack.org/developer/devstack/guides/single-machine.html - 25/05/2016

22. OpensTack Documentation - Overview - http://docs.openstack.org/mitaka/install-guide-ubuntu/overview.html#example-architecture - 25/05/2016

23. Packstack quickstart: Proof of concept for single node - https://www.rdoproject.org/install/quickstart/ - 25/06/2016

24. Devstack Overview - http://docs.openstack.org/developer/devstack/overview.html - 31/05/2016

25. Using DevStack with neutron Networking - http://docs.openstack.org/developer/devstack/guides/neutron.html - 31/05/2016