Universidad Nacional de La Plata

Facultad de Informática

TRABAJO FINAL INTEGRADOR

ESPECIALIZACIÓN EN REDES Y SEGURIDAD

TITULO:

ANÁLISIS DE UNA INFRAESTRUCTURA CLOUD OPEN SOURCE

ALUMNO: MARIA A. MURAZZO

DIRECTOR: DR. FERNANDO G. TINETTI (UNLP)

CO DIRECTOR: LIC. NELSON R. RODRÍGUEZ (UNSJ)

AÑO 2015

Índice de Contenidos

1. INTRODUCCIÓN

1.1. CONTENIDO DE ESTE TRABAJO

El presente trabajo aborda el análisis de una infraestructura de cloud privada y open source. Para lograr esto se comienza con un breve estudio sobre lo que es el cloud computing, sus características, modelos y despliegues. A continuación se pone especial énfasis en la infraestructura del cloud y la virtualización, la cual es el fundamento para montar una infraestructura.

Se hace una breve comparación de los productos disponibles para montar un cloud privado y open source: Eucalyptus, OpenNebula, CloudStack y OpenStack. Esta comparación tiene por objeto determinar cuál es el producto más adecuado para montar la infraestructura.

Una vez seleccionado el producto, se realiza un estudio minucioso de su arquitectura y componentes. Por último se realiza la instalación y puesta a punto de una infraestructura cloud privada de prueba.

1.2. ESTRUCTURA GENERAL DEL TRABAJO

El presente trabajo está organizado en siete capítulos. En el Capitulo 1, se muestra la motivación en realizar este trabajo y los objetivos que se persiguen.

En el Capítulo 2, se desarrollan los conceptos básicos, que constituyen los marcos generales de referencia que sustentan esta investigación. Se realiza una introducción al Cloud Computing, sus modelos de servicio y de despliegue, sus ventajas y desventajas y la arquitectura en capas que posee. Por último se hace especial énfasis en la capa de la infraestructura, la cual es el objeto de estudio del presente trabajo.

En el Capítulo 3, se realiza un análisis de los cuatro productos que ofrece el mercado de tipo open source: Eucalyptus, OpenNebula, CloudStack y OpenStack. También se realiza un análisis de las ventajas y desventajas de cada producto confeccionando un cuadro comparativo. Además, se realiza un análisis en función de las comunidades, tomando como base el numero mensual de temas, mensajes, participantes y accesos al repositorio de descarga (git); para concluir con la justificación de seleccionar a OpenStack como infraestructura a instalar.

En el Capitulo 4, se realiza un análisis de la infraestructura OpenStack, haciendo hincapié en su evolución, arquitectura, características, versionado, componentes, terminología, administración de la virtualización y topologías de instalación. También se hace referencia a herramientas desarrolladas por la comunidad que sirven para ayudar a la configuración, gestión y administración de OpenStack.

En el Capitulo 5, se muestra el proceso de instalación de OpenStack con una topología monolítica y usando como herramienta de gestión de instalación a DevStack. En el Capitulo 6, se realiza una prueba de la infraestructura mediante el uso de una instancia para correr un algoritmo de multiplicación de matrices.

Por último en el Capitulo 7 se brindan las conclusiones a las que se ha llegado con la instalación de la infraestructura y, las futuras ampliaciones que se podrían realizar.

1.3. MOTIVACIÓN

En los últimos años se ha visto evolucionar tecnologías vitales para el mundo empresarial en lo que a TIC (Tecnologías de Información y Comunicaciones) se refiere, tales como los servicios de telefonía, las telecomunicaciones, los datacenter, etc.

Así que la pregunta es, ¿por qué no conectarse a Internet para que alguien suministre todos los servicios de computación que la empresa necesita de manera simple y, se facture

mensualmente por ello?, de esta forma todo lo que sea computación se convierte en un servicio más. Esa idea no es nueva, se viene trabajando en este concepto desde hace algunos años, conceptos precursores son *utility computing*, *computación bajo demanda*, *computación elástica* o *grid computing*.

Internet usualmente se visualiza y conceptualiza como una gran nube (cloud) donde todo está conectado y donde al conectarse se suministran todos los servicios requeridos. A este esquema de trabajo se lo denomina Cloud Computing o Computación en Nube, la cual es similar a todos los esquemas antes nombrados, pero potenciada con las tecnologías de virtualización [1].

La característica básica de este modelo es que los recursos y servicios informáticos, tales como infraestructura, plataforma y aplicaciones, son ofrecidos y consumidos como servicios a través de Internet sin que los usuarios tengan que tener ningún conocimiento de lo que sucede detrás. El Cloud Computing es un esquema del tipo *aaS* o *as a Service* y que a veces se expresa como *XaaS* o *EaaS* para significar *Everything as a Service*; en general, cualquier cosa como un servicio [2].

Cloud computing ha ganado popularidad debido a que confronta el problema de muchas empresas tradicionales con respecto a la computación, en la que las empresa dedican tiempo y dinero en implementar una infraestructura, ya sea con personal o en un data center. El modelo tradicional de computación por lo general conduce a la subutilización de los recursos de TI, debido a la fragmentación de recursos y la distribución desigual de la carga de trabajo; mientras que cloud computing permite a las organizaciones virtualizar su infraestructura de TI, y abordar cualquier tipo de problema que esto conlleve.

1.4. OBJETIVO

El objetivo del presente Trabajo Integrador es el análisis de una Infraestructura para Cloud Computing privada, basada en tecnologías Open Source. Para lograr este objetivo se deberá realizar un análisis de los productos existentes en el mercado con la finalidad de determinar cuál es el más adecuado. Esta selección se realizará en base a parámetros previamente fijados por la comunidad académica tales como: soporte de documentación, versionado, comunidades de ayuda (foros, listas, etc.).

Una vez realizada la selección de la infraestructura a instalar, se deberá estudiar detalladamente su arquitectura, componentes, topologías, etc. Con el objetivo de realizar de la mejor forma posible la tarea de instalación, configuración y puesta a punto del cloud.

2. MARCO TEÓRICO

2.1. ¿QUÉ ES EL CLOUD COMPUTING?

Según IBM [3], el Cloud Computing es un modelo de aprovisionamiento rápido de recursos IT (Informatiob Technology) que potencia la prestación de servicios, facilitando la operativa del usuario final y del prestador del servicio. Además, todo ello se realiza de manera fiable y segura, con una escalabilidad elástica que es capaz de atender fuertes cambios en la demanda no previsibles a priori, sin que esto suponga apenas un incremento en los costes de gestión.

Otra definición complementaria es la aportada por el RAD Lab de la Universidad de Berkeley, donde se explica que el cloud computing se refiere tanto a las aplicaciones entregadas como servicio a través de Internet, como el hardware y el software de los centros de datos que proporcionan estos servicios.

El NIST (National Institute of Standards and Technology) [4], ha presentado una de las definiciones de Cloud más clara y comprensible. La define como un *modelo tecnológico que permite el acceso ubicuo, adaptado y bajo demanda en red a un conjunto compartido de recursos de computación configurables (por ejemplo: redes, servidores, equipos de almacenamiento, aplicaciones y servicios), que pueden ser rápidamente aprovisionados y liberados con un esfuerzo de gestión reducido o interacción mínima con el proveedor del servicio*.

Según dicha definición, cloud computing se entiende como un modelo de prestación de servicios informáticos cuya principal orientación es la escalabilidad. Desde el punto de vista de los usuarios, los servicios son elásticos, es decir, que pueden crecer o recuperar su tamaño original de manera rápida y sencilla. Esto posibilita que las organizaciones se despreocupen de la gestión física de los recursos y distribuyan adecuadamente su capital sin tener que entrar en costosas inversiones para atender requerimientos temporales, concentrando todos sus esfuerzos en los objetivos particulares de su negocio. Esta orientación permite que los usuarios que acceden a los servicios, perciban que todo funciona de manera simple y rápida, dando como resultado una experiencia más gratificante.

2.1.1. EL CLOUD COMPUTING SEGÚN EL NIST

Cloud Computing es un paradigma en evolución y la definición NIST caracteriza sus aspectos más importantes y su objetivo es realizar una comparación de los servicios y estrategias de despliegue. De esta manera ofrece una base a partir de la cual se podrá discutir sobre las mejoras prácticas en cloud computing.

Según el NIST el modelo de cloud computing está compuesto por cinco características claves, tres modelos de servicio y cuatro modelos de despliegue, como se muestra en la figura 1. A continuación se describen las características claves del cloud computing:

- *Rapidez y Elasticidad:* Elasticidad se define como la habilidad para escalar recursos de forma ascendente y descendente según se necesite. En algunos casos esta capacidad se realiza de manera automática para proveer el dimensionado correspondiente de forma rápida. Para los clientes, las capacidades disponibles aparentan ser ilimitadas y pueden adquirirse en cualquier cantidad y en cualquier momento.
- *Servicio Supervisado:* En un servicio supervisado, los aspectos del servicio cloud pueden seguirse, controlarse y notificarse, lo que aporta transparencia tanto para el proveedor como para el consumidor del servicio utilizado. Esto es crucial para la facturación, control de acceso, optimización de recursos, planificación de capacidad y otras tareas.

- *Auto-Servicio bajo Demanda:* Los aspectos de autoservicio y On-demand de cloud computing significan que un cliente puede usar servicios en el cloud, según necesite sin ninguna interacción humana con el proveedor de los servicios.
- *Ubicuidad de acceso de red:* Acceso ubicuo a la red significa que las capacidades del proveedor de servicios están disponibles en la red y pueden ser accedidas a través de mecanismos estándar que fomentan el uso por parte de plataformas de clientes heterogéneas tales como teléfonos móviles, notebooks, PDA, entre otras.
- *Fondo común de Recursos:* El fondo común de recursos permite a un proveedor de cloud computing servir a sus clientes a través de un modelo de multiposesión en el que los recursos computacionales se ponen en reservas en común para que puedan ser utilizados por múltiples clientes. Los recursos físicos y virtuales son asignados y reasignados dinámicamente de acuerdo a la demanda del consumidor. Hay una sentido de independencia de la ubicación física en la que los cliente generalmente no tiene control o conocimiento sobre la ubicación exacta de los recursos asignados, pero se puede especificar una ubicación a un nivel más alto de abstracción (por ejemplo, país, estado, o de centros de datos). Algunos ejemplos de recursos son: almacenamiento, procesamiento, memoria, ancho de banda y máquinas virtuales.

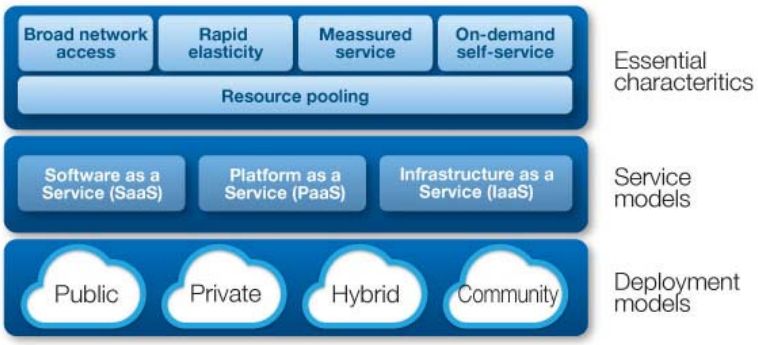


Figura 1: Definición del NIST de cloud computing

Los *Modelos de Servicios* se refieren a la forma en la cual están organizados los servicios de cloud computing. Esta organización obedece a una arquitectura de tres capas: software-as-a-service o SaaS, platform-as-a-service o PaaS e infrastructure-as-a-service o IaaS, tal como lo muestra la figura 2.

- El *IaaS,* es la capa más baja de cloud computing. El propósito es externalizar el hardware e infraestructura de red necesaria para correr una aplicación, como así también su mantenimiento y escalabilidad, mediante una plataforma virtualizada. Tal es el caso de servidores, discos, routers, swtiches, refrigeración de los equipos y demás.

  Con un IaaS lo que se tiene es una solución en la que se paga por consumo de recursos utilizados: espacio en disco, tiempo de CPU, cantidad de consultas a una base de datos, transferencia de datos en un determinado tiempo, etc. El IaaS permite desplazar una serie de problemas al proveedor relacionado con la gestión de los equipos, como el ahorro de costos al pagar sólo por lo consumido y olvidarse de tratar con hardware y su mantenimiento.

  El *PaaS,* brinda a los clientes la capacidad de desplegar sus aplicaciones en la infraestructura del cloud, utilizando diferentes lenguajes y herramientas de programación que el proveedor del servicio soporte.

  A diferencia de los servicios ofrecidos por el IaaS, los clientes no gestionan ni controlan la infraestructura del cloud, pero tienen el control sobre las aplicaciones desplegadas y su configuración. El proveedor es el encargado de escalar los recursos en caso de que la

aplicación lo requiera, del rendimiento óptimo de la plataforma, seguridad de acceso, etc. Un ejemplo, es Google App Engine que permite desarrollar, compartir y alojar aplicaciones Web en la infraestructura de Google.

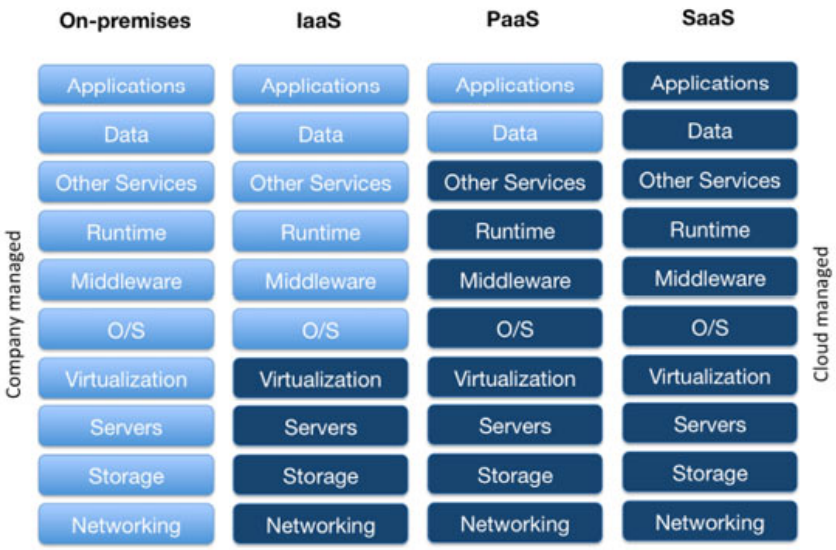


Figura 2: Modelo de Servicio provisto en cloud computing

- El *SaaS*, es la capa más alta y consiste en la entrega de una aplicación completa como un servicio. El proveedor SaaS dispone de una aplicación estándar desarrollada en algunos casos por él mismo que se encarga de operar y mantener y con la cual da servicio a multitud de clientes a través de la red, sin que estos tengan que instalar ningún software adicional.

  La distribución de la aplicación tiene el modelo de uno a muchos, es decir, se realiza un producto y el mismo lo utilizan varios clientes. Los proveedores de SaaS son responsables de la disponibilidad y funcionalidad de sus servicios no dejando de lado las necesidades de los clientes que son, al fin y al cabo, los que usaran el software.

  Un ejemplo de un SaaS es la aplicación para el manejo de correo electrónico (como Gmail, Hotmail y Yahoo) por medio de un web-browser.

Los *Modelos de Despliegue* hacen referencia a como se monta y pone en funcionamiento el sistema. Se definen cuatro Modelos de Despliegue, también conocidos como Tipos de Cloud, que ofrecen la infraestructura necesaria para soportar los Modelos de Servicio. Estos cuatro Modelos de Despliegue, se pueden ver en la figura 3 y son:

- *Cloud Público:* En este caso los recursos son proporcionados dinámicamente a través de Internet, siendo suministrados por un proveedor externo a la organización, que entrega los recursos y luego cobra por su uso. La infraestructura es propiedad de la organización que ofrece los servicios en el cloud. El término público no siempre significa gratis ni tampoco que los datos de los clientes son visibles para cualquiera. Generalmente los proveedores de Cloud Públicos ofrecen mecanismos de control de acceso para sus usuarios. Este tipo de modelo de despliegue, permite implementar soluciones de manera flexible y efectiva a nivel de costos.

  En este modelo, la infraestructura está orientada a servir a una sola organización la cual controla qué aplicaciones se deben correr y dónde. Puede existir en las dependencias propias de la organización o fuera de ellas. La diferencia principal entre Cloud Público y Privado es que en el Cloud Privado los datos y procesos son administrados dentro de

la organización sin restricciones de ancho de banda, vulnerabilidades de seguridad y complicaciones legales que podrían acarrearse al usar Cloud Público. Además el Cloud Privado ofrece al proveedor y usuario mayor control de la infraestructura Cloud, mejorando la seguridad debido a que la utilización de la red y el acceso de los usuarios están restringidos y designados.

- *Cloud Híbrido:* En el modelo híbrido combinan los modelos de Cloud Público y Privado, es decir, el cliente es propietario de una parte del cloud y la otra la comparte. En este modelo los clientes normalmente externalizan el cloud Pública la información de negocio no crítica y el procesamiento de información, mientras que mantienen bajo su control los servicios y datos críticos para el funcionamiento de la organización.
- *Cloud Comunitario:* En un modelo de cloud Comunitario la infraestructura es compartida y utilizada por un grupo de organizaciones que tienen intereses compartidos, tales como requerimientos específicos de seguridad, políticas o una misión común. Los miembros de la comunidad comparten el acceso a los datos y aplicaciones en el cloud

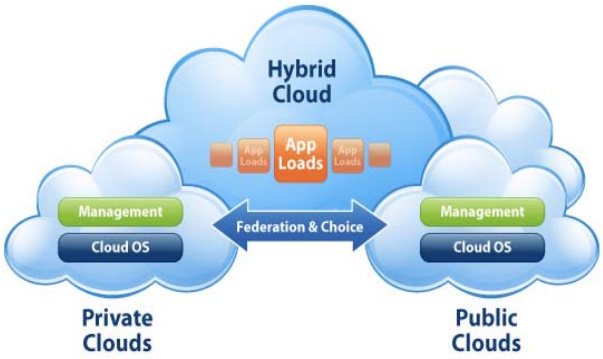


Figura 3: Modelo de Despliegue provisto en cloud computing

Mediante la combinación de los Modelos de Servicios y los Modelos de Despliegues, es posible contar con una amplia gama de soluciones que pueden adaptarse y configurarse en función de las necesidades de la organización.

2.1.2. VENTAJAS DEL CLOUD COMPUTING

El Cloud Computing es una tendencia tecnológica que está significando una gran revolución en todo el mundo. Pero como cualquier fenómeno, presenta sus oportunidades y amenazas que son importantes conocer. Entre las ventajas más importantes se pueden mencionar:

- *Reducción de costos:* No hay necesidad de adquirir hardware y software lo que reduce costos operativos en infraestructura, mantenimiento y energía. El cloud es más barato que la instalación y mantenimiento de un servidor propio o contratar los servicios de un proveedor.
- *Flexibilidad:* El servicio de cloud se paga de acuerdo a la demanda. Si, por ejemplo, una organización los días treinta incrementa el movimiento de su área contable y financiera por pagos a empleados y proveedores, puede decidir que requiere mayor capacidad de proceso o de almacenamiento de datos, y pagará por una mayor demanda, pero sólo el día 30.
- *Movilidad:* Los datos de una organización al quedar alojados en el cloud pueden ser consultados por los empleados desde cualquier lugar. Esta característica está significando un crecimiento del teletrabajo con todos sus efectos de tipo económico, social e incluso, inmobiliario.

- *Focalización:* Cloud Computing permite a las compañías centrarse en su core business (negocio principal). En lugar de hacer una alta inversión tecnológica en sistemas, una organización podría invertir en su infraestructura industrial o física o en capital humano para proseguir sus planes de expansión
- *Ecología:* Usar el cloud en una organización reduce la huella de carbono al ahorrar recursos y componentes que pasan de estar almacenados en componentes físicos a ser virtuales. Se ahorra también en consumo de energía con sus beneficios al medio ambiente.

2.1.3. DESVENTAJAS DEL CLOUD COMPUTING

Entre las desventajas de usar cloud computing, se pueden mencionar las siguientes:

- *Seguridad:* Se debe ser muy cuidadoso con el manejo de la información para evitar que los datos sean robados por hackers o extraviados en agujeros de seguridad.
- *Privacidad:* Datos confidenciales y sensibles como planes de mercadeo, lanzamientos de productos, información personal de empleados, datos financieros pueden quedar en manos de terceros si no se tienen las medidas preventivas.
- *Conectividad:* La velocidad de acceso a la información y la disponibilidad de las aplicaciones dependen de la velocidad de la conexión a internet. Sin acceso a Internet no hay Cloud Computing y este servicio puede caerse en cualquier momento por diversos factores.

2.2. ARQUITECTURA DE CLOUD COMPUTING

La arquitectura de cloud computing se la puede representar como un conjunto de capas que se encuentran acopladas para lograr darle funcionalidad al sistema. La arquitectura de cloud computing es similar a la arquitectura de red, desde un nivel físico a un nivel aplicación debido a que se usan protocolos similares a los que se usan en internet [5]. En la figura 5, se muestra la arquitectura genérica de cloud computing:

1. *Aplicación:* incluye servicios basados en web y SaaS. A diferencia de las aplicaciones tradicionales, las aplicaciones de esta capa escalan automáticamente, poseen mejor performance y mayor disponibilidad. Ejemplos de aplicaciones son Google App, CRM de Salesforce, entre otras.
2. *Plataforma:* esta capa incluye los frameworks de aplicación y su propósito es minimizar el impacto del desarrollo de aplicaciones directamente sobre la infraestructura montada. Ejemplos de plataformas son Google App Engine, Microsoft Azure, OpenShift de Red Hat, entre otras.
3. *Infraestructura:* incluye software que permite la manipulación de la infraestructura física subyacente. Entre las opciones más importantes que se encuentran en esta capa esta la provisión de un sistema operativo de cloud como OpenStack o Eye OS o las plataformas de infraestructura para construir cloud como OpenNebula o CloudStack.
4. *Virtualización:* incluye infraestructura virtual como un servicio, mediante la creación de un pool de recursos de almacenamiento y computo, lo cual se logra particionando los recursos físicos mediante algún hipervisor (tal como, Xen, KVM o Vmware).
5. *Recursos físicos:* esta capa es la responsable de administrar los recursos físicos del cloud, incluyendo servidores, almacenamiento y red. En la práctica esta capa se implementa en un data center.

Figura 5: Arquitectura de cloud computing

La arquitectura para Cloud Computing es una evolución natural del concepto de Clusters y Grids, y permite realizar la integración de grandes cantidades de recursos virtualizados (hardware, plataformas de desarrollo y/o servicios), fácilmente accesibles y utilizables por usuarios distribuidos geográficamente. Estos recursos pueden ser dinámicamente reconfigurados para adaptarse a una carga variable, permitiendo optimizar su uso.

Una *Arquitectura de Cloud Computing*, está formada por dos componentes: hardware y aplicaciones, los cuales trabajan de manera conjunta e intrínsecamente relacionados. Esto es, si las aplicaciones fallan el hardware no será capaz por sí solo de ejecutar los procesos necesarios para obtener los resultados deseados; por otro lado, si el hardware falla toda aplicación será incapaz de ejecutarse.

Lo más importante de contar con una Arquitectura con características de modularidad es que las aplicaciones se construyen sobre la arquitectura cloud, de forma tal que use la infraestructura subyacente solo cuando la aplicación lo necesite. En este ambiente, las aplicaciones piden los recursos que necesitan para realizar sus tareas, realiza el trabajo especifico y luego los libera; mientras la aplicación esta en operación, escala elásticamente con respecto a los recursos necesarios [6].

2.3. La Infraestructura como base del Cloud Computing

La infraestructura como servicio es un modelo de distribución de infraestructura de cómputo como un servicio, normalmente mediante el uso de una plataforma de virtualización. Los clientes de IT en vez de adquirir equipamiento físico como servidores, espacio en un data center o equipamiento de redes, compran todos estos recursos a un proveedor de servicios externo. Esto es completamente diferente respecto al uso de un hosting virtual, ya que el aprovisionamiento de estos servicios se hace de manera completamente integral a través de la web.

Esta capa de la arquitectura cloud proporciona un entorno totalmente escalable a las necesidades fluctuantes del cliente. La escalabilidad dinámica que proporciona IaaS hace que para las pequeñas organizaciones, el trasladarse al cloud sea posible, ya que se contrata según las necesidades. Muchas organizaciones tienen sus servidores y data center inactivos durante muchas horas del día, limitándose su actividad a la jornada laboral, aun así, los servidores tienen que ser mantenidos. Los proveedores de servicios cloud utilizan multi-tenancy para compensar estos costos en su extremo, y son capaces de ofrecer IaaS a sus clientes en un paquete asequible y escalable. La externalización de las responsabilidades de mantenimiento de la infraestructura libera al departamento de TI y le permite enfocarse en otras áreas.

Mediante la implementación de un IaaS, los usuarios podrán acceder a la infraestructura a nivel mundial, utilizando cualquier dispositivo que pueda conectarse a Internet. Los proveedores

de servicios cloud deberán de poder flexibilizar esta accesibilidad ofreciendo adecuados mecanismos de confidencialidad e integridad de la información y de los accesos.

En algunos casos la utilización de IaaS se plantea como una paulatina eliminación de los servidores físicos propios y su sustitución por servidores virtuales ubicados en centros de procesamiento de datos remotos. Esta solución redunda de forma inmediata en importantes ahorros de costos.

Para lograr este nivel de administración y abstracción de los recursos físicos la base sobre la cual se apoya infraestructura cloud es la virtualización. Es más, la características más interesante del cloud es la elasticidad, la cual permite ante un pico elevado de requisitos de cómputo, un usuario puede solicitar mayor capacidad de cálculo, que provocará el despliegue automático de nuevas máquinas virtual sobre la infraestructura física existente.

Desde el punto de vista de la persona encargada de la instalación, configuración y gestión del cloud, la capa más importante es la infraestructura y la posibilidad de virtualizar recursos con el objeto de aumentar la performance del cloud. La Virtualización se refiere estrictamente a la optimización de los recursos de la máquina real usando máquinas virtuales; las implementaciones del software de las máquinas virtuales ejecutan programas como si no se hubiera separado las máquinas físicas.

Todos los productos disponibles para instalar una infraestructura cloud cuentan con una capa de virtualización que permite instanciar de forma virtual cada uno de los recursos físicos. Esta capa es el hipervisor, el cual es una herramienta de software capaz de comunicarse con el sistema anfitrión, ya sea hardware o un sistema operativo, para interoperar entre un sistema de virtualización y el sistema físico. Su objetivo es gestionar los cuatro recursos principales de un computador (CPU, memoria, red y discos de almacenamiento) repartiendo dinámicamente dichos recursos entre todas las máquinas virtuales.

Existen dos tipos fundamentales de hipervisores:

- *Hipervisores de tipo 1 o bare-metal:* capaces de ejecutarse directamente sobre el hardware real de la máquina sin necesidad de tener instalado ningún SO. Este tipo de hipervisores consiguen dar un mayor rendimiento y escalabilidad y menos sobrecarga. La desventaja es que no todo el hardware soporta este tipo de software, pero son los que se suelen emplear normalmente en grandes infraestructuras virtualizadas. Ejemplos de este tipo de hipervisores son Xen, KVM o VMware ESXi.
- *Hipervisores de tipo 2 o Hosted:* necesitan un sistema operativo anfitrión para poder ejecutarse. Al no ejecutarse directamente sobre el hardware, el rendimiento de este tipo de hipervisores es menor. Suelen utilizarse en entornos de escritorio y en infraestructuras virtualizadas de prueba. Algunos de ellos son Qemu, VirtualBox, VMware Player o Virtual PC.

Una Infraestructura cloud, desacopla las máquinas virtuales de la localización física disponible, creando una capa de virtualización distribuida que se sitúa entre la infraestructura física existente y el servicio que se presta a los usuarios. Esta capa permite extender los beneficios de las plataformas de virtualización a múltiples recursos y permite gestionar de forma independiente los recursos físicos y los servicios. Esto aporta ventajas tales como la gestión centralizada, el escalado y particionado de recursos dinámico, monitorización y la provisión bajo demanda de máquinas virtuales [8] [9].

3. ANÁLISIS DE LAS INFRAESTRUCTURAS DEL MERCADO

En los últimos años el número de soluciones open source dedicadas a la administración y gestión de entornos de Cloud Computing ha crecido exponencialmente, grandes compañías como Citrix o Red Hat están apostando fuertemente por este tipo de soluciones.

Un licenciamiento gratuito permite reducir los costos de la infraestructura. Pero este hecho debe estar balanceado con los costos de soporte y de desarrollo para personalizar el código de la solución. El licenciamiento en open source normalmente es menos complicado que en soluciones propietarias.

Tradicionalmente el software comercial fue diseñado para entornos estáticos. Por lo tanto, puede ser un pequeño dolor de cabeza como licenciar un entorno dinámico como es cloud. Adicionalmente, las soluciones open source permiten tener un control mayor sobre las fases de pruebas y evaluación de tecnologías cloud, por tanto permiten crear el entorno ideal para este proyecto.

Las principales soluciones open source actuales son:

- *Eucalyptus*
- *Open Nebula*
- *Open Stack*
- *CloudStack*

Todas ellas soluciones de probada reputación, respaldadas por grandes empresas y/o organizaciones y con un amplio soporte en la comunidad.

EUCALYPTUS (Elastic Utility Computing Architecture Linking Your Programs To Useful Systems)

www.eucalyptu.com

Eucalyptus [10] [11] es una arquitectura software open source basada en Linux que implementa clouds privados e híbridos dentro de una infraestructura de TI de una organización. Inicialmente fue diseñado y desarrollado por el equipo del profesor Rich Wolski, como un proyecto del Computer Science Department de la Universidad de California.

Se diseñó para ser fácil de instalar de forma lo menos intrusiva posible. Proporciona una capa de red virtual de tal forma que se aísla el tráfico de red de diferentes usuarios y permite que uno o más clusters parezcan pertenecer a la misma LAN (sólo Enterprise Edition). Además, tiene la capacidad de interactuar con Amazon EC2 y los servicios S3 de cloud público ofreciendo la posibilidad de crear un cloud híbrido, ofreciendo interfaces REST y SOAP.

Las principales funcionalidades incluyen contar con una API compatible con AWS, soporte de múltiples hipervisores y el gestor Walrus de almacenamiento compatible con S3.

OPENNEBULA

www.opennebula.org

OpenNebula [12] [13] es un software open source que permite construir cualquier tipo de cloud privado, público e híbrido. Se trata de un producto software bajo licenciamiento Apache 2.0, desarrollado en la Universidad Complutense de Madrid y financiado por varios proyectos europeos (Reservoir, 4CaasT, ...) y nacionales (Nuba, HPCcloud, Medianet, …).

Ha sido diseñado para ser integrado con cualquier tipo de red y almacenamiento, para así adaptarse a cualquier centro de datos existente. La gestión del almacenamiento, imágenes y todos los archivos se realiza mediante NFS o GlusterFS.

Posee soporte para múltiples hipervisores, provee dos APIs, Open Cloud Computing Interface (OCCI) API y EC2 API. La seguridad la maneja mediante autenticación y autorización de grupos y usuarios basada en claves, SSH, RSA y certificados X509 o LDAP.

CLOUDSTACK

cloudstack.apache.org

CloudStack [14] [15] es una solución software open source desarrollada por Cloud.com, que permite efectuar el despliegue, configuración y gestión de entornos de computación elástica, tanto para hipervisores Xen Server como KVM. Además de la versión CloudStack Community Edition, soportada por la comunidad, se ofrecen dos versiones comerciales, CloudStack Enterprise Edition, para empresas y CloudStack Service Provider Edition, para proveedores de servicios.

CloudStack puede desplegarse en uno o más servidores de gestión de tal forma que se conectarían a una única base de datos MySQL. Opcionalmente, se podrían distribuir las peticiones Web mediante el empleo de gestores de balance de carga. Además, una copia de seguridad de la base de datos del servidor de gestión podría desplegarse empleando la replicación MySQL en un sitio remoto.

La infraestructura se basa en la utilización de nodos de computación, PODS (conjunto de nodos de computación) y zonas de disponibilidad (conjunto de PODS y almacenamiento secundario). Se pueden añadir nodos de computación adicionales en cualquier momento para proporcionar mayor capacidad a las máquinas virtuales huésped.

Se ofrecen dos versiones comerciales, CloudStack Enterprise Edition, para empresas y CloudStack Service Provider Edition, para proveedores de servicios.

OPENSTACK

www.openstack.org

OpenStack [16] [17] es un proyecto que se inició en el año 2010 por la empresa Rackspace Cloud y por la agencia espacial norteamericana, NASA. Rackspace aporta la parte de almacenamiento, llamada Swift, la NASA, colabora con Nova, base de su plataforma NEBULA, ambos servicios están desarrollados utilizando Python.

OpenStack puede ser utilizado por cualquier organización que busque desplegar un cloud de gran escala tanto para uso privado como público. Ofrece la capacidad de control gran cantidad de recursos de cómputo, almacenamiento y red permitiendo el aprovisionamiento elástico de estos recursos. Está escrito en phyton y su arquitectura es totalmente modular. Provee soporte de múltiples hipervisores y su uso por defecto depende de la arquitectura de instalación. Como posee una arquitectura modular, posee un modulo diferente para administrar la seguridad, el almacenamiento la virtualización de recursos y la red.

3.1. COMPARACIÓN DE ASPECTOS TÉCNICOS

Para poder seleccionar una de ellas, se deben analizar una serie de características [18] [19] [20] [21]:

- *Adaptabilidad:* a los distintos entornos hardware.
- *Escalabilidad:* posibilidad de variar el número de nodos de manera sencilla.
- *Topologías:* que permitan crear Clouds y que los nodos se repartan las funciones de formas distintas.
- *Interfaz Web:* que aporte la funcionalidad necesaria para controlar el cloud de una forma simple.
- *Instalación:* ha de ser lo más sencilla posible.
- *Manejo de Instancias.*
- *Soporte.*

En primer lugar, *CloudStack* es un producto que cumple con gran parte de los requisitos, dado que posee una amplia comunidad de desarrollo que implica a algunas grandes empresas, como Apache. Esta comunidad ha desarrollado una infraestructura cloud adaptable a cualquier tipo de servidor, con una interfaz web sencilla y que permite gestionar el cloud y manejar las instancias de recursos virtualizados con facilidad. Por el contrario, esta plataforma está en cierto modo limitada en cuanto a las distintas topologías que se pueden configurar, dado que solo permite una arquitectura basada en PODS gestionados desde nodos controladores. En cuanto a la escalabilidad, permite añadir y eliminar PODS sin que el sistema se vea afectado.

Con respecto a *Eucalyptus*, es un producto con soporte pues está basada en Ubuntu Enterprise Cloud, es sencilla de instalar y se adapta a cualquier servidor. Además permite configurar distintos tipos de topologías en las que los nodos de cómputo pueden agruparse en grupos controlados por nodos de gestión. Esta infraestructura permite configurar distintas interfaces web de control con lo que ofrece compatibilidad con controladores públicos como EC2 o S3 de Amazon. Sin embargo, estas interfaces web, no aportan una gestión tan sencilla y eficaz como las de otras plataformas.

OpenNebula se destaca por ser adaptable a cualquier entorno y por su escalabilidad, dado que una de sus características más importantes es la capacidad de integración con otras cloud públicas como EC2 de Amazon. Aporta una interfaz web propia sencilla, pero a su vez no resulta muy eficaz en el manejo de instancias virtuales. Una desventaja es la complejidad de su instalación y que solo permite un tipo de arquitectura clásica en la que un nodo frontend controla el sistema y los nodos de cómputo realizan el trabajo.

Por último, *OpenStack* es el producto más nuevo y cuenta con apoyos de numerosas y grandes empresas, lo que hace que esté garantizado su buen funcionamiento. La interfaz web que controla el sistema aporta al usuario la capacidad de gestionar el de forma sencilla. Resulta muy destacable el manejo de instancias ya que pueden crearse, modificarse y eliminarse instancias virtuales, creando patrones que sirven para crear nuevas máquinas virtuales. Este producto permite crear clouds con distintas topologías que se adaptan a cualquier tipo de sistema, pudiendo añadir nuevos servidores al sistema o eliminando los que ya forman parte del mismo sin que se pierda la integridad. Para terminar, hay que destacar uno de los aspectos fundamentales y es que la plataforma OpenStack cuenta con una distribución llamada DevStack que aporta una solución para configurar la instalación semiautomática de OpenStack. Además cuenta con TryStack, una plataforma que permite utilizar OpenStack gratuitamente para hacer pruebas y para probar configuraciones. Es una manera de comenzar con OpenStack que solo permite lanzar instancias con unos recursos limitados.

Otro aspecto importante que se debe avaluar es el roadmap y la agilidad de desarrollo, y en este punto solo OpenStack ofrece un *Road Map totalmente abierto* y *agilidad de desarrollo.* Esto se debe a que se sabe con anticipación que va a traer el producto en la próxima versión e incluso se puede probar aun en Beta, esto brinda la posibilidad de planificar la evolución de de la infraestructura cloud. La agilidad de desarrollo es otro punto fundamental, a diferencia de de los otros tres productos que suele tener una versión por año, OpenStack tiene un ciclo de versionado de seis meses, ofreciendo una evolución más constante que aporta diferenciales en el producto final.

A modo de resumen, se puede ver en la tabla 1 un detalle de los aspectos técnicos, y más relevantes de cada una de las cuatro infraestructuras analizadas.

Versión	Eucalyptus	OpenNebula	CloudStack	OpenStack
Inicio	2008	2008	2010	2010
Licencia	Propietaria (GPL v3)	Apache 2.0	Apache 2.0	Apache 2.0
Implementación	Privado, Hibrido	Privado, Publico, Hibrido	Privado, Publico	Privado, Publico, Hibrido
Interface de acceso para el usuario	Web Service, Línea de comando	API EC2 y OCCI	Interface web	Interface web (Horizon)
Virtualización	Xen, KVM	VMWare, Xen, KVM	Software Embebido	QEMU, Hyper-V, VMWare, Xen, KVM, UML , LXC
Lenguaje de Programación	C, Java	C, C++, Java, Rubi, Shell	Phyton, Java	Phyton
Soporte de SO	Linux	Linux	Linux	Linux (Ubuntu)
Arquitectura	Modular / Monolítica	Estilo Cluster	Jerárquica	Fragmentada Monolítica / Distribuida

Tabla 1: Comparación de aspectos técnicas de las cuatro infraestructuras

3.2. COMPARACIÓN DE COMUNIDADES

Los cuatro productos analizados: *Eucalyptus, CloudStack, OpenStack* y *OpenNebula* son infraestructuras open source, por lo que su desarrollo no puede separarse de las contribuciones de la comunidad. Una comunidad puede consistir de individuos, equipos y organizaciones, por lo cualquier usuario, un desarrollador, un proveedor de servicios, incluso una corporación, puede unirse a la comunidad y contribuir a construir una mejor cloud open source.

En la Universidad de Sydney, desde el año 2011 se genera el *"Open Source Iaas Community Analysis — OpenStack vs OpenNebula vs Eucalyptus vs CloudStack"*, el cual es el resultado del Proyecto de investigación *"Análisis de Comunidades"* y cuyo objetivo es ofrecer un informe trimestral con el movimiento de las comunidades de OpenStack, OpenNebula, Eucalytpus y CloudStack: Esto se realiza analizando las comunicaciones entre los miembros de la comunidad en forma de listas de correo o foros de debate púbico. Los datos que se discuten incluyen el número total de temas, mensajes y participantes.

Para este trabajo se ha tomado el informe *"CY15-Q1 Community Analysis — OpenStack vs OpenNebula vs Eucalyptus vs CloudStack"*, el cual pertenece al análisis de las comunidades del primer trimestre de 2015.

Las figuras 6 y 7 representan el número mensual de temas y mensajes. Se puede observar que durante los últimos 12 meses, las discusiones relacionadas con OpenStack aumentaron en forma casi lineal, mientras que las relacionadas con CloudStack disminuyeron rápidamente. Mientras que el volumen de las discusiones en torno a OpenNebula y Eucalyptus mostraron tendencias a disminuir.

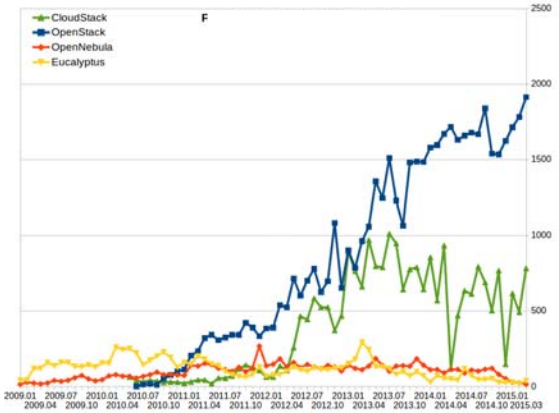


Figura 6: Número mensual de temas

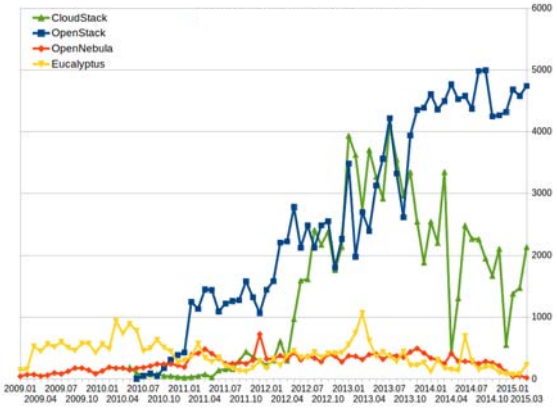


Figura 7: Número mensual de mensajes

En términos generales, el número de respuestas a un tema específico representa que está recibiendo atención, y la profundidad de la discusión que a ese tema se le da en particular. Cuando el número de post (el post original que comenzó un tema en particular) es mayor que el número de respuestas, se puede deducir que la participación en la lista o foro es muy baja. Por lo tanto, la relación entre *"el número de post"* y *"el número de temas"* representa la tasa de participación de una comunidad en línea, lo que se llama tasa de participación.

Como se puede ver en la figura 8, durante los últimos 12 meses, las tasas de participación de CloudStack y Eucalyptus son relativamente más altas, mientras que las tasas de participación de OpenStack y OpenNebula son relativamente más bajos.

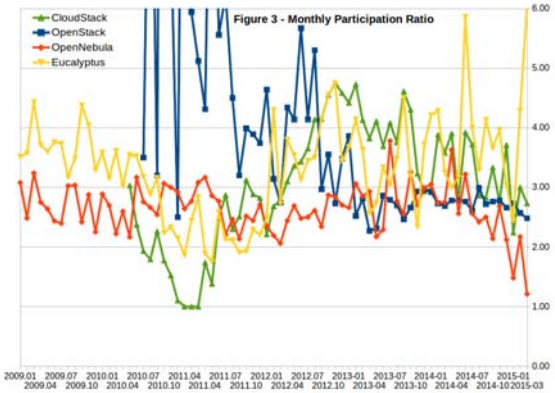


Figura 8: Tasa de participación mensual

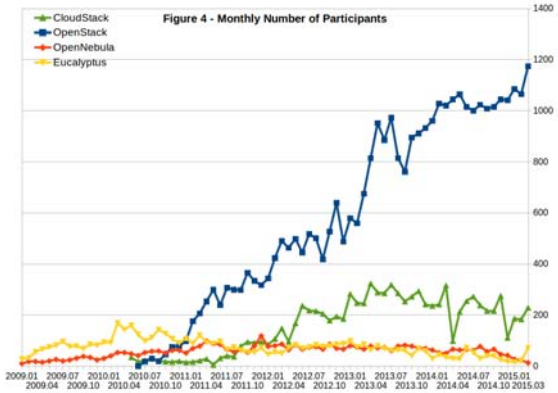


Figura 9: Número de participantes mensuales

La figura 9 muestra los participantes activos de los cuatro proyectos, se puede observar que el número de participantes activos de OpenStack es mucho mayor que los otros tres proyectos. El número de participantes activos de CloudStack también es significativamente más alto que OpenNebula y Eucalyptus. Para el caso de OpenStack, el número de participantes activos para crece de manera constante, mientras que el número de participantes activos para CloudStack, Eucalyptus y OpenNebula exhibió disminución significativa.

Otro aspecto importante es el acceso a los repositorios de paquetes para mantener cada versión de los productos, por lo que para entender las actividades de desarrollo en estos proyectos se pueden analizar los registros de actividades de desarrolladores y organizaciones contribuyentes, así como su frecuencia. Aprovechando el hecho de que los cuatro proyectos utilizan git como software de gestión de versiones para sus paquetes, se puede utilizar "git log" para obtener información de registro de actividades desde los repositorios git. Cabe señalar que para el proyecto OpenStack, la fuente de datos incluye todos los sub-proyectos en OpenStack (137 subproyectos) y OpenStack infra (114 subproyectos) en github.com.

La figura 10 muestra el número mensual de operaciones para estos cuatro proyectos. En términos generales, la frecuencia de acceso a los repositorios de OpenStack es mucho mayor con respecto a los otros tres proyectos, esto es debido a que los paquetes para OpenStack incluye un total de 251 sub-proyectos, lo que es mucho más grande que los otros tres proyectos. La frecuencia de acceso de CloudStack es ligeramente superior a la de Eucalyptus y OpenNebula.

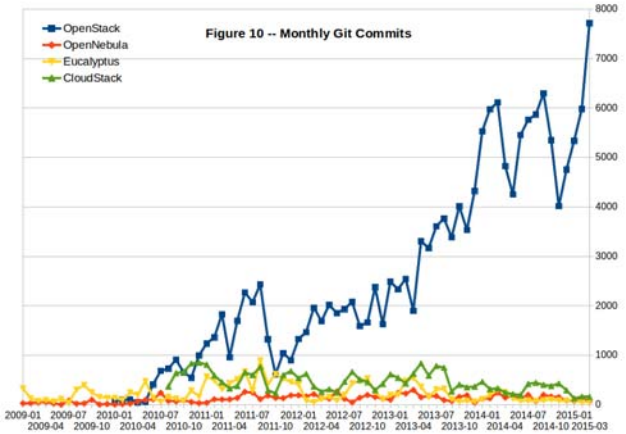


Figura 10: Accesos mensuales a Git

3.3. INFRAESTRUCTURA SELECCIONADA

En función de todos los aspectos analizados, se puede llegar a la conclusión que *OpenStack* es el producto más adecuado para implementar una infraestructura cloud. En la actualidad, es el producto para instalar infraestructura cloud con mayor aceptación entre la comunidad de usuarios y desarrolladores.

Además, es importante destacar el apoyo que le están dando grandes compañías de desarrollo de software como IBM, Red Hat y especialmente Canonical que desde el año 2013 incorpora a las versiones servidoras de Ubuntu una preinstalación de OpenStack.

Desde el punto de vista del diseño, OpenStack, permite la administración y gestión de una infraestructura flexible y escalable con capacidades de convertirse en una solución hibrida. La principal ventaja es la administración distribuida de las instancias de recursos físicos y su migración en caliente para lograr balanceo de carga de las máquinas virtuales.

4. OPENSTACK

4.1. GENERALIDADES

OpenStack [22] cuyo logo se muestra en la figura 11, es una solución de cloud computing del tipo IaaS, más aun, es una colección de tecnologías Open Source que proporcionan un software para el despliegue escalable de un cloud computing. Mas precisamente, se puede definir a OpenStack como un sistema operativo de cloud capaz de administrar y controlar un gran conjunto de recursos de cómputo, almacenamiento y red.

OpenStack, es un proyecto que se inició en el año 2010 por la empresa Rackspace Cloud y por la agencia espacial norteamericana, NASA. Actualmente más de 150 empresas se han unido al proyecto, entre las que se encuentran AMD, Intel, Canonical, SUSE Linux, Red Hat, IBM, Dell, HP, Cisco, etc. OpenStack puede ser utilizado por cualquier organización que busque desplegar un cloud de gran escala tanto para uso privado como público.

Figura 11: logo de OpenStack

El Proyecto es administrado por la OpenStack Foundation (www.openstack.org/foundation/), una entidad corporativa sin fines de lucro establecida en Septiembre del 2012 para promover el software de OpenStack y su comunidad, que está integrada por miembros de su comunidad de programadores, usuarios y patrocinadores y cuyos principios se resumen en los siguientes puntos:

- *Licencia Apache 2.0, la cual es una licencia de software libre permisivo y no copyleft.*
- *No existe versión "enterprise".*
- *Proceso de diseño abierto.*
- *Repositorios públicos de código fuente.*
- *Todos los procesos de desarrollo deben estar documentados y ser transparentes.*
- *Orientado para adoptar estándares abiertos.*
- *Diseño modular que permite flexibilidad mediante el uso de APIs.*

OpenStack podría considerarse un CMP (Cloud Management Plataform) que permite orquestar y gestionar los diferentes aspectos necesarios para montar una infraestructura de cloud basada en estándares open source. Su misión es proveer una solución flexible tanto para cloud públicos como privados, de cualquier tamaño, y para esto se consideran dos requerimientos básicos: los cloud deben ser simples de implementar y masivamente escalables.

Lo interesante de OpenStack es su capacidad de extensibilidad a través de APIs que son fáciles de consumir (al estilo de AWS), públicos y son "vendor free", por lo que muchos proveedores de servicios, están evaluando migrar su infraestructura a OpenStack.

Desde el punto de vista técnico, OpenStack posee cuatro servicios principales:

- *OpenStack Service Clients ("clients")*
- *OpenStack APIs ("APIs")*
- *OpenStack Cloud Services ("services")*
- *Infrastructure Pool Providers ("providers")*

4.2. ARQUITECTURA DE OPENSTACK

OpenStack está divido en diferentes componentes [23] que trabajan en conjunto. Esta integración es lograda a través de APIs, que cada servicio ofrece y consume. Gracias a estas APIs, los servicios pueden comunicarse entre ellos y además se posibilita que un servicio sea reemplazado por otro de similares características siempre que se respete la forma de comunicación. Es decir, OpenStack es extensible y se ajusta a las necesidades de quien desee implementarlo.

Actualmente, son cinco los componentes principales de OpenStack, como se muestra en la figura 12:

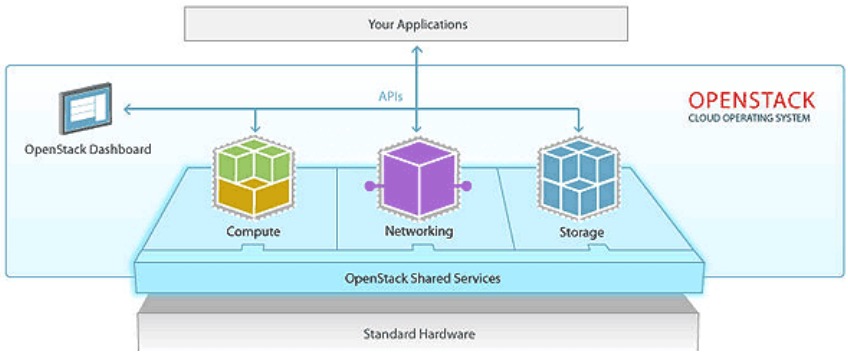


Figura 12: Componentes de OpenStack

- *OpenStack Compute:* es el controlador de la estructura básica del Cloud. Es el encargado de iniciar las instancias (máquinas virtuales) de los usuarios y grupos. También es el servicio encargado de la gestión de la red virtual para cada instancia o para las múltiples instancias que formen parte de un proyecto (tenant).
- *OpenStack Object Storage:* es el servicio encargado del almacenamiento masivo de objetos a través de un sistema escalable, redundante y tolerante a fallos. Las posibles aplicaciones de Object Storage son numerosas, como por ejemplo: almacenamiento simple de ficheros, copias de seguridad, almacenamiento de streamings de audio/vídeo, almacenamiento secundario/terciario, desarrollo de nuevas aplicaciones con almacenamiento integrado, etc.
- *OpenStack Identity Service:* es un servicio usado para la autenticación entre el resto de componentes. Este servicio utiliza un sistema de autenticación basado en tokens y se incorporó en la versión 2012.1 de OpenStack.
- *OpenStack Image Service:* es un servicio para la búsqueda y recuperación de imágenes de máquinas virtuales. Este servicio puede almacenar las imágenes directamente o utilizar mecanismos más avanzados como: usar Object Storage como servicio de almacenamiento, usar Amazon's Simple Storage Solution (S3) directamente, ó usar Object Storage como almacenamiento intermedio de S3.
- *OpenStack Dashboard:* es un panel web para el manejo de instancias y volúmenes. Este servicio es realmente una aplicación web desarrollada en Django que permite comunicarse con las diferentes APIs de OpenStack de una forma sencilla. OpenStack Dashboard es fundamental para usuarios noveles y en general para realizar acciones sencillas sobre las instancias.

4.2.1. ARQUITECTURA CONTEXTUAL DE OPENSTACK

La arquitectura de OpenStack está diseñada para crear un sistema operativo cloud masivamente estable y escalable. Para lograr esto, cada uno de los componentes y servicios que lo constituyente están diseñados para trabajar en conjunto y proporcionar una completa infraestructura como servicio. Esta integración se ve facilitada a través de APIs que ofrece cada servicio. Conceptualmente, se pueden representar las relaciones entre los servicios a través del diagrama que muestra la figura 13:

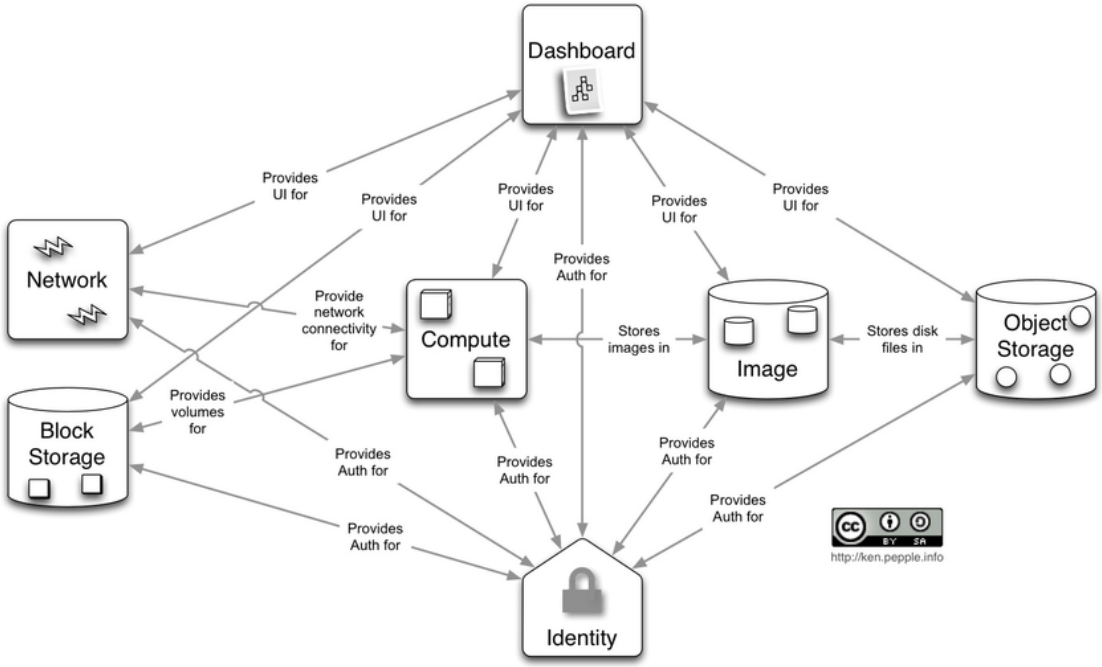


Figura 13: Diagrama conceptual de OpenStack

En la figura quedan claras las siguientes relaciones:

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

Openstack compute consta de varios elementos principales. Un "nodo controlador" que contiene muchos componentes, y representa el estado global e interactúa con todos los demás componentes. Un API de servidor actúa como la interfaz web (Dashboard) quien da el servicios para el controlador del cloud. El controlador proporciona los recursos informáticos del servidor y por lo general contiene el servicio de cómputo, de objetos, opcionalmente proporciona servicios de almacenamiento. Un gestor de autenticación proporciona servicios de autenticación y la autorización cuando se utiliza el sistema de almacenamiento, o puede usar el servicio de identidad (Keystone) como un servicio de autenticación independiente.

Un controlador de volumen proporciona un servicio rápido y permanente a nivel de bloque de almacenamiento para el servidor de almacenamiento. Un controlador de red proporciona redes virtuales para permitir servidores de cómputo de interactuar entre sí y con la red pública. El administrador del cloud selecciona el controlador más adecuado para levantar una instancia.

Openstack se basa en una *responsabilidad compartida basada en arquitecturas*. Puede ejecutar la totalidad de los principales componentes en varios servidores, mediante comunicación a traves de HTTP (HyperText Transfer Protocol), usando un planificador AMQP (Protocolo de resolución de problemas de Avanzada).

4.3. CARACTERÍSTICAS DE LAS VERSIONES DE OPENSTACK

Las versiones de OpenStack (https://wiki.openstack.org/wiki/Release_Naming) están numeradas usando el esquema basado en año, de la forma YYYY.N, por ejemplo la primera fue en 2010 y 2010.1 es el número de versión. El ciclo de desarrollo de las release de OpenStack es de seis meses y se le da saporte durante cinco años. Cada release se identifica por un nombre, los cuales están ordenados alfabéticamente y representan nombre de ciudades, pueblos, condados o calles, donde se realizan las cumbres de diseño de OpenStack, así Austin fue la primer release, Bexar la segunda, Cactus la tercera, etc.

Los componentes existentes en OpenStack, depende de la versión a la cual se hace referencia. A continuación se muestra la tabla 2, la cual discrimina las diferentes versiones y los componentes que posee. Cabe aclarar que cada release, es actualizada continuamente y sostenida en el tiempo, publicando las diferentes versiones.

Versión	**Fecha de Liberación**	**Nombre de los componentes**
Austin	21 de octubre de 2010	Nova, Swift
Bexar	3 de febrero de 2011	Nova, Glance, Swift
Cactus	15 de abril de 2011	Nova, Glance, Swift
Diablo	22 de setiembre de 2011	Nova, Glance, Swift
Essex	5 de abril de 2012	Nova, Glance, Swift, Horizon, Keystone
Folsom	27 de setiembre de 2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Grizzly	4 de abril de 2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Havana	17 de octubre de 2013	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer
Icehouse	17 de abril de 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove
Juno	16 de octubre de 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara
Kilo	30 de abril de 2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic
Liberty	Octubre de 2015	Se agregaran: Zaqar (servicio de colas), Manila (sistema de archivos compartidos), Barbican (administracion de claves).

Tabla 2: Versiones de OpenStack

4.4. COMPONENTES DE OPENSTACK

Si bien los componentes de OpenStack varían con la versión, pues cada release incorpora componentes nuevos; desde Folsom los componentes que siempre están presentes y se consideran los más importantes y fundamentales para la instalación de la infraestructura son:

Componente Horizon (http://docs.openstack.org/developer/horizon/)

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

- Horizon normalmente se despliega a través del módulo de *Apache mod_wsgi*, el cual implementa la interfaz WSGI que permite al servidor Apache ejecutar aplicaciones Python. El código de Horizon está separado en dos módulos Python reutilizables, uno de ellos mantiene toda la lógica de la aplicación y es el encargado de interactuar con varias de las APIs de OpenStack, mientras que el otro es el encargado de la presentación, permitiendo fácilmente la adaptabilidad e integración con la apariencia del sitio web.
- Horizon almacena muy pocos datos, ya que utiliza los datos del resto de servicios.
- Desde el punto de vista de la red, este servicio debe ser accesible por los usuarios a través de la web (tráfico HTTP), de la misma forma que necesita poder acceder a las APIs públicas del resto de servicios. Si además se usa la funcionalidad de administración, necesita además conectividad a las APIs de administración de los endpoints (las cuales no son accesibles por los usuarios finales).

Componente Nova (http://docs.openstack.org/developer/nova/)

No ha cambiado mucho desde las anteriores versiones, se han añadido ciertas mejoras en determinados servicios para la compatibilidad de EC2 y servicios de consola. Nova depende de los siguientes demonios para su funcionamiento:

- *nova-api* es la encargada de aceptar y responder a las llamadas del usuario final a las APIs de nova-compute y nova-volume. El demonio nova-api soporta la API de OpenStack, la API EC2 de Amazon y la API especial de administración (para usuarios con privilegios que realicen tareas administrativas). Además, este demonio es el encargado de la coordinación de ciertas actividades (como la ejecución de una instancia) y la aplicación de ciertas políticas (como la comprobación de cuotas).
- El demonio *nova-compute* es el principal encargado de crear y acabar con las máquinas virtuales (instancias) utilizando para ello las APIs del hipervisor utilizado, los cuales pueden ser: KVM/QEMU, XenAPI para XenServer/XCP y VMwareAPI para Vmware. El proceso completo de creación/destrucción de instancias es bastante complejo, pero la base es muy simple: aceptar acciones de la cola de mensajes y ejecutar un conjunto de comandos del sistema asociados (como lanzar una instancia de KVM), todo mientras que se actualiza el estado en la base de datos.
- *nova-volume* gestiona la creación, conexión y desconexión de volúmenes persistentes a las instancias, de forma similar a como lo realizar el servicio EBS (Elastic Block Storage) de Amazon. Se pueden utilizar volúmenes de diferentes proveedores como iSCSI ó RADOS Block Device (RBD) de Ceph .
- El demonio *nova-network* es muy parecido a los demonios nova-compute y nova-volume. Acepta tareas de red desde la cola de mensajes y realiza ciertas que modifican

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

- El demonio *nova-scheduler* es conceptualmente la pieza de código más simple de Nova. A partir de un mensaje de solicitud de creación de una instancia, determina qué nodo de OpenStack debe ejecutar dicha instancia de acuerdo a un algoritmo previamente seleccionado. La elección se realiza entre todos los nodos que ejecutan el demonio nova-compute.
- La cola de mensajes (queue) proporciona un switch centralizado para el intercambio de mensajes entre todos los demonios. Como cola de mensajes, se utiliza actualmente *RabbitMQ*, pero se puede utilizar cualquier otra cola de mensajes compatible con AMQP como por ejemplo Apache Qpid.
- Una base de datos SQL. El sistema gestor de BBDD será el encargado de almacenar toda la información del cloud así como el estado inicial y de ejecución. Esto incluye los tipos de instancia que están disponibles para el uso, las instancias creadas en un momento determinado, las redes disponibles, los proyectos existentes, etc. Teóricamente, Nova soporta cualquier base de datos soportada por *SQL-Alchemy*, pero las bases de datos realmente utilizadas actualmente son PostgreSQL y MySQL.

Una característica muy importante de Nova, es que ha visto aumentado sus servicios de consola. Los servicios de consola permiten a los usuarios finales acceder a sus máquinas virtuales a través de una consola de texto (caso de GNU/Linux) o consola gráfica (caso de máquinas virtuales Linux y Windows). Este acceso se realiza utilizando un proxy y se basa en los demonios *nova-console* y *nova-consoleauth*.

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

Componente Swift (http://docs.openstack.org/developer/swift/)

Es un almacén de objetos distribuido que cumple una función análoga al Simple Storage Service (S3) de Amazon Web Services. Dada su naturaleza, Swift es altamente escalable tanto en términos de tamaño (varios petabytes) como capacidad (billones de objetos) y cuenta con capacidad nativa de redundancia y tolerancia a fallos.

¿Qué es un objeto para Swift? Este concepto es fundamental para entender este servicio puesto que la administración de objetos es su principal objetivo. Un objeto es una entidad que contiene información, pero a diferencia de los archivos con los que usualmente trabajamos, los objetos no son organizados en una jerarquía. Cada objeto existe en el mismo nivel en un espacio de direcciones llamado pool de almacenamiento. Un objeto no puede ser almacenado dentro de otro objeto. Tanto los archivos como los objetos tienen metadatos asociados a los datos que contienen, pero los objetos son caracterizados por tener metadatos extendidos. Cada objeto tiene asignado un identificador único que permite a un servidor o usuario final recuperarlo sin necesidad de conocer la ubicación física de la información. Esto es muy útil para automatizar y racionalizar almacenamiento de datos en entornos cloud.

Este componente incluye los siguientes componentes:

- Un *servidor proxy*, que acepta solicitudes a través de la API OpenStack Object o directamente a través de HTTP. Una solicitud puede ser una subida de un archivo, modificación de los metadatos o la creación de un contenedor. También acepta solicitudes de descarga de archivos o del listado de objetos del contenedor a través de

un navegador web. Para mejorar el rendimiento, el servidor proxy puede, de forma optativa, utilizar una caché (normalmente memcache).

- Un *servidor de cuentas de usuario*, que gestiona las cuentas de usuario definidas con el servicio de almacenamiento de objetos.
- Un *servidor de objetos*, que gestionan los objetos reales (como archivos o contenedores) en los nodos de almacenamiento.
- Hay también un conjunto de procesos que se ejecutan de forma periódica que realizan ciertas tareas de limpieza sobre los datos. El más importante de estos procesos es el servicio de replicación, los cuales aseguran consistencia y disponibilidad en todo el cluster. Otros procesos periódicos incluyen auditores, actualizadores y reapers.

La autenticación se realiza normalmente a través de Keystone.

Componente Glance (http://docs.openstack.org/developer/glace/)

El proyecto Glance proporciona los servicios necesarios para la búsqueda, localización y obtención de imágenes para las máquinas virtuales del cloud. Al igual que el resto de componentes de OpenStack, Glance posee una *API RESTful* que permite solicitudes tanto de los metadatos de las imágenes para las máquinas virtuales, como la solicitud en sí de una imagen.

Las imágenes que están disponibles a través de Glance, se pueden almacenar en diferentes ubicaciones, desde un simple sistema de ficheros que es la opción por defecto a un sistema de almacenamiento de objetos como OpenStack Swift.

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

- El demonio *glance-api*. Encargado de aceptar peticiones a través de su API para el descubrimiento, recuperación y almacenamiento de imágenes.
- El demonio *glance-registry*. Encargado de almacenar, procesar y recuperar metainformación sobre las imágenes (tamaño, tipo, etc.).
- Una base de datos para almacenar dicha metainformación. De la misma forma que Nova, la base de datos es optativa, pero la decisión siempre gira en torno a *MySQL* ó *PostgreSQL* para entornos en producción.
- Un repositorio de almacenamiento para los ficheros de imágenes. Este almacenamiento es configurable y pueden utilizarse desde directorios locales al propio servicio Swift. Otras soluciones pasan por volúmenes iSCSI, directorios NFS ó CIFS, RADOS block device, Amazon S3 ó HTTP.

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

Componente Cinder (http://docs.openstack.org/developer/cinder/)

Es el servicio de almacenamiento por volumen o bloque (block storage). Permite que las máquinas virtuales puedan acceder e interactuar con dispositivos de almacenamiento virtuales por medio de una interfaz iSCSI.

Brinda una funcionalidad comparable a la que tendrías utilizando un sistema SAN, con sus ventajas (alto desempeño para lectura/escritura) y limitaciones (un dispositivo solamente puede

ser usado por una sola máquina virtual al mismo tiempo). Sin embargo, a diferencia de un SAN convencional, el API de Cinder permite que de manera programática puedas asignar un dispositivo a una u otra máquina virtual conforme lo necesites.

Componente Keystone (http://docs.openstack.org/developer/keystone/)

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

Cada función de Keystone puede conectarse a un backend distinto que permite realizar esa misma función de diferentes formas utilizando un servicio distinto. De esta forma, Keystone puede integrarse fácilmente con diferente almacenamiento como SQL, LDAP ó KVS (Key Value Stores). Esto es muy útil en cuanto a la integración de los servicios de autenticación de OpenStak con los servicios de autenticación existentes en un despliegue en concreto.

Este componente, tiene dos funciones principales:

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

Los conceptos fundamentales de la gestión de usuarios son:

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

Los conceptos fundamentales del registro de servicio son:

- *Servicio:* Corresponde a un componente de OpenStack que puede utilizar el módulo de autenticación.
- *Endpoints:* Representa las URL que nos permiten acceder a las API de cada uno de los servicios o componentes de OpenStack

Componente Neutron (http://docs.openstack.org/developer/neutron/)

Este componente facilita la conectividad entre las instancias entre sí o con la Red Pública. Este componente ofrece tres opciones: Flat Networks (con IPs fijas), Flat Network con DCHP, Vlan Network. Los procesos:

- *neutron-server* acepta pedidos de la API y los ruteo hacia los plugins de Neutron que correspondan.
- Los plugins y agentes de Neutron son los encargados de realizar las tareas, como levantar/bajar puertos, crear redes y subredes y direccionamiento de IPs.
- También dispone de una cola para rutear información entre el *neutron-server* y los distintos agentes.
- Posee una base de datos para almacenar el estado de determinados plugins.

4.5. TERMINOLOGÍA MANEJADA POR OPENSTACK

Una vez visto los componentes o servicios de los que consta OpenStack, también hay que conocer la terminología que maneja para poder gestionar la infraestructura correctamente.

Los elementos gestionados por el servicio Block Storage (Cinder) son:

- *Imagen:* Sistema Operativo previamente configurado que se almacena en Glance. Esta imagen es la que se clona y se instancia. OpenStack utiliza imágenes de sistemas operativos previamente instalados para crear las instancias. Estas imágenes pueden ser sistemas operativos limpios recién instalados o sistemas con muchas aplicaciones completamente configuradas, pueden ser incluso instantáneas (snapshots) de instancias que se están ejecutando o se han ejecutado en alguno de los nodos de computo.

  Las imágenes soportan diferentes formatos. Dichos formatos deben ser adecuados para el manejo del hipervisor. Los diferentes formatos soportados por las imágenes son los siguientes:

  - *raw:* formato de imagen de disco no estructurado. Es el formato básico que puede crearse por ejemplo con la aplicación dd, no está optimizado para su utilización en virtualización, pero puede manejarse con herramientas básicas del sistema (dd, parted, fdisk, mount, kpartx, ...)
  - *vhd:* usado por herramientas de virtualización como VMWare, Xen, Microsoft, Virtual Box y otros.
  - *vmdk:* otro formato usado por herramientas de virtualización como por ejemplo VMWare player.
  - *vdi:* formato soportado por VirtualBox y el emulador QEMU
  - *iso:* formato de ficheros o contenido de un dispositivo óptico como un CDROM o DVD.
  - *qcow2:* formato de disco soportado por el emulador QEMU que permite expandir dinámicamente y soporta Copy on Write.
  - *aki:* indica que la imagen guardada en Glance es una Amazon Kernel Image.
  - *ari:* indica que la imagen guardada en Glance es una Amazon Ramdisk Image.
  - *ami:* indica que la imagen guardada en Glance es una Amazon Machine Image.

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

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

- *Volumen:* Nova-volume es el servicio de OpenStack que permite proporcionar almacenamiento extra a las instancias, de forma similar al servicio Elastic Block Storage (EBS) de Amazon EC2 pero con una implementación distinta. Es una solución iSCSI, proporciona almacenamiento a nivel de bloque utilizando el Gestor de

Volúmenes Lógico (LVM, Logical Volume Manager) de Linux. Al tratarse de almacenamiento a nivel de bloque, un volumen solo puede conectarse con una instancia en un momento determinado, no se trata de almacenamiento compartido como el proporcionado por sistemas de ficheros como NFS ó GlusterFS. Este servicio presenta volúmenes lógicos LVM a los nodos de computación que ejecutan las instancias a través del protocolo iSCSI (Internet SCSI, SCSI sobre TCP/IP).

En resumen, un volumen es la porción de disco en donde se desplegaran las instancias. Existen dos tipos de almacenamiento: el temporal que no guarda los estados de las máquinas, es decir, cada que se reinicie la instancia tendrá los mismos valores y el almacenamiento por volumen, en donde, se crean discos que guardan estados de las máquinas como discos extraíbles que al momento de su implementación son capturas de los sistemas operativos en funcionamiento.

Los elementos gestionados por el servicio Compute (Nova) son:

- *Sabor (flavor):* Posee las características de la máquina virtual de la instancia. Define el número de CPUs virtuales, los cores, la RAM, la capacidad de disco duro, si es disco raíz o disco externo. Los sabores están preconfigurados, pero el administrador del sistema puede crear nuevos sabores.

  Las opciones que se instalan por defecto son cinco: *tiny* (512 MB de RAM, 1GB de disco, 1 core), *small* (2048 MB de RAM, 20GB de disco, 1 core), *medium* (4096 MB de RAM, 40GB de disco, 2 cores), *large* (8192 MB de RAM, 80GB de disco, 4 cores) y x*large* (16384 MB de RAM, 160GB de disco, 8 cores).

- *Instancia:* Es una máquina virtual que el usuario va a usar, se crean a partir de una imagen y un sabor. Cada instancia tiene IP, nombre, claves SSH propias para el usuario. La instancia puede estar ejecutándose, en pausa o parada. También se puede borrar, perdiendo su contenido. La gestión de las instancias es manejada por nova-scheduler y nova-api.

- *Instantánea:* Una de las opciones más interesantes para crear una imagen es a partir de una instantánea de una instancia que se está ejecutando en alguno de los nodos de computo del cloud. Es como una copia de seguridad. De esta manera se pueden duplicar las imágenes y los volúmenes y utilizarlos en instancias. Este procedimiento no sustituye a los mecanismo de copias de seguridad que deben seguir realizándose en el sistema, sino se debe ver como un mecanismo de réplica de imágenes o volúmenes.

- *IP Fija:* Es la dirección IP de la instancia. Se asigna automáticamente por DHCP en la creación de la instancia y tiene la misma duración que la vida de la instancia.

- *IP Flotante:* Es una IP que se asocia a una instancia. Se realiza esta asociación para acceder desde fuera de la instancia, se realiza un NAT. Pueden existir instancias que no tengan una IP flotante, porque no interese que se acceda desde fuera, ya que se puede querer que forme parte de la red interna solamente.

- *Grupo de Seguridad:* Es un conjunto de reglas de firewall, que por debajo está realizando iptables, para controlar el acceso a las instancias mediante las IP Flotantes. Puede configurarse varios grupos de seguridad.

- *Par de claves SSH:* Las imágenes que se descargan desde el repositorio de OpenStack o desde el repositorio de los fabricantes no llevan contraseña por defecto o no es conocida. Al instanciar estas imágenes es necesario un mecanismo de acceso seguro, y éste se realiza desde el par de claves. Si no se posee una clave privada, existe un mecanismo en OpenStack para la generación de la clave pública y la privada. La pública

se queda en el sistema y la privada se descarga al nodo para permitir el acceso mediante SSH.

Los elementos gestionados por el servicio Object Storage (Swift)

- *Contenedor:* Sirve para organizar objetos. Los contenedores no se pueden anidar.
- *Objeto:* Son los elementos que se almacenan, también se almacenan los metadatos. Se manejan grandes volúmenes de datos a través del servicio swift de OpenStack. Es el equivalente a S3 en AW.

Una vez conocida toda la terminología anterior se va a poder entender de forma adecuada el proceso de creación de un proyecto (tenant) usando máquinas virtuales (instancias) que se basan en imágenes, volúmenes o instantáneas y que poseen unas determinadas características (número de CPU, memoria, disco, etc...) que lo marca el sabor. Todas estas máquinas virtuales serán las que pueda acceder el usuario, ya sea a su red interna mediante las IP fijas o a la red externa mediante la IP flotante. Las instancias (máquinas virtuales) están integradas en una red y su acceso se realiza gracias al mecanismo de grupos de seguridad y par de claves. Existe también la posibilidad de manejar grandes volúmenes de datos a través del almacenamiento de objetos o guardar datos en bloques a través del dispositivo de volúmenes.

4.6. VIRTUALIZACIÓN EN OPENSTACK

OpenStack Compute soporta muchos tipos de hipervisores, la mayoría de las instalaciones usan un hipervisor por defecto, pero el administrador puede seleccionar otro en función de requerimientos particulares (en http://docs.openstack.org/developer/nova/support-matrix.html se puede consultar la matriz de características de los hipervisores soportados por OpenStack).

En el nodo donde está instalado y operativo Nova, es donde deben correr todas las máquinas virtuales. Por defecto y dependiendo de la arquitectura, los hipervisores por defecto son QEMU o KVM. Sin embargo es posible cambiar el hipervisor por defecto (esto no es posible en las arquitecturas monolíticas) modificando el archivo *nova.conf* y reiniciando el servicio Nova.

OpenStack soporta los siguientes hipervisores:

- *KVM (Kernel-based Virtual Machine):* Los formatos de disco virtual que soporta se heredan de QEMU, ya que utiliza un programa QEMU modificado para poner en marcha la máquina virtual. Los formatos soportados incluyen imágenes raw, la qcow2 y formatos de Vmware.
- *LXC (Linux Containers):* se usa mediante libvirt, para correr máquinas virtuales basadas en Linux.
- *QEMU (Quick EMUlator):* por lo general se usan cuando la infraestructura se usará con proposito de prueba o desarrollo.
- *UML (User Mode Linux):* por lo general se usan cuando la infraestructura se usará con proposito de desarrollo.
- *VMware vSphere 4.1 update 1 and posteriores:* permite instanciar imágenes basadas en Linux Vmware y Windows mediante conexión con un servidor o directamente con un host ESXi.
- *Xen - XenServer, Xen Cloud Platform (XCP):* se usa para correr máquinas virtuales Linux o Windows. Se debe tener en cuenta que para usar este hipervisor se debe instalar el servicio nova-compute en una máquina virtual con paravirtualización.

- *Hyper-V - Server virtualization con Microsoft's Hyper-V:* se usa para correr máquinas virtuales con Windows, Linux y FreeBSD. Permite correr nova-compute nativo sobre una plataforma de virtualización Windows.

4.7. DISEÑO DE LA ARQUITECTURA OPENSTACK

OpenStack es altamente configurable para satisfacer las diferentes necesidades en función de los requisitos de cómputo, redes y almacenamiento. Los servicios vistos anteriormente deben estar disponibles en el sistema. Existen diferentes formas de agrupar estos servicios para dar consistencia a la infraestructura OpenStack. Estos servicios se configuran o instalan en nodos (las máquinas reales o virtuales) donde van a ejecutar los servicios.

Existen varias agrupaciones de estos servicios para hacer más escalable y distribuida la infraestructura. A continuación, se describen las posibles arquitecturas que se pueden instalar en OpenStack.

- *Monolítico Virtualizado:* en este caso la infraestructura se instala en una máquina virtual y el hipervisor por defecto es QEMU. Este tipo de arquitectura es la más lenta y la menos recomendada pues obliga a la infraestructura a trabajar sobre dos capas de virtualización.
- *Monolítico Nativo:* en este caso, todos los componentes de la infraestructura se encuentran instalados en un único nodo, como lo muestra la figura 14. El principal inconveniente que se presenta con esta arquitectura es que usa como hipervisor por defecto a QEMU, lo cual puede hacer más lenta la administración de las instancias. Una infraestructura montada con esta arquitectura es un cloud de prueba.

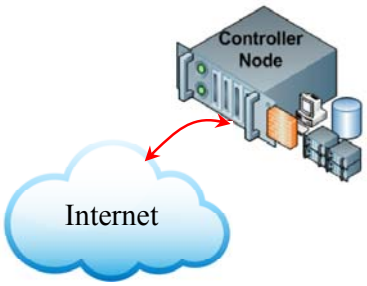


Figura 14: Arquitectura monolítica nativa

- *Dual Node:* en este caso, los componentes de la infraestructura se encuentran instalados en dos nodos, como lo muestra la figura 15. En un nodo se instala Nova y en el otro nodo el resto de los componentes, la principal ventaja de esta arquitectura es que por defecto se instala como hipervisor a KVM, lo cual agiliza la administración y gestión de las instancias; sin embargo la principal desventaja es la degradación de la performance debido a los retardo de las comunicaciones entre los dos nodos. Una infraestructura montada con esta arquitectura es un cloud de producción.

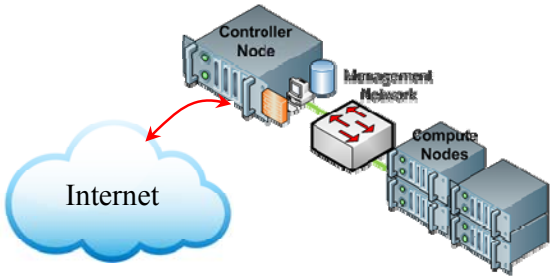


Figura 15: Arquitectura dual node

- *Multinodo:* en este caso, los componentes de la infraestructura se encuentran distribuidos en al menos 4 nodos, como lo muestra la figura 16. En cada nodo se instala cada uno de los componentes (Nova, Neutron, Swift y Cinder), la principal ventaja de esta arquitectura es que se puede seleccionar el hipervisor con el que se quiere trabajar; la principal desventaja es la complejidad del entorno de red y los requerimientos, debido a que es necesario una interconexión de alta velocidad. Una infraestructura montada con esta arquitectura es un cloud de desarrollo, donde se puede trabajar con cluster como servicio.

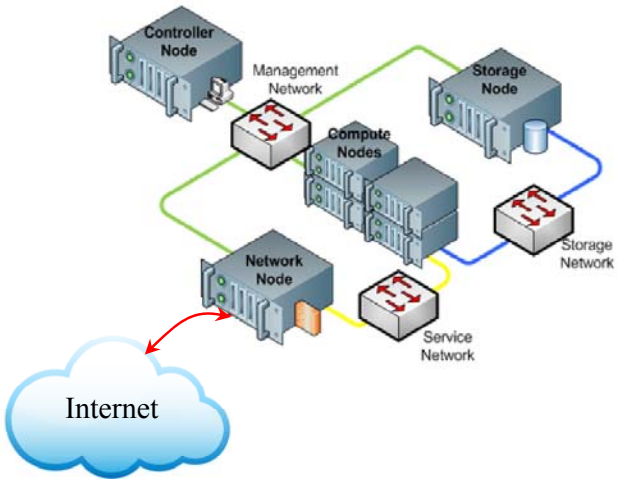


Figura 16: Arquitectura multinodo

4.8. HERRAMIENTAS DE OPENSTACK

Una de las características más interesantes de OpenStack es su arquitectura modular y la posibilidad de poder seleccionar que componentes instalar, según las necesidades del administrador del cloud. Si bien estas son muy buenas ventajas, la desventaja de esto es la dificultad en una orquestación de los componentes y su configuración.

Para solucionar este problema, la comunidad OpenStack ha desarrollado dos herramientas que permiten: la primera provee la posibilidad de realizar una orquestación de servicios semi automatizada mediante DevStack (www.devstack.org) y la segunda, provee un cloud publico OpenStack que permite probar configuraciones con TryStack (www.trystack.org).

4.8.1. DEVSTACK

DevStack es un conjunto de scripts en bash que permiten instalar OpenStack de forma semi automática, el objetivo de DevStack es proporcionar y mantener las herramientas utilizadas para la instalación de los servicios de OpenStack desde el repositorio git. También demuestra y documenta ejemplos de configuración y ejecución de los servicios, así como el uso de cliente de línea de comandos.

La característica más interesante es el soporte de un gran número de opciones de configuración y servicios soportados, siempre teniendo en cuenta que es necesario armar los archivos de configuración necesarios para que se levanten los servicios que se desean instalar.

De estos archivos de configuración el más importante es *local.conf* el cual se encuentra en el directorio raíz de DevStack y contiene los rangos de direcciones fijas y flotantes, las claves de acceso a los servicios y que servicios son los que se van a levantar con su correspondiente token de acceso. A continuación se muestra un archivo *local.conf* estándar:

```
[[local|localrc]]
ADMIN_PASSWORD=secrete
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
SERVICE_TOKEN=a682f596-76f3-11e3-b3b2-e716f9080d50
FIXED_RANGE= x.x.x.x/a
FLOATING_RANGE= y.y.y.y/b
HOST_IP= z.z.z.z
```

Los componentes soportados por DevStack estan definidos en el OpenStack Technical Committee (TC) e incluye:

- *Sistemas Operativos:* Ubuntu, Fedora y RHEL.
- *Mensajería:* Rabbit y Qpid.
- *Base de Datos:* MySQL y PostgreSQL.
- *Servidor Web:* Apache.
- *Administrador de Red:* Nova Network (FlatDHCP) y Neutron (OpenVSwitch).

Los servicios que por defecto estan preconfigurados en DevStack son Identity (keystone), Object Storage (swift), Image Service (glance), Block Storage (cinder), Compute (nova), Networking (nova), Dashboard (horizon), Orchestration (heat). Si es necesario instalar algún otro servicio, se debe incluir la invocación al plugin del repositorio remoto en el script de instalación.

Para poder configurar servicios extras es necesario agregar la línea *enable_service* para cada plugin, como se muestra a continuación:

```
[[local|localrc]]enable_plugin <NAME> <GITURL> [GITREF]
```

donde:

- name: es un nombre arbitrario, por ejemplo glustfs, docker, zaqar, congress, etc.
- giturl: es una URL valida que se puede clonar.
- gitref: es una referencia a git (branch / ref / tag), por defecto se clona el master.

Antes de comenzar a usar DevStack, es necesario que se configure adecuadamente las interfaces de red en el archivo */etc/network/interfaces*. Una vez realizado esto se deberá actualizar el repositorio con:

```
# apt-get update
```

Una vez finalizada la actualización se debe instalar GIT para que permita descargar los archivos necesarios para la instalación de DevStack ya que estos archivos se encuentran en un repositorio GIT: *# apt-get install git*

Los desarrolladores de DevStack se encargan de actualizar estos repositorios a las nuevas versiones de OpenStack. Una vez instalado Git, se debe clonar la herramienta desde el repositorio (https://github.com/openstack-dev/devstack), de la siguiente manera:

```
# git clone https://github.com/openstack-dev/devstack.git
```

Antes de ejecutar las script de instalación se debe elegir la versión de OpenStack que se desea instalar con Devstack (se debe estar dentro de la carpeta devstack), de la siguiente manera:

```
$ git checkout stable/version_a_instalar
```

En este momento ya se puede comenzar con la instalación, mediante la ejecución de la script de inicio: *./stack.sh.*

4.8.2. TRYSTACK

Es un servicio gratuito que provee un ambiente real de OpenStack donde los desarrolladores pueden subir sus aplicaciones y probar cómo funcionan en este ambiente. Vale la pena mencionar que TryStack es tan solo para probar, no para código en producción (ya que las apps son periódicamente eliminadas). Aun así, es una buena opción para aquellos que tienen ganas de probar OpenStack y quieren evitar tener que instalar el stack por su cuenta.

TryStack permite elegir entre una arquitectura x86 o ARM, según lo que se desea probar. Por el momento la versión más actualizada de OpenStack disponible en TryStack es Juno y, si bien algunas funcionalidades incorporadas en los últimos lanzamientos de OpenStack no están disponibles, desde allí podrán sentir lo que es manejarse con una estructura subyacente distinta y más poderosa.

La Arquitectura de TryStack cuenta con 156 cores, 1040 Gb de memoria y 59.1Tb de disco. Con importantes empresas soportando el proyecto como HP, con sus servidores Calxeda-Redstone, Calxeda, Canonical, Dell, NTT, Rackspace..

El sistema como ya hemos dicho es gratuito y se debe entender como un sandbox y no como un servidor de producción y para controlar su uso, existen una serie de normas, entre ellas el auto-borrado de instancias cada 24h, la limitación a Ubuntu Oneiric Ocelot o a Natty Narwhal, el uso de un sistema de créditos y kharma para beneficiar a los usuarios que ajusten el uso de la infraestructura a sus necesidades reales.

Para poder acceder a los recursos de TryStack es necesaria la suscripción a su grupo de facebook, tras lo cual un comité examinara la petición y asignará dado el caso un acceso a la plataforma. También está abierto un proceso para reclutar a colaboradores con la plataforma, realmente muy interesante.

5. INSTALACIÓN DE LA INFRAESTRUCTURA

En función de lo analizado en los apartados anteriores, se ha decidido instalar OpenStack en su penúltima versión estable, denominada Juno cuyos componentes se pueden ver en la figura 17. La selección de la versión se debe a que las diferencias existentes entre las versiones Juno y Kilo, no son relevantes a los objetivos para los cuales se instala la infraestructura, debido a esto no se instalará Trove (Bases de Datos como servicio para soporte de bases de datos SQL y NoSQL)

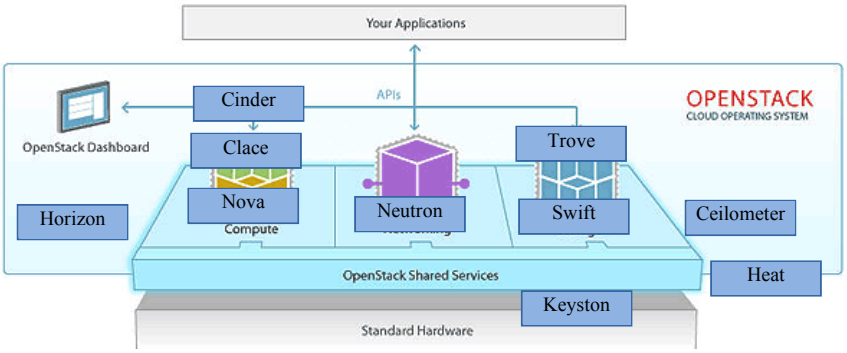


Figura 17: Componentes de la versión Juno

Para este trabajo se ha decidido instalar una topología Monolítica Nativa con el objeto de contar con una infraestructura cloud para prueba de producción. Se realizara la instalación sobre un equipo con arquitectura Intel de 4 cores, de 64 bits, con 8 Gb de RAM y disco de 1Tb, el sistema operativo instalado es Ubuntu 14.04 LTS.

Antes de comenzar con la instalación es necesario que se haya instalado GIT, para dar soporte al clonado de los elementos.

```
# apt-get upgrade
# apt-get install git
```

La mayoría de los servicios de OpenStack usan una *base de datos SQL* para guardar su información. OpenStack provee soporte para MariaDB, MySQL y PostgreSQL, para este trabajo se decidió instalar *MySQL*.

```
# apt-get install python-mysqldb
```

OpenStack usa un *message broker* para coordinar las operaciones entre los diferentes servicios. OpenStack soporta múltiples middleware de mensajería tales como RabbitMQ, Qpid y ZeroMQ. Para este trabajo se decidió instalar *RabbitMQ*.

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

Una vez realizadas estas tareas se debe clonar el repositorio git de Devstack, para la rama de la versión *juno*:

```
$ git clone -b stable/juno https://github.com/openstack-dev/devstack.git
$ cd devstack
```

Antes de comenzar con la ejecución se debe personalizar las distintas opciones de configuración, en el caso particular de la implementación que se realizo fue necesario configurar la interface de red en la correrá el servidor OpenStack. Es recomendable que esta interface se configurada en forma estática, con el objetivo de no generar conflictos debido a que el servidor usa exhaustivamente el networking. Para resolver este tema se decidió configurar la interface *eth1* a tal fin. Para ello se debe editar el archivo */etc/network/interfaces* y modificarlo con el siguiente contenido:

```
auto lo
iface lo inet loopback
auto eth1
iface eth1 inet static
address 192.168.0.19
netmask 255.255.255.0
network 192.168.0.1
```

Una vez configurada la interface de red es necesario crear el archivo de configuración de arranque el cual se deberá guardar en la raíz del directorio */home/usuario/devstack* con el nombre *local.conf* de la siguiente manera:

```
[[local|localrc]]
DEST=/opt/stack
SCREEN_LOGDIR=/opt/stack/logs
# HOST
    HOST_IP=192.168.0.19
# Networking
    FIXED_RANGE=10.0.0.0/24
# Service Up
    disable_service n-net
    enable_service q-svc
    enable_service q-agt
    enable_service q-dhcp
    enable_service q-l3
    enable_service q-meta
    enable_service neutron
    enable_service q-lbaas
    disable  service tempest
    enable_service s-proxy s-object s-container s-account
#ml2
    Q_PLUGIN=ml2
    Q_AGENT=openvswitch
# vxlan
    Q_ML2_TENANT_NETWORK_TYPE=vxlan
# gre
    Q_ML2_TENANT_NETWORK_TYPE=gre
# Credentials
    ADMIN_PASSWORD=openstack
    MYSQL_PASSWORD=openstack
    RABBIT_PASSWORD=openPrstack
    SERVICE_PASSWORD=openstack
    SERVICE_TOKEN=tokentoken
    RECLONE=yes
# Ceilometer
# Enable the ceilometer metering services
    enable_service ceilometer-acompute ceilometer-acentral ceilometers
    -anotification ceilometer-collector
# Enable the ceilometer alarming services
    enable_service ceilometer-alarm-evaluator,ceilometer-alarm-notifier
# Enable the ceilometer api services
    enable_service ceilometer-api
# Swift
    enable_service s-proxy s-object s-container s-account
    SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f6
#scheduler
    enable_service n-sch
    SCHEDULER=nova.scheduler.chance.ChanceScheduler
```

En este momento ya es posible comenzar con la instalación e OpenStack, mediante la ejecución del script de arranque:

 # ./stack.sh

Esta script es un instalador desatendido que realiza la instalación por defecto de *Ceilometer, Cinder, Glance, Heat, Horizon, Keystone, Nova, Neutron* y *Swift*, guardando los valores por defecto en el archivo *stackrc*. Si se desea instalar otro servicio se deberá modificar el archivo de configuración *local.conf*.

Si se ha clonado correctamente la versión de OpenStack y si se ha configurado con los parámetros adecuados el archivo *local.conf*, la instalación se realiza con mínima intervención del administrador, hasta que aparece en consola el cartel que muestra la figura 18:

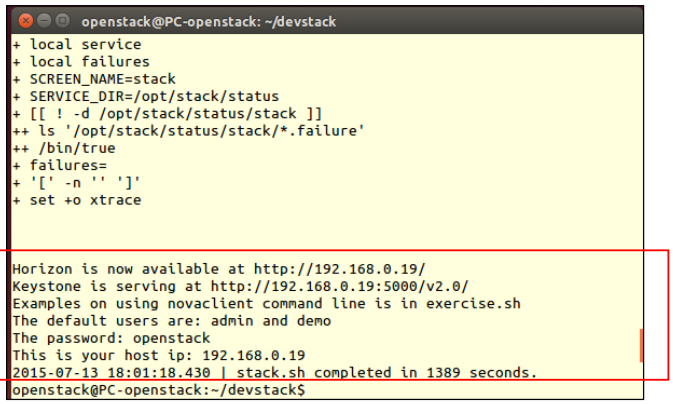


Figura 18: Mensaje de fin de instalación e informe de donde corre Horizon y Keystone

En este punto ya es posible acceder al cloud desde el Dashboard de Horizon, el cual está disponible, para el caso de este trabajo en la dirección IP 192.168.0.19. Accediendo a esta dirección se presenta la pantalla de autenticación como se muestra en la figura 19.

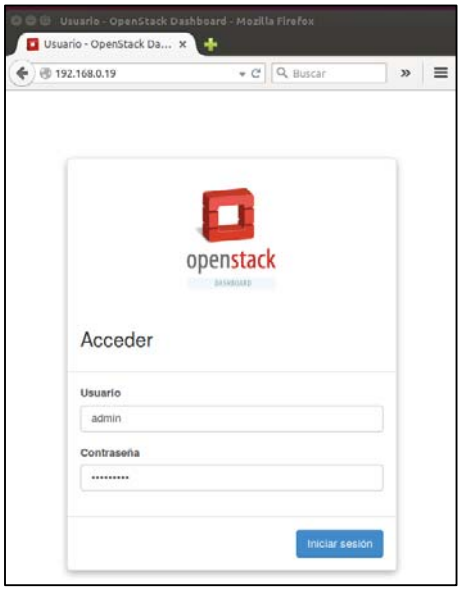


Figura 19: Pantalla de autenticación

Por defecto se han creado dos usuarios: *demo y admin*, Para este trabajo se ha usado el usuario admin, pues es el que permite tener un control total sobre el cloud. Una vez que el logueo ha sido exitoso es posible acceder al panel de administración como muestra la figura 20.

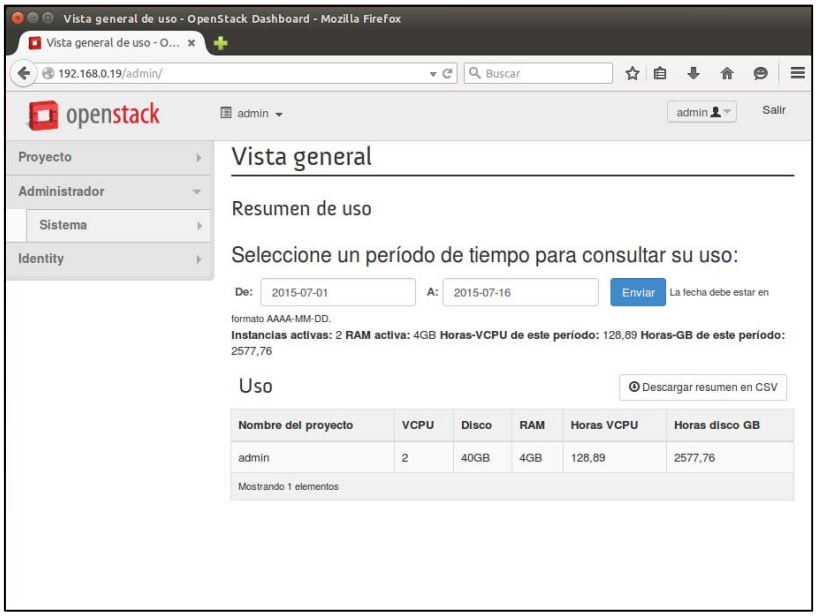


Figura 20: Pantalla de administración

El servicio Horizon permite configurar tres módulos a los cuales se puede acceder:

6. *Proyecto:* permite tener acceso a todos los detalles del proyecto actual. Es posible que un usuario tenga más de un proyecto asociado.
7. *Administrador:* permite acceder a todas las características del cloud. Cuando se está como usuario admin, se puede acceder a la información de todos los proyectos y de todos los usuarios.
8. *Identidad:* Permite crear nuevos proyectos y usuarios que serán administrados por el usuario al que se haya loguedo.

Modulo Sistema:

Accediendo a este modulo es posible acceder a las características del cloud. Es posible definir imágenes, sabores, proyectos y usuarios dentro del cloud.

En la figura 21, se puede ver la *Pantalla Vista General* donde se puede acceder al *resumen de uso* dentro de un periodo determinado. Para este caso el periodo es del 1 al 17 de julio de 2015.

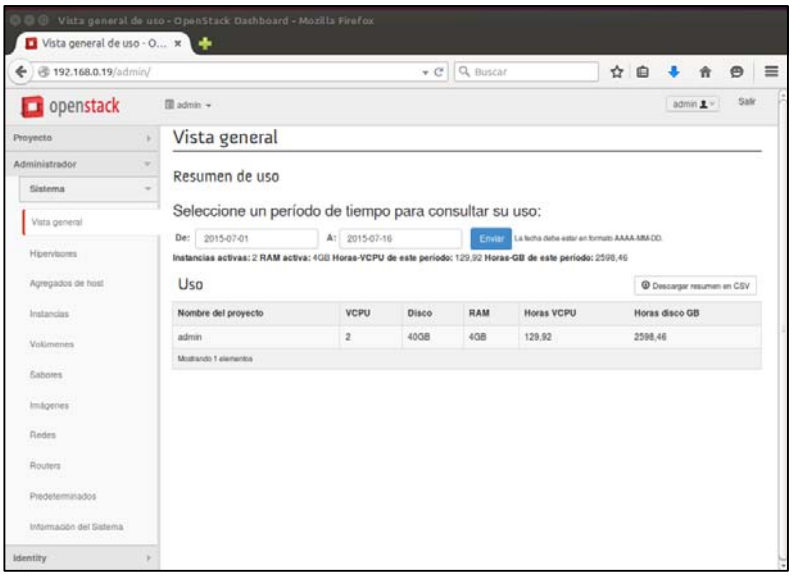
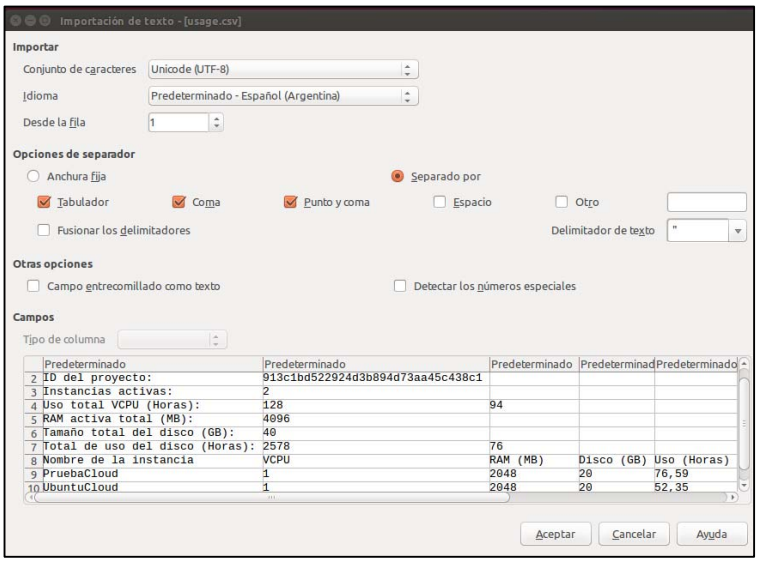


Figura 21: Pantalla de administración

En la figura 22 se puede ver el archivo generado con el resumen de uso del 1 al 17 de julio de 2015. En este archivo se informa instancias activas, uso total de VCPU (Virtual CPU) en horas, RAM activa total, tamaño total de disco y total de uso del disco.



	Predeterminado	Predeterminado	Predeterminado	Predeterminado	Predeterminado
2	ID del proyecto:	913c1bd522924d3b894d73aa45c438c1			
3	Instancias activas:	2			
4	Uso total VCPU (Horas):	128	94		
5	RAM activa total (MB):	4096			
6	Tamaño total del disco (GB):	40			
7	Total de uso del disco (Horas):	2578	76		
8	Nombre de la instancia	VCPU	RAM (MB)	Disco (GB)	Uso (Horas)
9	PruebaCloud	1	2048	20	76,59
10	UbuntuCloud	1	2048	20	52,35

Figura 22: Archivo de resumen de uso en formato CSV

En la figura 23 se puede acceder a la información sobre *Resumen del Hipervisor*, la información que se brinda es la cantidad de VCPU (máquinas virtuales) usadas del total de disponibles, uso de memoria y uso de disco. Además, se ofrece información de los hipervisores disponibles y la cantidad de VCPU para cada hipervisor. Para el caso de este trabajo el hipervisor

disponible es QEMU solamente debido a que esta es una de las características de la topología monolítica nativa.

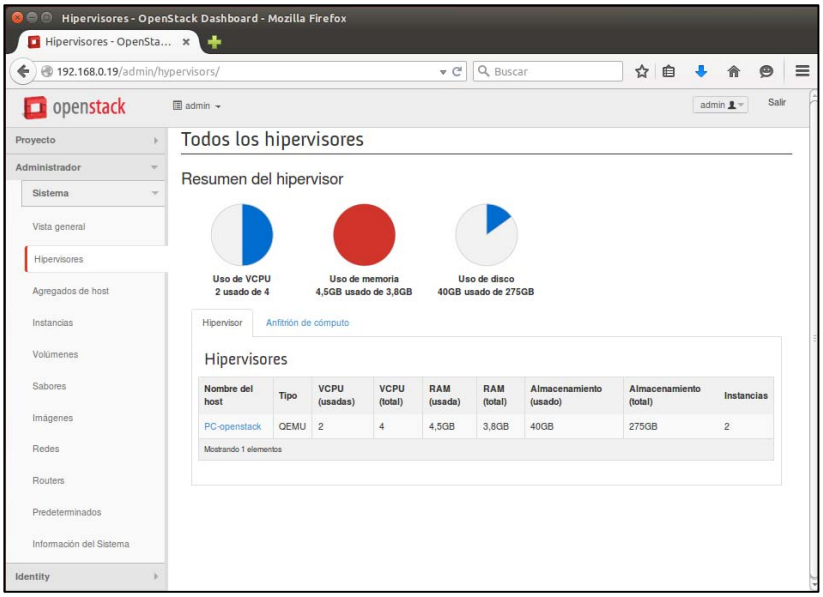


Figura 23: Resumen de Hipervisores

En la figura 24, se muestra la pantalla de *Agregados de Host*, esto permite crear límites virtuales en torno a un grupo de host, de esta manera es posible agrupar lógicamente hosts independientemente de las características que puedan tener en común. Por ejemplo, los hosts no tienen que tener la misma arquitectura, configuración de red o almacenamiento.

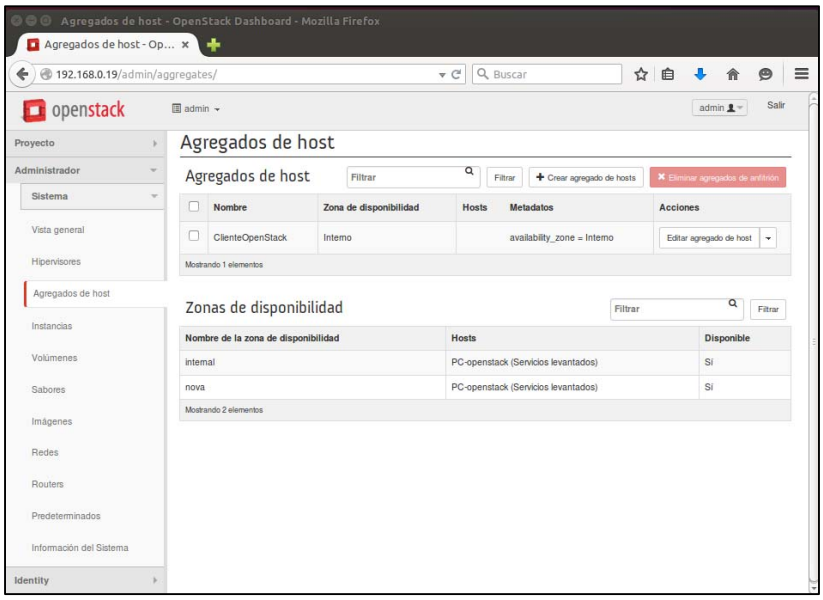


Figura 24: Agregados de host

La figura 25 muestra la interface que permite administrar los *Sabores.* Es posible modificar los parámetros de los sabores por defectos o crear un sabor nuevo.

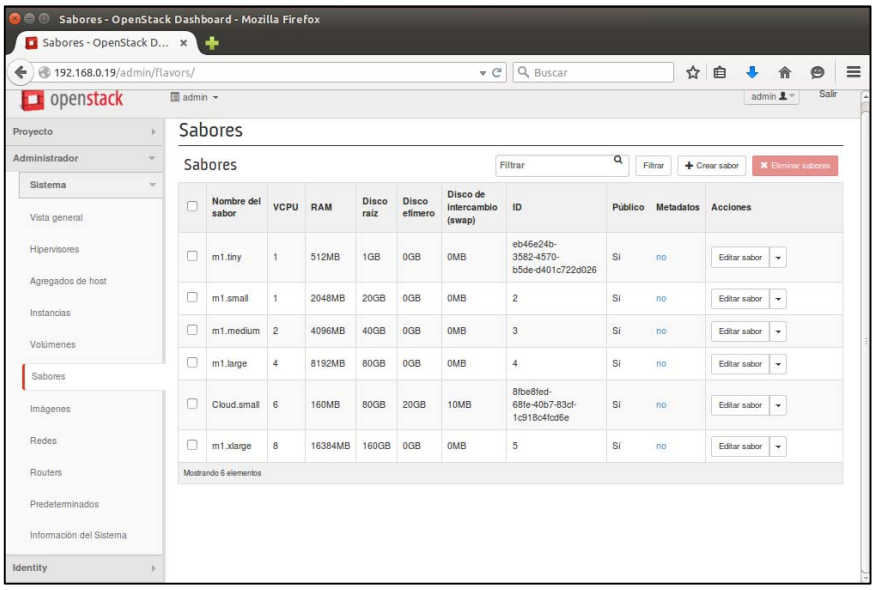


Figura 25: Sabores

Para este trabajo se ha *creado un sabor* nuevo, tal como lo muestra la figura 26, el sabor se denomina *Cloud.small* y sus características son: 6 VCPU, 80 GB de disco y 160 MB de RAM.

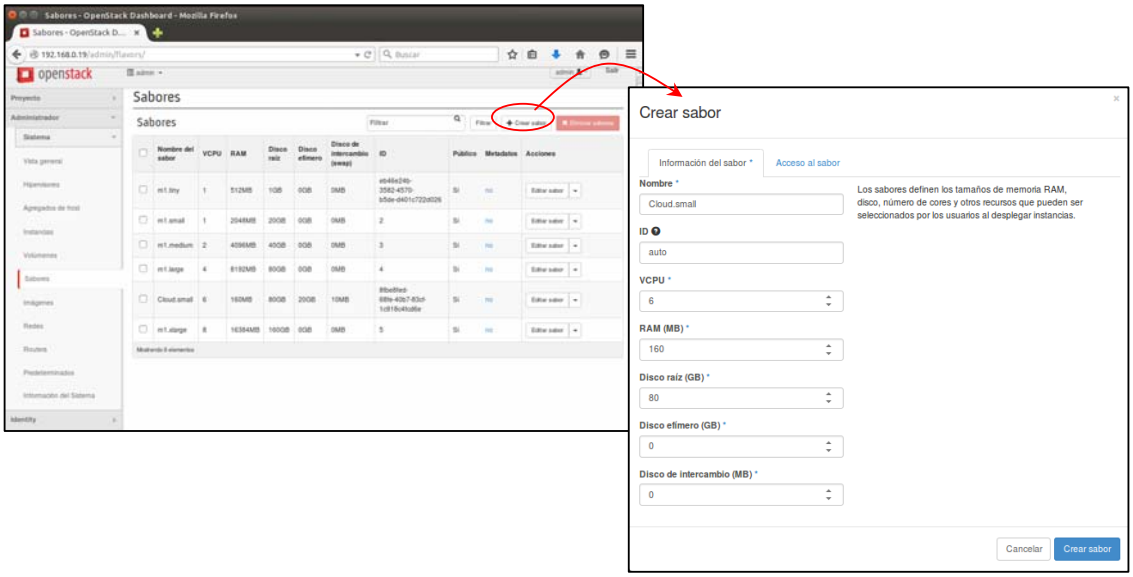


Figura 26: Creación de un sabor

En la figura 27 muestra la interface para acceder a las Imágenes, como ya se ha mencionado antes, se pueden usar las imágenes que se instalan por defecto o se pueden descargar desde Internet.

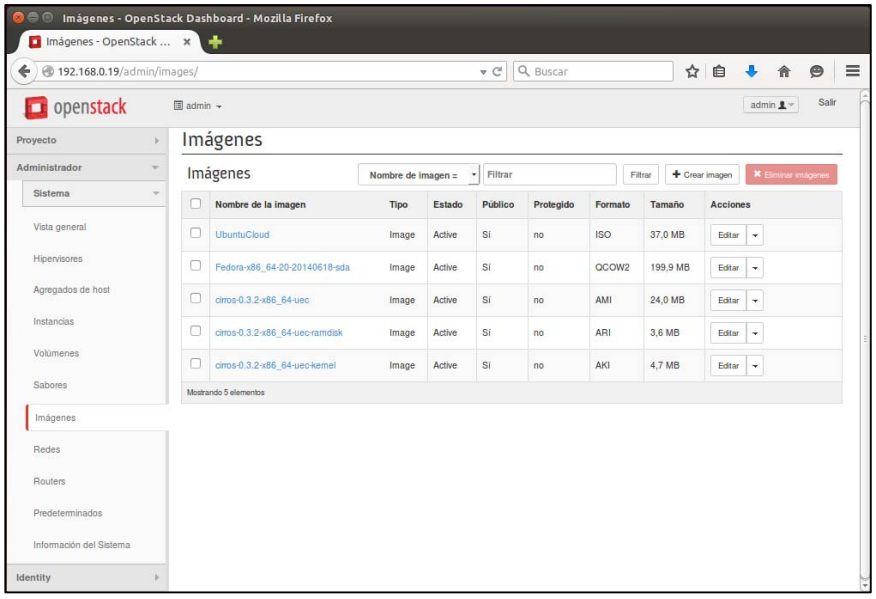


Figura 27: Imágenes

La característica de las imágenes pre cargadas es que poseen muy pocas funcionalidades, es por ello que es conveniente bajar una imagen adecuada a las necesidades del usuarios. Una opción es descargar las imágenes CirrOs desde https://launchpad.net/cirros/+download, la otra opción es bajar una ISO.

Para este trabajo se ha bajado una mini iso de Ubuntu 64-bit 14.04 desde archive.ubuntu.com/ubuntu/dists/trusty/main/installer-amd64/current/images/netboot/mini.iso, a partir de la cual se ha creado la imagen *UbuntuCloud*, como se muestra en la figura 28.

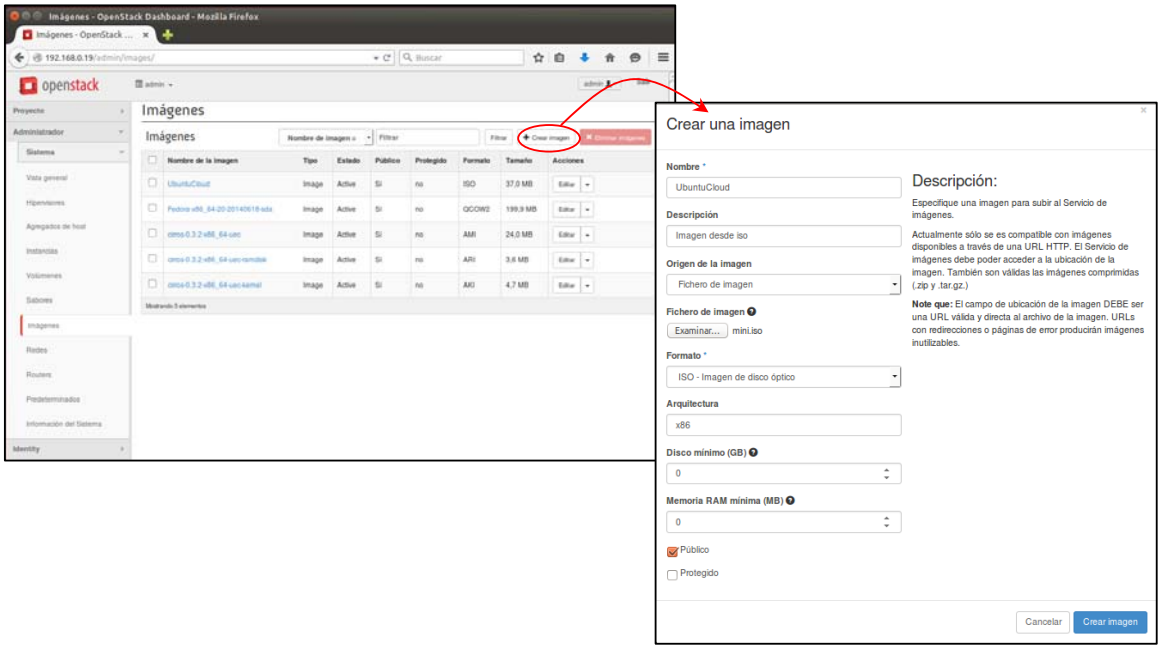


Figura 28: Crear una Imagen

La figura 29 muestra la pantalla de *Instancias*, donde se pueden ver todas las instancias existentes en el cloud, además es posible terminar la instancia o generar una instantánea de ella. También es posible acceder a los detalles de cada instancia.

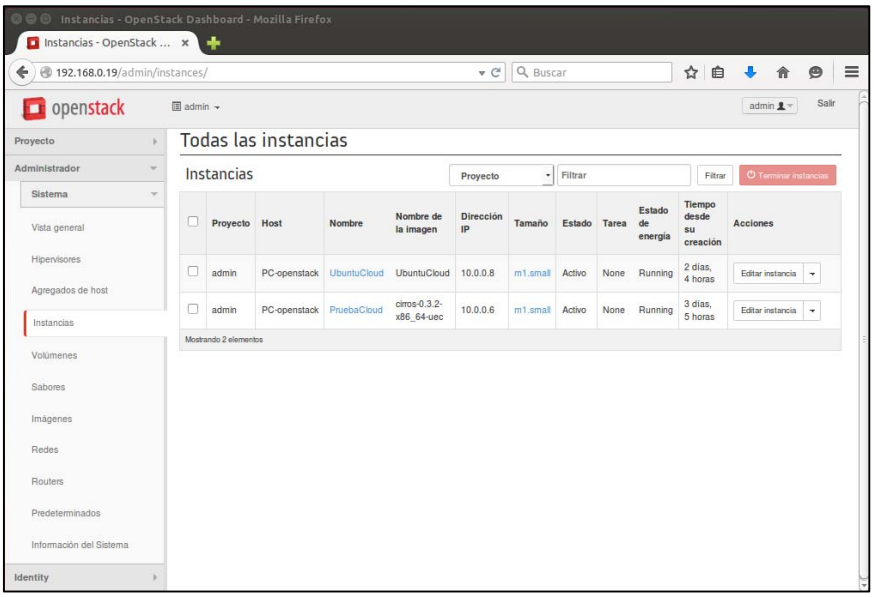


Figura 29: Instancias

Las instancias se crean a partir de una Imagen y un sabor, para el caso de este trabajo se han creado dos instancias: la *PruebaCloud* (a partir de una imagen cirros y un sabor small) y la UbuntuCloud (a partir de la imagen *UbuntuCloud* y el sabor Cloud.small). Esta última instancia es la que se usa para probar el cloud.

En la figura 30, se puede ver la pestaña *Vista General* de *Detalles de la instancia UbuntuCloud*, donde se discrimina nombre, id, estado, fecha de creación, tiempo de encendido, especificaciones (sabor, RAM, disco, VCPU), dirección IP, etc.

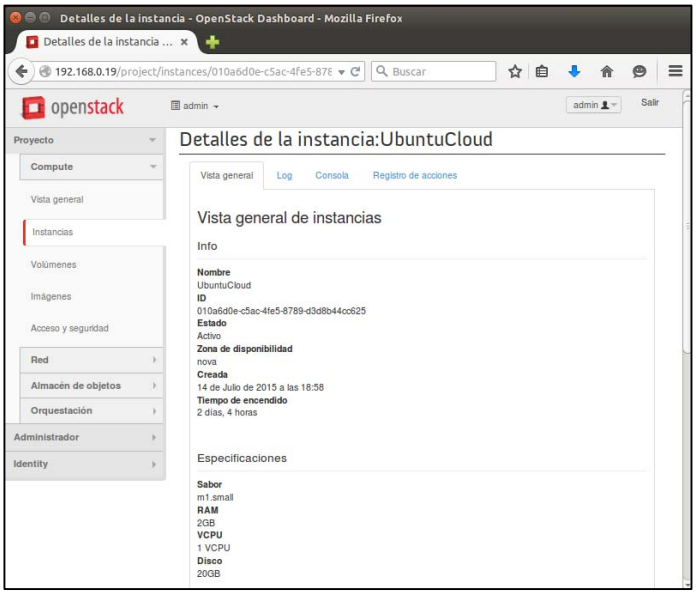


Figura 30: Detalle de la Instancia – Vista General

En la figura 31, se puede ver la pestaña *Log* de *Detalles de la instancia UbuntuCloud*, donde se puede acceder a la Consola de log de la instancia donde se guarda todas las actividades realizadas para que la instancia sea lanzada.

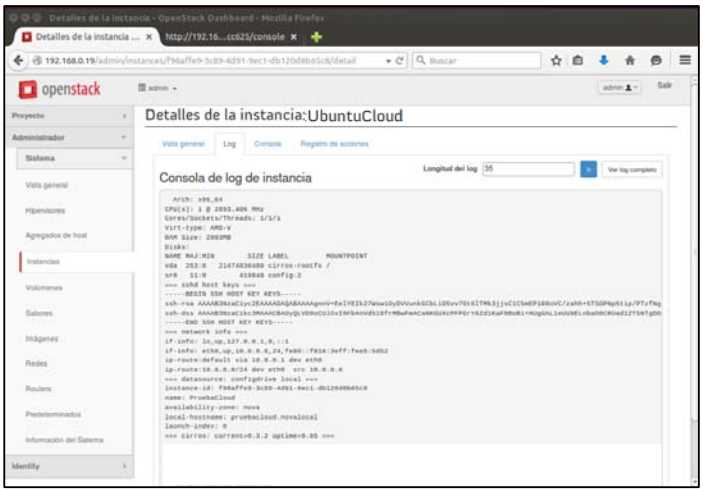


Figura 31: Detalle de la Instancia – Log

En la figura 32, se puede ver la pestaña *Consola* de la instancia *UbuntuCloud*, donde se puede acceder a la instancia en modo consola, en este punto es posible usar la instancia para producción, pues brinda un entorno Linux.

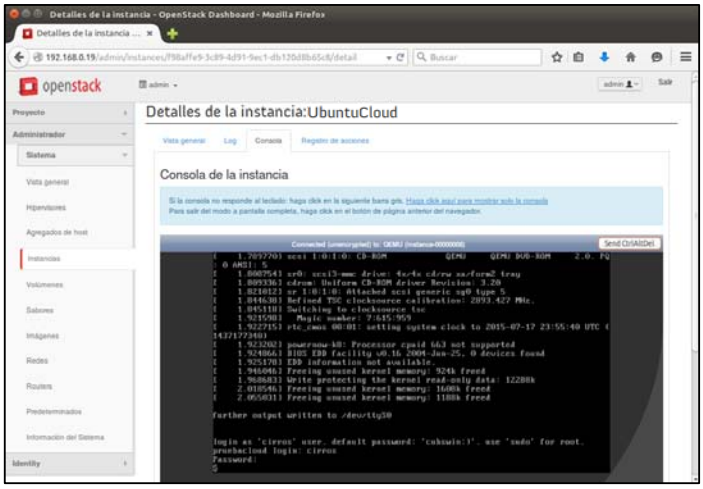


Figura 31: Detalle de la Instancia – Consola

Es importante asegurarse de que se puede acceder a las instancias a través de SSH y que responden a los mensajes ICMP de solicitud de eco (pings). Esto se puede conseguir a través del comando *nova secgroup-add-rule.*

```
# nova secgroup-add-rule default icmp -1 -1 -s 0.0.0.0/0
# nova secgroup-add-rule default tcp 22 22 -s 0.0.0.0/0
```

Si tras la ejecución de estos comandos no se puede acceder a la instancia, se recomienda ejecutar los siguientes comandos:

```
# killall dnsmasq
# service nova-network restart
```

En la figura 32 se puede ver los valores predeterminados del cloud, estos valores pueden ser modificados con el objetivo de modificar las cuotas máximas de cada recurso del cloud.

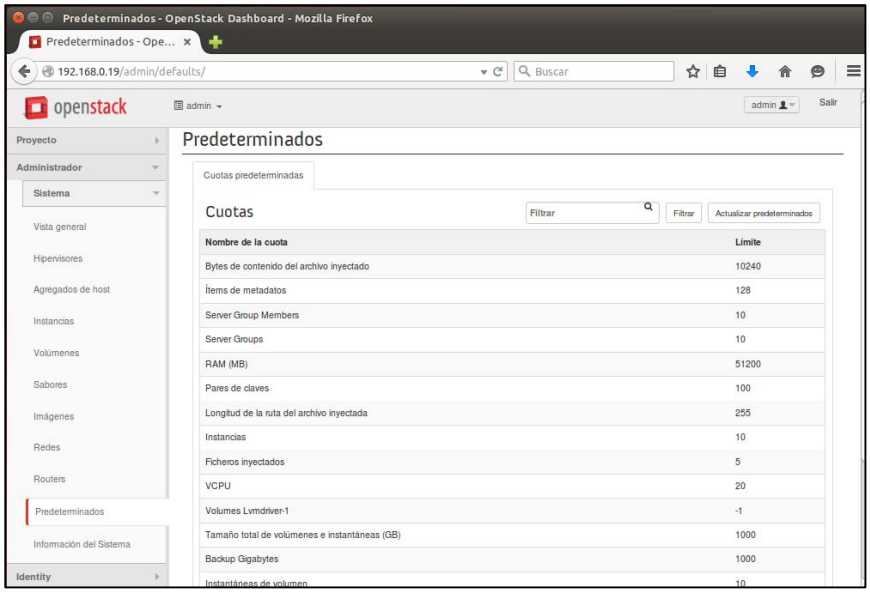


Figura 32: Predeterminados

En la figura 33, se puede acceder a la información sobre los servicios que están habilitados en el cloud y en que dirección IP están disponibles. Mediante las diferentes pestañas se puede acceder a los servicios generales, de cómputo, de almacenamiento y de red.

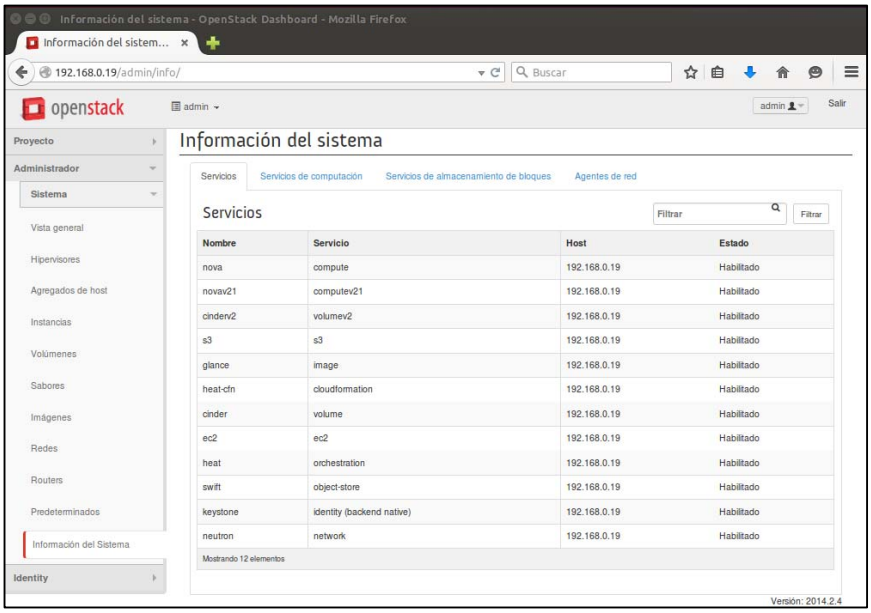


Figura 33: Información del Sistema

Modulo Proyecto:

Accediendo a este modulo es posible acceder a tres secciones: Cómputo, Red y Orquestación. La sección *Cómputo* permite definir a las características del cloud. Es posible definir imágenes, sabores, proyectos y usuarios dentro del cloud. La sección *Red*, permite crear redes públicas, privadas y virtuales mediante vSwitch. La sección *Orquestación*, permite definir las pilas de recursos que se deben orquestar para cada proyecto, esta sección es de uso obligatorio cuando se tiene habilitado el servicio Ceilometer que permite la facturación del uso de recursos; para el caso de este trabajo esta sección no se usará.

En la figura 34, se puede acceder a la pantalla de Vista General del proyecto. Esta pantalla brinda información del uso de los recursos asignados a cada proyecto en función de las instancias que se hayan lanzado.

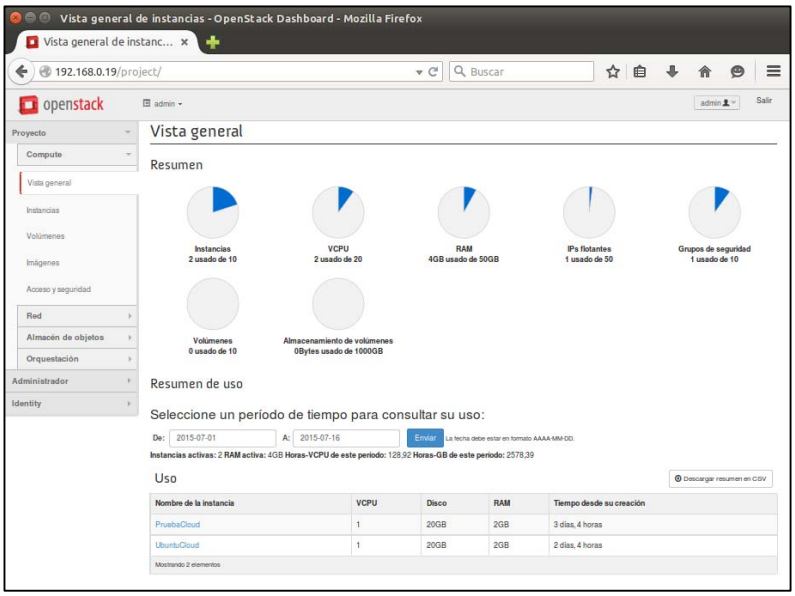


Figura 34: Información General

Como se analizó en el *Modulo Sistema*, las instancias se crean a partir de una imagen y un sabor. El administrador del cloud crea una cuota con las cantidades de instancias que un usuario puede lanzar, además, el administrador puede crear una instantánea de la instancia, terminarla, pausarla o migrarla. En la figura 35 se puede ver como lanzar la instancia *UbuntuCloud*.

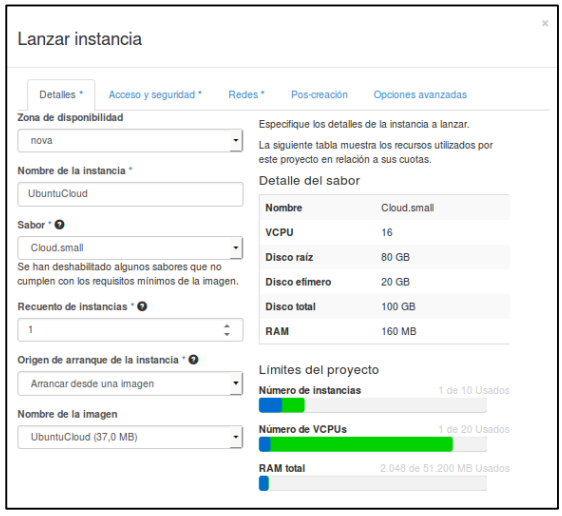


Figura 35: Lanzar Instancias

Unos de los aspectos a los cuales OpenStack le pone especial énfasis es el manejo de la seguridad, por ello además del manejo de los usuarios y sus permisos de acceso a cada servicio del cloud, OpenStack administra *Grupos de Seguridad*. Estos grupos son conjuntos de reglas de filtros de IP que son aplicadas a la configuración de red para la máquina virtual. Después de que el grupo de seguridad es creado, es posible agregar reglas al grupo, este proceso se puede ver en la figura 36. Cada proyecto posee su grupo de seguridad y este es válido para todas las instancias que en él se ejecuten, para el caso de este trabajo se ha creado el grupo de seguridad *GrupoCloud* para el cual se van a administrar las reglas de trafico permitido entrante y saliente.

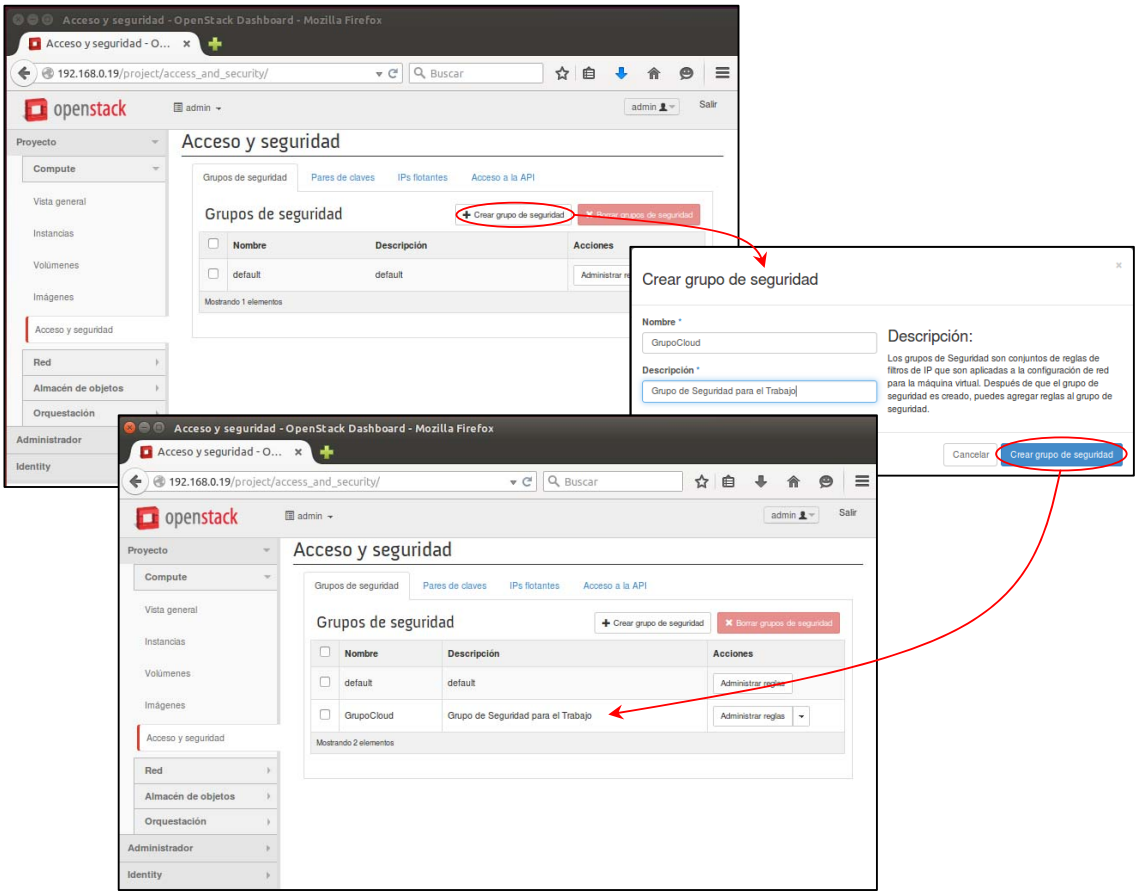


Figura 36: Grupos de Seguridad

En la figura 37, se puede ver que se han agregado dos reglas de tráfico entrante, la primera permite el tráfico entrante de tipo ICMP con el objeto de poder hacer un ping a las instancias; la segunda permite el tráfico entrante desde el puerto 22 (SSH) de TCP para poder acceder en forma remota a las instancias. Esta última regla es muy importante pues los pares de clave SSH, son el método utilizado para ingresar en una instancia una vez que se ha lanzado; si no se especifican estas claves, cualquier usuario puede ingresar a la instancia sin restricciones.

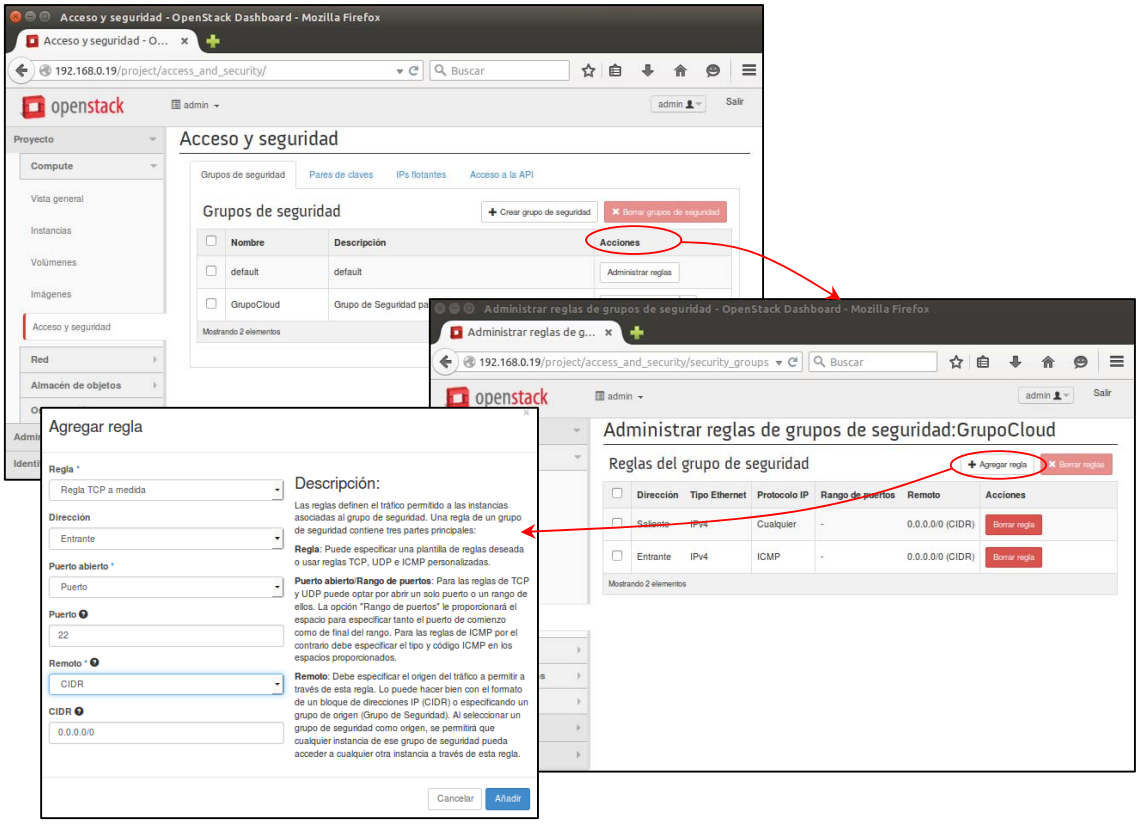


Figura 37: Administrar reglas

En la figura 38 se puede ver el proceso de obtención de un par de claves SSH para un grupo de seguridad *GrupoCloud,* el cual será usado para lanzar las instancias del proyecto.

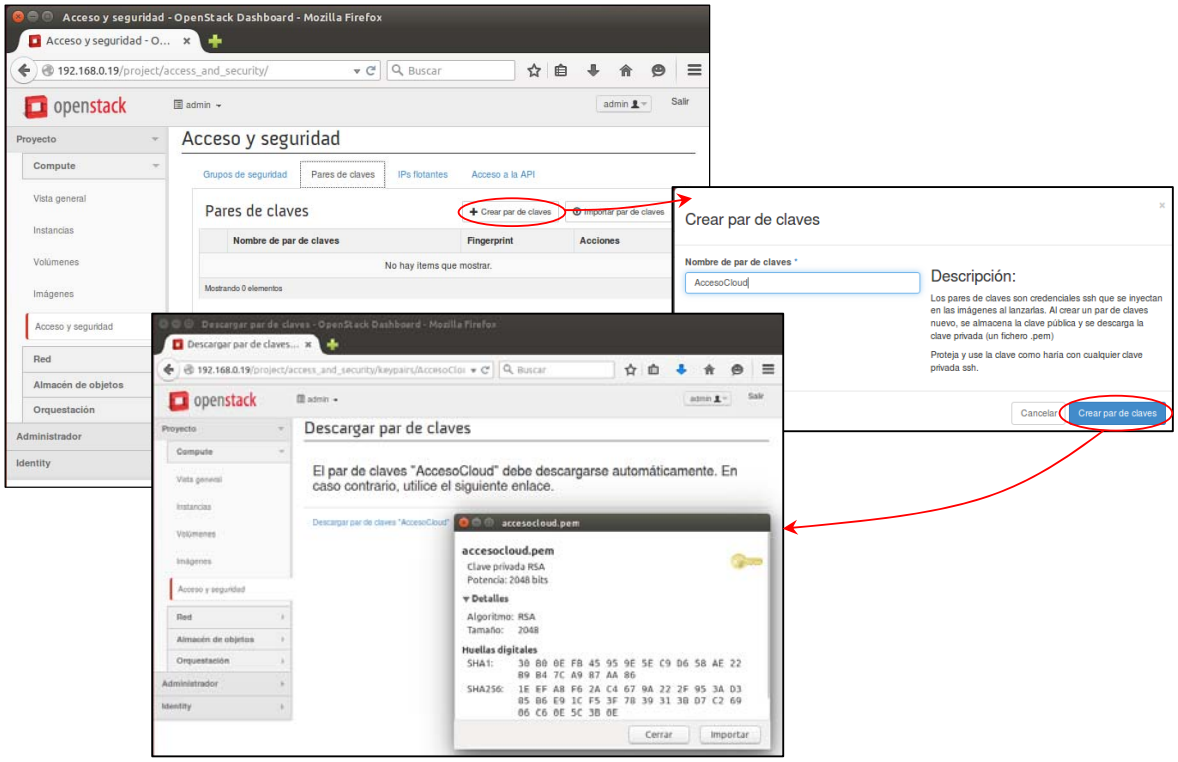


Figura 38: Obtención de Par de claves

Otro aspecto importante, es la forma que tienen las instancias para comunicarse entre si dentro del cloud y para que sean accedidas desde afuera por SSH. Las direcciones IP privadas que se le asigna a una instancia cuando se lanza sirven solo para comunicación interna. Para poder acceder a una instancia desde fuera del cloud es necesario asociarle una dirección IP Flotante. Para lograr esto, la figura 39 muestra el proceso de asignación de IP Flotante a un proyecto, para que luego se le puedan asignar a las instancias lanzadas, hay que tener en cuenta que se deben asignar al proyecto tantas IP Flotantes como instancias tenga.

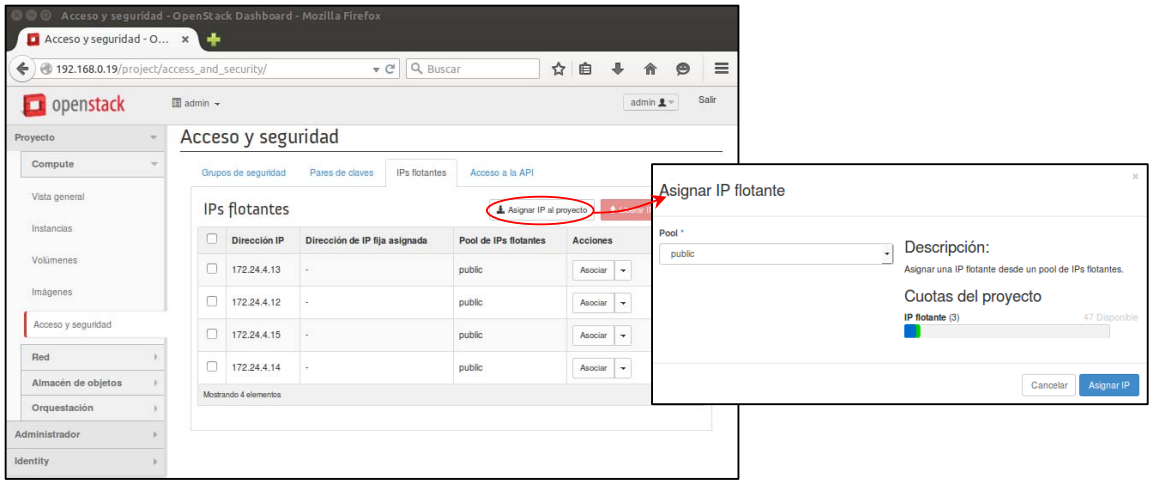


Figura 39: Asignación de IP Flotante

Una vez que se le ha asignado a cada instancia una IP Flotante, ya es posible que sea accedida por SSH en forma remota desde cualquier PC. Es posible testear la conectividad de la instancia *UbuntuCloud*, mediante el envío de ICMP. Cabe aclarar que este tipo de topología instalada solo permite el acceso a las instancias manipulando las IP privado, pues el nodo donde se instaló el cloud posee una sola placa de red.

Modulo Identidad

Este modulo, está fuertemente relacionado con los servicios ofrecidos de Keystone, el cual provee la integración de los servicios de OpenStack proporcionando servicios de autenticación, gestión de tokens y mantenimiento del catálogo y el repositorio de políticas de identidad. Si bien el manejo de la seguridad es transversal a toda la arquitectura, en este módulo se administran los usuarios y los proyectos que puede tener el cloud.

Keystone introduce en OpenStack un sistema de autenticación basado en tokens, de manera que todos los elementos del cloud (usuarios y servicios principalmente), no se autentican directamente unos a otros, sino que lo hace con un actor intermedio mediante tokens, este actor intermedio encargado de verificar la autenticidad de cada uno de los elementos es Keystone. Un proceso típico de autenticación en OpenStack puede verse en la figura 40, en la que se muestran los pasos que se dan desde que el usuario se acredita frente a Keystone hasta que lanza una instancia.

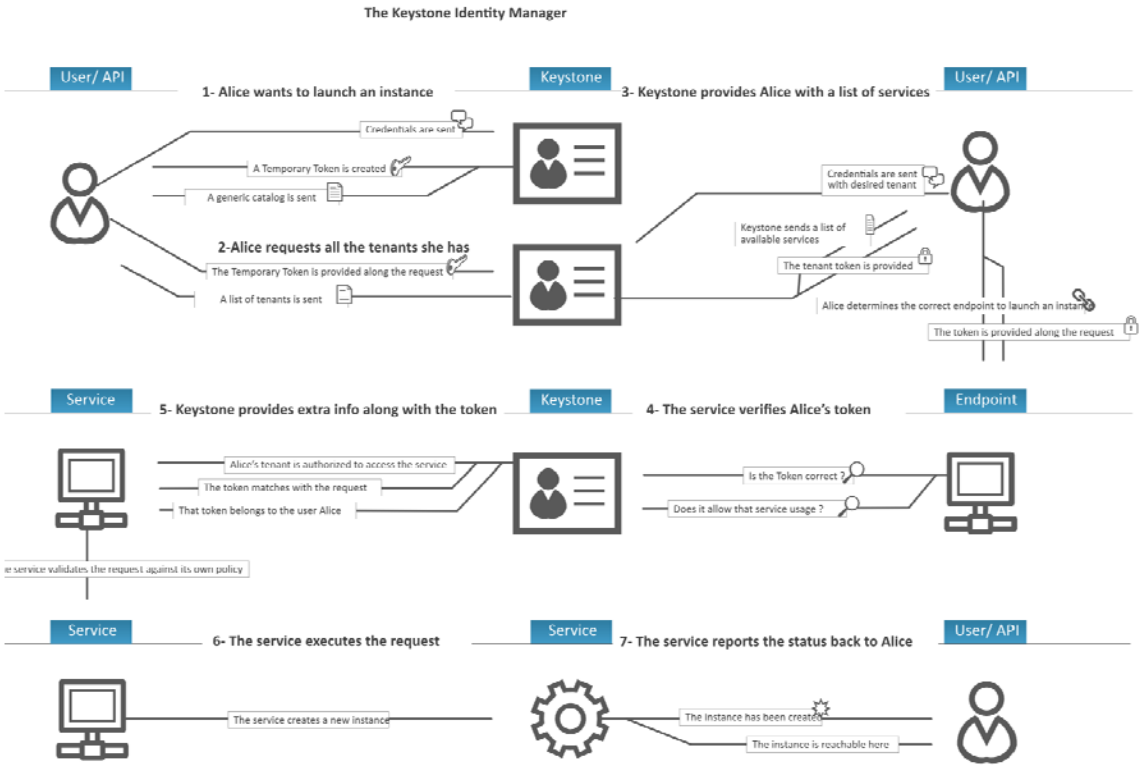


Figura 40: Diagrama que describe el funcionamiento típico de keystone

Durante el proceso de instalación del cloud, se creó por defecto un usuario con rol de administrador y proyecto asociado. Cuando se quiera crear un usuario nuevo es necesario crear primero el proyecto que tendrá asignado ese usuario. En la figura 41, se puede ver el proceso de creación de un proyecto. Este proceso involucra tres pasos:

- *Información del Proyecto:* En esta pestaña se define el nombre del proyecto y una breve descripción a los objetos que se tenga mas informacion del objetivo del proyecto.
- *Miembros del Proyecto:* Aquí se define los usuarios físicos que van a tener derechos para acceder al proyecto y los servicios a los cuales se puede acceder.

- *Cuota:* Esta pestaña permite definir los valores máximos de los recursos a los cuales los usuarios pueden acceder, en función de estos valores el cloud crecerá elásticamente hasta estos máximos.

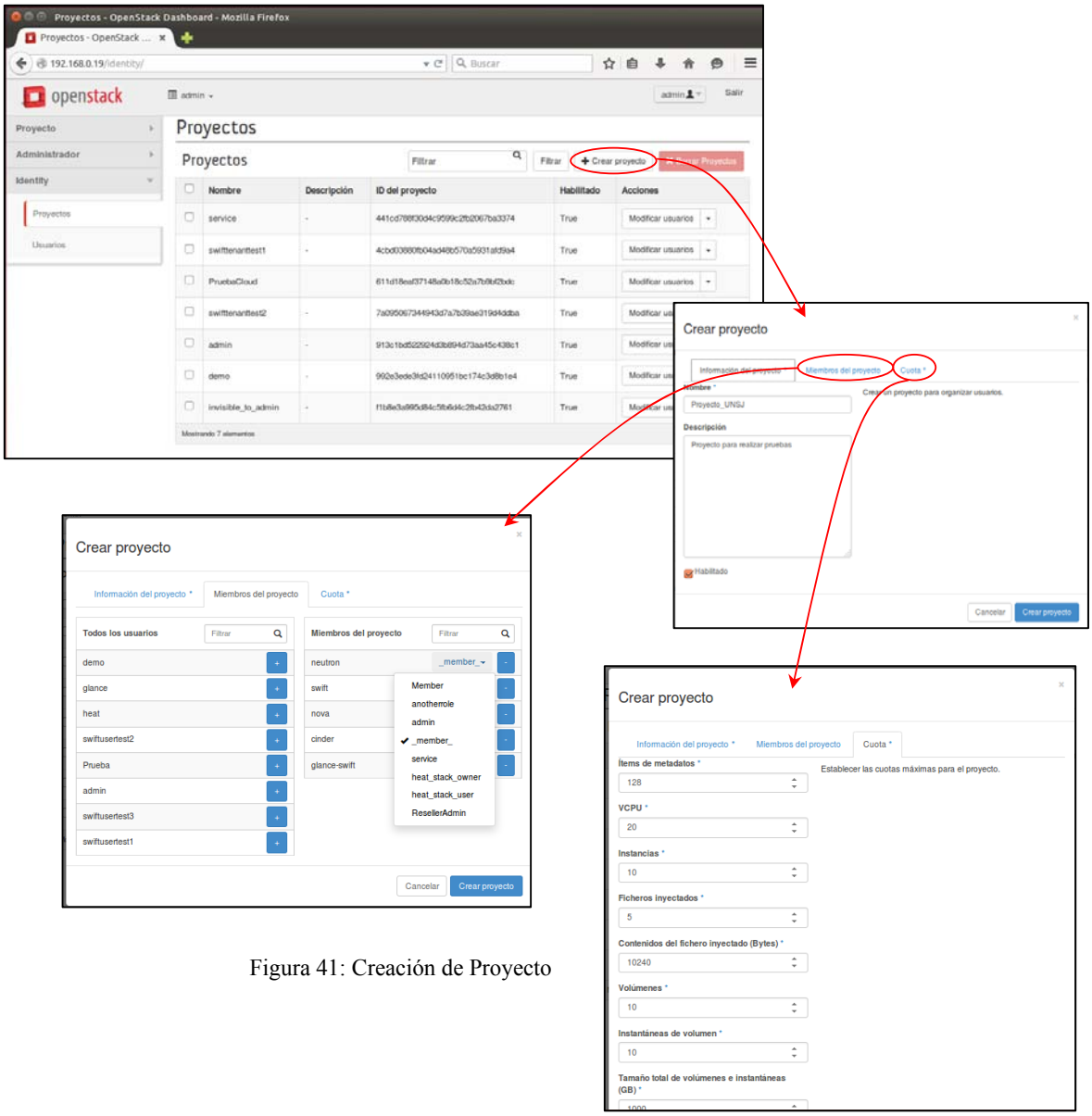


Figura 41: Creación de Proyecto

Una vez que se cuenta con el proyecto, se pueden asignar los usuarios que estarán autorizados para usarlo. En la figura 42 se puede ver el proceso de creación de usuarios, cabe aclarar que para OpenStack, todo es un servicio que posee asociado un usuario.

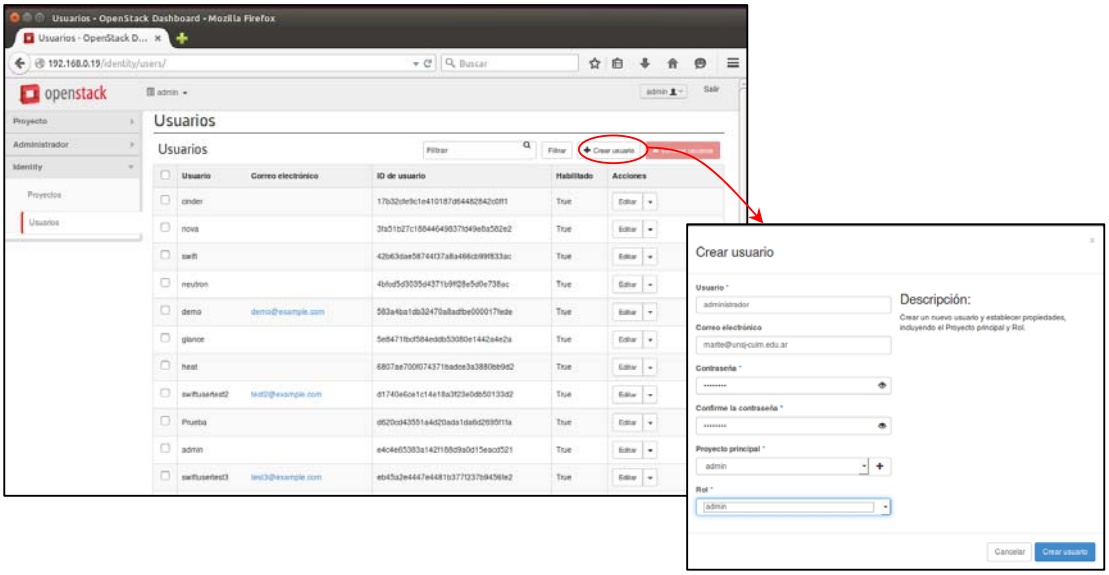


Figura 42: Creación de usuarios

Una vez que se han creado los usuarios, el proceso de autenticación se realiza mediante el modulo Keystone y sigue los siguientes pasos.

- *Usuario envía credenciales a través de la API*
- *Keystone le envía un token temporal y le envía un catálogo de los servicios.*
- *Usuario pregunta por los proyectos a los que puede acceder.*
- *Keystone le responde con la lista de proyectos a los que puede acceder el usuario.*
- *Usuario envía credenciales con su correspondiente proyecto.*
- *Keystone envía al usuario los servicios que tiene disponible dentro de ese proyecto y el token de este.*
- *El usuario determina el endpoint correcto para iniciar una instancia a la vez que se le provee a este del token recibido anteriormente por Keystone.*
- *El endpoint pregunta al modulo Keystone si es correcto el token recibido y si se le permite usar el servicio por el que envía la petición.*
- *Keystone le dice al servicio que el usuario está autorizado para solicitar ese servicio y que los tokens coinciden.*
- *El servicio crea una nueva instancia y le envía la información al usuario.*

6. PRUEBA DE LA INFRAESTRUCTURA

Una vez que la infraestructura está instalada y configurada, se pueden acceder a las instancias de los recursos mediante SSH desde cualquier computadora conectada a la red o directamente en la computadora donde está instalada la infraestructura.

Previo a realizar la prueba se levantó la instancia UbuntuCloud, la cual se mostró el proceso de creación en el apartado anterior. Esta instancia se realizó a partir de una imagen con mini iso de Ubuntu 14.04 de 64 bits y un sabor con 8 Gb de RAM y 6 cores; y como hipervisor por defecto Qemu para manejar las instancias dentro del cloud.

La prueba de ejecución de la infraestructura consistió en correr un algoritmo que permite la multiplicación de matrices. Para lograr esto se usó una implementación en C y el compilador por defecto gcc. En la figura 43 se puede ver la instancia corriendo, con los resultados del algoritmo para matrices de 200x200 y de 500x500.

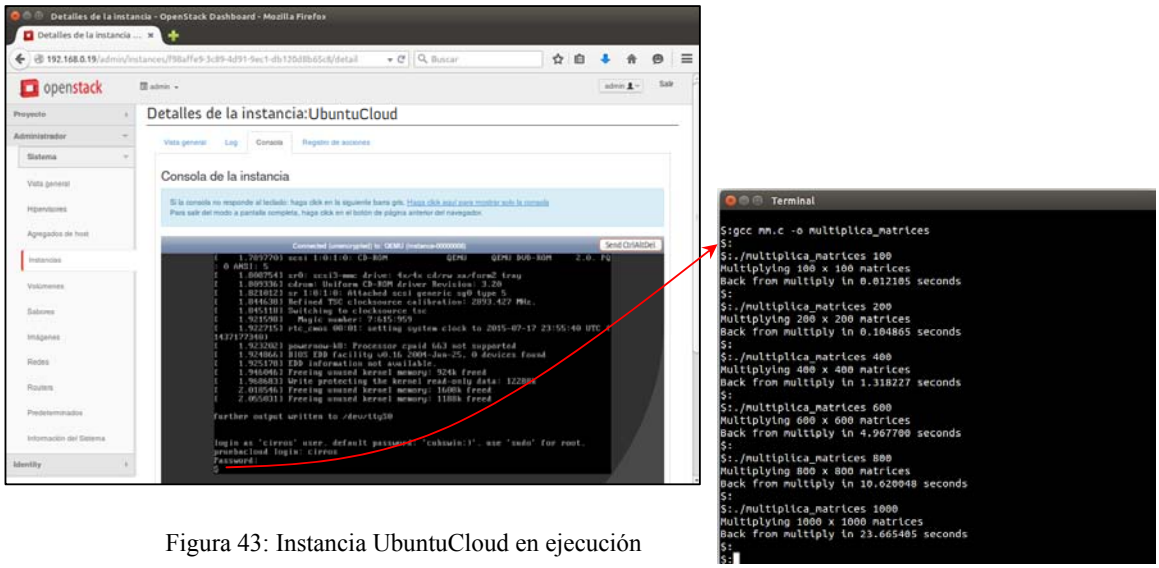


Figura 43: Instancia UbuntuCloud en ejecución

Un aspecto de sumo interés es remarcar que la infraestructura se instaló sobre una topología monolíca, lo cual implica que por defecto instala Qemu. Este hipervisor es adecuado cuando se quiere montar una infraestructura cloud de prueba, donde el principal objetivo es comenzar a familiarizarse con la infraestructura, su configuración y gestión; pero no se recomienda su implementación en entorno de producción debido a que genera mucha latencia en el manejo de los recursos virtualizados.

7. CONCLUSIONES Y FUTURAS AMPLIACIONES

7.1. CONCLUSIONES

Cloud Computing es un modelo de computación que permite que los recursos computacionales, tales como infraestructura, aplicaciones, software o procesamiento puedan ser ofrecidos y consumidos bajo demanda como un servicio más en Internet. Cloud Computing ha dejado de ser una *buzzword,* para pasar a ser uno de los puntos de interés de los especialistas en IT.

Cloud computing es un tecnología que está en constante evolución, prueba de esto es la apuesta que grandes empresas, tales como IBM, Google, Amazon, Microsoft, Red Hat, VMWare, entre otros están haciendo para ofrecer soluciones que saquen provecho de sus beneficios. Sin embargo, y a pesar de las ventajas que ofrecen las soluciones públicas, las organizaciones están demandando mayor grado de seguridad y privacidad en sus datos, por ello es que están tomando gran impulso las cloud privadas. De los productos que ofrece el mercado para implementar una infraestructura cloud privada, los productos open source son las que mayor aceptación están teniendo debido a la robustez, el apoyo de empresas líderes en el área (como Canonical), la disponibilidad del código en repositorios públicos (como Git o GitHub) y la comunidad de desarrolladores que constantemente se encarga de mantener actualizaciones.

La posibilidad de instalar, configurar y gestionar, en el ámbito académico una infraestructura de cloud privado open source, permitirá contar con herramientas de investigación en temas de gran actualidad. El contar con un cloud de estas características se plantea como una paulatina migración hacia una infraestructura distribuida de recursos virtuales con escalada elástica.

En el mercado de la infraestructura cloud existen una gama variada de tecnologías de las cuales en este trabajo se abordan las soluciones Open Source para cloud privadas. Una solución open source tienen sentido en aquellas organizaciones que quieran reducir los costos y obtener una infraestructura más flexible y elástica. Una solución de este tipo permitirá lograr costos reducidos, rápida innovación y licenciamiento. Teniendo en cuenta estos aspectos se considera que las soluciones open source ofrecen flexibilidad y un menor costo, la oportunidad de una innovación y desarrollo más rápido debido a ciclos de testing y desarrollo más cortos.

En este trabajo se han estudiado las cuatro soluciones open source con mayor penetración en el mercado: Eucalyptus, Open Nebula, OpenStack y CloudStack. Como se analizó en la sección 3, *OpenStack* es el producto más adecuado para implementar una infraestructura cloud debido a sus características técnicas, soporte de documentación, actualizaciones continuas y por contar con la mayor aceptación entre la comunidad de usuarios y desarrolladores.

OpenStack es una plataforma de cloud computing íntegramente open source, cuyos objetivos de creación fueron la necesidad de contar con estándares abiertos y cubrir las necesidades de los proveedores e integradores de cloud públicas y privadas, con independencia de su tamaño. La idea es que sea muy escalable y sin excesiva complejidad. Su concepción transparente y abierta (no solo en su código fuente, sino en su gobernanza a través de una Fundación) protege a los clientes del *vendor lockin.* Además, gracias a su ecosistema de empresas, desarrolladores e integradores cooperando y compitiendo entre sí al mismo tiempo, todo va mucho más rápido. No menos importante es su énfasis en la interoperabilidad, que proporciona la libertad para federarse entre proveedores y la capacidad de moverse entre clouds.

Este conjunto de características técnicas y comunitarias, junto a su esquema de licenciamiento permisivo y colaborativo, han convertido a OpenStack en el proyecto de software libre más dinámico de los últimos años y con mayor apoyo por parte de la industria. En función de todo esto, no caben dudas que la instalación de un cloud privado requiere de todas las características ofrecidas por OpenStack, el cual se está convirtiendo en la implementación de referencia, para un cloud en fase de producción.

Las principales ventajas que ofrece OpenStack están relacionadas con su característica de código abierto, tales como: ahorro de costos, mayor independencia frente a soluciones propietarias, mayor modularidad, mayores opciones de personalización, etc. Un aspecto muy importante es la capacidad de versionado y la estabilidad de cada una, esto se debe a que cada seis meses se libera una versión y se deja disponible en el repositorio *Git*, cada una de ellas mejora al menos dos componentes de la versión anterior y agrega al menos un componente nuevo.

El mayor inconveniente que se presentó al instalar la infraestructura fue la configuración de los archivos de sistemas y la necesidad de dependencia de ciertos componentes de software necesarios para comenzar el montado (por ejemplo es necesario clonar los repositorios, instalar las librerías necesarias para el soporte de virtualización, conectividad, etc.). Una vez solucionado estos problemas, la herramienta *DevStack*, facilita el proceso de instalación automatizando las tareas de configuración de cada módulo.

Cuando se ha finalizado la instalación de OpenStack, el manejo del entorno es bastante amigable desde el Dashboard, sin embargo, se pueden manipular los componentes desde línea de comando. Esta tarea si bien es más complicada y requiere de mayor conocimiento de los detalles de configuración, es más potente. Ya sea que se use la interfaz gráfica o la consola, se pueden definir roles, proyectos, levantar imágenes, lanzar instancias, etc. de manera gráfica.

Desde el punto de vista académico, el contar con una infraestructura cloud privada resulta un tema interesante de investigación y evaluación de cuán buena es esta solución para comenzar a migrar los datos, aplicaciones y cómputo a un entorno virtualizado. En particular, instalar OpenStack ha sido una tarea de investigación integral y continua que ha llevado a generar líneas de investigación que permitirán seguir avanzando en el tema.

7.2. FUTURAS AMPLIACIONES

Uno de los mayores problemas que se presentaron con la implementación de la arquitectura monolítica fue el soporte de un único hipervisor (QEMU) y la imposibilidad de contar con una infraestructura de hardware para la producción. Debido a esto es que se está trabajando en la migración a una arquitectura distribuida, donde cada componente de la infraestructura estará instalado en un nodo diferente, lo que permitirá que se pueda seleccionar qué hipervisor instalar. Además, otro aspecto importante a tener en cuenta es la necesidad de migrar a la última versión estable: Kilo, esta versión tiene la ventaja de contar con el componente IRONIC que permite ofrecer virtualización como servicio, esto permitirá que la infraestructura virtualise en forma nativa sobre el hardware, lo cual disminuye los retardos provocados por el proceso de virtualización.

A partir de esta nueva implementación será posible medir el rendimiento de algoritmos sobre diferentes hipervisores con el objeto de determinar cuál es el de mejores resultados en términos de rendimiento de la virtualización, En este sentido. La idea es tomar algoritmos tradicionales y correrlos sobre arquitecturas tradicionales y sobre OpenStack para determinar cuánto se degrada la performance en una infraestructura de cloud virtualizada. En este sentido, toda la experiencia adquirida en la instalación, configuración y puesta en marcha del cloud será directamente aprovechable.

Entre los aspectos más importantes de la implementación de OpenStack Juno, es la ventaja que provee su modularidad, permitiendo instalar los componentes que se necesiten, el otro aspecto es que esta implementación es no apropiativa, debido a que los servicios del cloud se levantan cuando se necesitan y puede convivir con otras aplicaciones. Una vez que se ha montado el cloud, se levanta el servicio por consola y se usa mediante el dashboard.

Otro aspecto importante es la posibilidad de instalación sobre hardware sin demasiados requerimientos de memoria y disco, esto ha permitido poder usar nodos con características

estándares, esto se debe fundamentalmente a que la instalación se realizó con Ubuntu 14.04 como sistema operativo de base, cuyos requerimientos para instalación son 4GB de RAM y 500 Mbyte de disco. Este es un aspecto distintivo sobre otras infraestructuras de cloud, como por ejemplo OpenNebula cuyo sistema operativo de base recomendable es Debian, el cual requiere mayor capacidad de hardware.

BIBLIOGRAFÍA

[1] Armbrust, Fox, Griffith, Joseph, Katz, Konwinski, Lee, Patterson, Rabkin, Stoica, Zaharia. "A View of Cloud Computing" Communications of the ACM, Vol. 53 No. 4, Pages 50-58.

[2] Rodríguez, Chávez, Martin, Murazzo, Valenzuela. "Interoperabilidad en cloud computing". WICC 2011. Rosario Argentina.

[3] Zhu, Fang et al. "IBM Cloud Computing Powering a Smarter Planet", Libro Cloud Computing, Volumen 5931/2009, Páginas 621- 625.

[4] Mell, Grance. "The NIST definition of cloud computing." NIST Special Publication 800 – 145, 2011.

[5] Murazzo, Rodríguez, Villafañe, Gallardo. "Desarrollo de aplicaciones colaborativas para cloud computing". Mar del Plata, Argentina. CACIC 2013.

[6] Rodriguez, Murazzo, Chavez, Guevara. "Arquitectura de cloud computing híbrida basada en tecnología open source". Buenos Aires, Argentina.CACIC 2014.

[7] Murazzo, Tinetti, Rodriguez. "Infraestructura de Cloud Computing". Salta, Argentina. WICC 2015.

[8] Aggrawal, Kumar, Kaushik, Karel, Mathur. "Virtualization: A Concept Implementation for Cloud". International Journal of Engineering and Technical Research (IJETR). 2014.

[9] Manvi, Shyam. "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey". Journal of Network and Computer Applications, 41, 424-440. 2014.

[10] Araujo, Matos, Alves, Maciel, Souza, Trivedi. "Software aging in the eucalyptus cloud computing infrastructure: characterization and rejuvenation". ACM Journal on Emerging Technologies in Computing Systems (JETC). 2014.

[11] Kumar, Gupta. "Open Source Infrastructure for Cloud Computing Platform Using Eucalyptus". Global Journal of Computers & Technology Vol, 1(2). 2014.

[12] Deshpande, Sharma, Peddoju. "Implementation of a private cloud: a case study". Proceedings of the Third International Conference on Soft Computing for Problem Solving (pp. 635-647). Springer India. 2014.

[13] Rahim. "Analysis of design patterns in OpenNebula". *Computer and Information Sciences (ICCOINS), 2014 International Conference on* (pp. 1-6). IEEE.

[14] AL-Mukhtar, Mardan. "Performance Evaluation of the CloudStack Private Cloud". International Journal of Cloud Computing and Services Science (IJ-CLOSER), 403-416. 2014

[15] Kumar, Jain, Maharwal, Jain, Dadhich. "Apache CloudStack: Open Source Infrastructure as a Service Cloud Computing Platform". International Journal of advancement in Engineering technology, Management and Applied Science. 2014.

[16] Kumar, Gupta, Charu, Jain, Jangir. "Open Source Solution for Cloud Computing Platform Using OpenStack". International Journal of Computer Science and Mobile Computing, pp 89-98. 2014.

[17] [17] Singh, Pratap, Pachauri. "Critical Analysis of Cloud Computing Using OpenStack"

IJCSMC, 3(3), pp 121-127. 2014.

[18] Parmar, Champaneria. "Comparative Study of Open Nebula, Eucalyptus, Open Stack and Cloud Stack". International Journal of Advanced Research in Computer Science and Software Engineering, 4(2). 2014.

[19] Nagar, Suman. “Architectural Comparison and Implementation of Cloud Tools and Technologies”. 4 th IEEE International Conference on Electronics Computer Technology (ICECT'12), Kanyakumary, India, IEEE Explore. 2014.

[20] Lixandroiu, Maican. “A Model for Comparing Free Cloud Platforms”. Informatica Economica, 18 (4). 2014.

[21] Barkat, dos Santos, ikken. “Open Source Solutions for Building IaaS Clouds”. Scalable Computing: Practice and Experience, 16(2). 2015.

[22] OpenStack. “OpenStack: The Open Source Cloud Operating System”. URL: http://www.openstack.org/software.

[23] Rosado, Bernardino. “An overview of openstack architecture”. Proceedings of the 18th International Database Engineering & Applications Symposium (pp. 366-367). ACM. 2014.