ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRONICA, REDES Y COMUNICACIÓN DE DATOS

TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA

AUTOR:

HERNAN ALFREDO HINOJOSA RIZZO

MILTON PATRICIO ULLOA IBARRA

TEMA: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE NUBE COMPUTACIONAL BASADOS EN LA PLATAFORMA DE CÓDIGO ABIERTO OPENSTACK

DIRECTOR: ING. ROMERO, CARLOS

CODIRECTOR: ING. SAENZ, FABIAN

SANGOLQUÍ, FEBRERO 2014

CERTIFICACIÓN

Certificamos que el presente proyecto de grado titulado: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE NUBE COMPUTACIONAL BASADOS EN LA PLATAFORMA DE CÓDIGO ABIERTO OPENSTACK, ha sido desarrollado en su totalidad por los señores HERNAN ALFREDO HINOJOSA RIZZO y MILTON PATRICIO ULLOA IBARRA, bajo nuestra dirección.

Atentamente

Ing. Carlos Romero G.	Ing. Fabián Sáenz E.
DIRECTOR	**CODIRECTOR**

DECLARACIÓN DE RESPONSABILIDAD

HERNAN ALFREDO HINOJOSA RIZZO

MILTON PATRICIO ULLOA IBARRA

DECLARAMOS QUE:

El proyecto denominado “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE NUBE COMPUTACIONAL BASADOS EN LA PLATAFORMA DE CÓDIGO ABIERTO OPENSTACK”, ha sido desarrollado en base a una investigación exhaustiva, respetando los derechos intelectuales de terceros, conforme a las fuentes que se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 21 de Febrero de 2014.

Hernán Alfredo Hinojosa Rizzo	Milton Patricio Ulloa Ibarra

AUTORIZACIÓN

HERNAN ALFREDO HINOJOSA RIZZO

MILTON PATRICIO ULLOA IBARRA

Autorizamos a la Universidad de las Fuerzas Armadas “ESPE” la publicación, en la biblioteca virtual de la Institución del trabajo “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE NUBE COMPUTACIONAL BASADOS EN LA PLATAFORMA DE CÓDIGO ABIERTO OPENSTACK”, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Sangolquí, 21 de Febrero de 2014.

Hernán Alfredo Hinojosa Rizzo	Milton Patricio Ulloa Ibarra

DEDICATORIA

Dedicamos el presente trabajo a quienes nos acompañaron durante nuestra vida universitaria, en especial a nuestras Familias que nos brindaron su apoyo y soporte incondicional.

Hernán y Patricio

AGRADECIMIENTOS

Nuestro agradecimiento al Cuerpo de Docentes de la ESPE por los conocimientos compartidos. Un aprecio y gratitud especial al Ing. Carlos Romero y al Ing. Fabián Sáenz por el empuje y la ayuda brindada para que podamos finalizar nuestra carrera.

Otro inmenso a nuestras Familias por la motivación y el tiempo que nos dispensaron para dedicarnos a culminar el presente proyecto.

PRÓLOGO

La implementación de una nube computacional ofrece características especiales tales como alta estabilidad y uso bajo demanda, ambas propiedades son ampliamente requeridas por los usuarios con equipos cada vez más poderosos, así como también por los proveedores a quienes las demandas del mercado les exigen cada vez más servicios y a menor precio.

Openstack se presenta como la mejor opción para crear nubes computacionales, cuenta con el soporte de una amplia Comunidad así como también de Empresas como Rackspace quienes brinda apoyo para la implementación de la solución en Nubes públicas, privadas o híbridas. No obstante que la instalación de la plataforma requiere de un cuidado extremo en los detalles ya que al ser de amplio espectro, cuenta con muchas opciones al momento de realizar una implementación personalizada.

En el **Capítulo 1** se detalla el concepto de Nube Computacional en general para luego, en el **Capítulo 2** extender más a detalle el alcance teórico de la aplicación Openstack exclusivamente. El **Capítulo 3** describe los modelos de implementación existentes y llevar a la práctica el proceso de instalación en el **Capítulo 4.**

El **Capítulo 5** detalla las prácticas realizadas en la plataforma y finalmente, en el **Capítulo 6** se presentan las conclusiones y recomendaciones obtenidas de todo el proceso de análisis, presentando de los resultados de la implementación.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El concepto de Nube Computacional ha tomado gran importancia en los últimos tiempos por las ventajas operacionales que brinda a los usuarios. Aplicaciones ya se encuentran disponibles en forma de infraestructura rentada, denominada como IaaS, o directamente programas y aplicaciones que pueden ser utilizadas de forma remota y bajo demanda; SaaS, sin necesidad de haberlas instalado y configurado localmente librando de esta manera a los usuarios de la responsabilidad de implementar, mantener y actualizar una plataforma computacional. Existen algunas opciones en el mercado para este tipo de implementación, el presente documento describe una de ellas; el proyecto Openstack, desarrollado inicialmente como una colaboración de NASA® y Rackspace® para sus propios Centros de Cómputo, pero que ahora se encuentra disponible bajo licencia Apache 2.0 con el aporte permanente de Comunidad de Desarrolladores y de Fabricantes de equipos informáticos que han adherido Openstack a sus soluciones comerciales para convertirla en un estándar referencial de este tipo de implementaciones. Openstack incluye entre otros, al proyecto Neutron considerado como precursor o base para implementaciones de SDN.

Palabras Clave:

- Nube Computacional
- Openstack
- SDN
- IaaS

ABSTRACT

The concept of Cloud Computer has become very important in recent times, either because their implementation leverages the computing infrastructure that contains or provides operational benefits to users. Computational Cloud Applications are now available as virtual infrastructure referred also as IaaS or directly leased software and applications that can be used remotely and on-demand basis without request any installation and configuration procedures, known also as SaaS. Thus ridding users the responsibility to implement, maintain and continuously update a computer platform. There are many options on the market for the implementation of a Computer Cloud , this document describes one of them, the Openstack project, initially developed as a collaboration between NASA® and Rackspace® for their own datacenter, but now is available under Apache 2.0 license with free distribution and with the continuing collaboration of a global Community of Developers and IT Manufacturers that have include Openstack in their business solutions turning it as an standard reference for Cloud Computing deployments. Openstack includes among many projects, the project Neutron considered as a precursor or base for SDN implementations.

Key Words:

- Cloud Computing
- Openstack
- SDN
- Iaas

GLOSARIO

HIPERVISOR: (En inglés hypervisor) o monitor de máquina virtual (virtual machine monitor) es una plataforma que permite aplicar diversas técnicas de control de virtualización para utilizar, al mismo tiempo, diferentes sistemas operativos (sin modificar o modificados en el caso de paravirtualización) en una misma computadora. Es una extensión de un término anterior, «supervisor», que se aplicaba a los kernels de los sistemas operativos.

VIRTUALIZACIÓN: Es la creación -a través de software- de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red..." "..Esta capa de software (VMM) maneja, gestiona y arbitra los cuatro recursos principales de una computadora (CPU, Memoria, Almacenamiento y Conexiones de Red) y así podrá repartir dinámicamente dichos recursos entre todas las máquinas virtuales definidas en el computador central. Esto hace que se puedan tener varios ordenadores virtuales ejecutándose en el mismo ordenador físico.

TOKENS: También llamado **componente léxico** es una cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación (Wikipedia, http://es.wikipedia.org/wiki/Token_(inform%C3%A1tica), 2013)

FLAVORS: Receta que se define a los distintas opciones de máquinas virtuales disponibles en Openstack

Mod_wsgi: Módulo Web Server Gateway Interface: Especificación que describe la comunicación entre un servidor de aplicaciones web, en nuestro caso Apache, y la aplicación, en este caso Dashboard. (WSGI.org, 2013)

VLAN: Virtual Local Address Network.

Plugin: Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica (Wikipedia, Wikipedia, 2013)

FreeBSD: es un sistema operativo libre para computadoras basadas en lasCPU de arquitectura Intel, basadas en la versión 4.4 BSD-Lite del Computer Systems Research Group (CSRG).

Illumos: es un proyecto de software libre derivado de OpenSolaris. Fue anunciado por conferencia web desde Nueva York el 3 de agosto de 2010.

RPC - Remote Procedure Call : es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. (Wikipedia, 2013)

HAProxy: Acrónimo de High Availability Proxy o proxy de alta disponibilidad **Openstack Foundation**: Grupo de Empresas que se han unido para, de manera conjunta, determinar los lineamientos del desarrollo de Openstack. Más información en https://www.openstack.org/foundation/

Openstack Community -: Grupo de desarrolladores que aportan conocimiento al código de Openstack y siguiendo los direccionamientos emitidos desde Openstack Foundation. Más información en https://www.openstack.org/community/

ROI: Acrónimo del término financiero Return of Investment – Retorno de la Inversión.

1. CAPÍTULO I: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE NUBE COMPUTACIONAL BASADOS EN LA PLATAFORMA DE CÓDIGO ABIERTO OPENSTACK

1.1. Antecedentes

La nube computacional o Cloud Computing, consiste en la aplicación simultánea de varios conceptos relacionados con las tecnologías de la información, como son la virtualización, el diseño de aplicaciones distribuidas y el diseño de redes, permitiendo el acceso a los recursos computacionales (equipos, almacenamiento, sistemas operativos, aplicaciones, entornos de desarrollo) mediante servicios web, mientras es transparente para el usuario final la ubicación e implementación física de los equipos que alojan estos recursos y servicios, lo que lo hace de cierta manera abstracto o en como si estuviera en una “nube”.

Tratando de lograr una definición sencilla pero que a la vez demuestre toda la fortaleza del proyecto Openstack, podemos denominarla de la siguiente manera: Openstack es un conjunto de herramientas basadas en software que permite la implementación de una nube computacional en un entorno de acceso privado o público utilizando la infraestructura de modo más eficiente.

Siendo este un proyecto de titulación cuyo objetivo principal es el demostrar de forma práctica el funcionamiento en general de Openstack, debemos considerar todo el marco teórico que conlleva la operación de los diferentes componentes de la aplicación en sí misma. Por lo tanto vamos a

analizar brevemente los diferentes conceptos proyectados en la definición que hicimos y más adelante procederemos con un análisis más a detalle de los mismos.

En la definición se usaron varios términos importantes; nube computacional, de inicio conocida con el prefijo de *“paradigma” de nube computacional*. Buscando la definición más apropiada a nuestro tema sobre el significado de paradigma encontramos la siguiente:

Conjunto cuyos elementos pueden aparecer alternativamente en algún contexto especificado (RAE, 2013)

Estos elementos que conforman una nube computacional, para nuestro análisis, son los diferentes tipos de servicio que puede brindar. Desde el punto de vista del usuario puede significar alta disponibilidad de recursos, tanto en rapidez de implementación como en capacidad de servicio. Ahora desde el punto de vista del proveedor significa mayor aprovechamiento de la infraestructura y un bajo costo de implementación.

Apuntando hacia el tema específico de la relación entre nube computacional y Openstack, se puede decir que Openstack es el administrador de la nube o quién orquesta todos los servicios que podemos encontrar en la nube computacional. Esta administración incluye a los usuarios y sus aplicaciones, analizando más detenidamente, lo hace a través de tres componentes básicos que definen el conjunto de herramientas enunciadas en la definición inicial;

FIGURA 1 : Componentes de Openstack

El componente computacional o Compute que a su vez tiene varios sub componentes cuyo objetivo es el de encargarse del control y operación de los diferentes servicios ofertados al usuario.

El componente de Almacenamiento o Storage también con varias aplicaciones internas que es donde se guardan y almacenan tanto la información de la aplicación así como la información generada por el usuario.

Y finalmente el componente de Networking o Red donde se realizan las tareas de comunicación de las aplicaciones hacia el exterior de la nube, como por ejemplo hacia el Internet o con un equipo en red local, por ejemplo hacia un servidor de Active Directory.

Todos los componentes principales y los subcomponentes se comunican entre ellos a través de una cola de mensajería única, permitiendo que Openstack sea escalable y modular pero a la vez permite que se relacione fácilmente con otras nubes computaciones ya sean públicas o privadas.

1.2. Justificación e Importancia

Debido a la creciente popularidad que ha tomado el concepto de nube computacional entre los usuarios y en especial en el segmento de

administradores IT, su aplicación ha sido mejorada mediante el uso de interfaces Web y APIs de fácil integración.

Openstack surge como resultado de esta creciente demanda de información y aplicación del tema, sobre todo considerando quienes están detrás del proyecto, se puede intuir que muy pronto se convertirá en una herramienta estándar para quienes se encuentran relacionados con el área de infraestructura y hardware, ya sean como usuarios administradores de la nube o como parte de su desarrollo hacia nuevos niveles de servicio.

Así mismo hay quienes han opinado en contra de la aplicación de las nubes computacionales basándose en que se trata de la vieja aplicación cliente-servidor que fue superada cuando los equipos personales alcanzaron niveles altos de desempeño y los programas se volvieron más amigables con los usuarios.

Justamente la alta disponibilidad computacional ociosa afianza la fortaleza de las nubes computacionales, ya que al momento la mayoría de equipos en los grandes centros de cómputo tiene su capacidad de procesamiento y almacenamiento subutilizados. Toda la capacidad desocupada puede ser aprovechada como herramienta para otros usuarios que no cuentan con el dinero suficiente para instalar sus propios centros de cómputo y que ven a la nube computacional como una oportunidad de acceder a ellos mediante una renta de bajo presupuesto.

1.3. Objetivos

1.3.1. Objetivo general

- Instalación de un sistema de nube computacional mediante la plataforma en código abierto Openstack.

1.3.2. Objetivos específicos

- Análisis de la teoría de Nube Computacional.
- Descripción de los diferentes tipos de implementaciones disponibles tanto en código abierto como bajo marcas comerciales.
- Análisis de los componentes de Openstack
- Diseño de la infraestructura necesaria para la implementación
- Instalación y puesta en marcha de la aplicación.
- Dejar instalado y en funcionamiento la plataforma de Nube computacional para futuros análisis y estudios por parte de los estudiantes del DEEE-ESPE.

1.4. Marco teórico

1.4.1. Conceptos y definiciones de una nube computacional

El término “Nube Computacional”, “Computación en la Nube” o “Cloud Computing” se ha vuelto más popular en los últimos tiempos, inicialmente solo se lo trataba en medios técnicos pero ahora se ha convertido en una herramienta de promoción de servicios hasta captar el interés de quienes, de una forma u otra, mantienen una relación con la tecnología de los Sistemas de Información y el público en general.

La nube computacional o Cloud Computing se compone de dos términos; Nube o Cloud, que es el símbolo que se usa generalmente para representar de cierta manera a un sistema abstracto, como por ejemplo al Internet; y Computacional o Computing, que hace referencia a los términos relacionados con la informática, programas y almacenamiento de datos. Por tal motivo muchos autores denominan a la nube computacional como un paradigma que permite ofrecer servicios de computación a través de una red de datos tipo LAN, WAN o a través del Internet.

Aquí se detallan algunas definiciones, según IBM:

"Es una solución todo-incluido en la cual todos los recursos computacionales (hardware, software, networking, almacenamiento, etc) son provistos a los usuarios de manera rápida según las exigencias de sus necesidades" (IBM, Cloud computing for the enterprise: Part 1: Capturing the cloud, s.f.)

El Laboratorio de Tecnologías de la Información, integrado en el *National Institute of Standards and Technology* (NIST) del Departamento de Comercio del Gobierno Federal de los Estados Unidos, ha definido *Cloud Computing* de la siguiente forma:

"Cloud Computing es un modelo que permite el acceso bajo demanda y a través de la red a un conjunto de recursos compartidos y configurables (como redes, servidores, capacidad de almacenamiento, aplicaciones y servicios)

que pueden ser rápidamente asignados y liberados con una mínima gestión por parte del proveedor del servicio" (Peter Mell, 2011)

La nube en sí misma facilita la entrada, procesamiento y salida de la información como un servicio. Los servicios de la nube incluyen el software, infraestructura y almacenamiento en la red basadas en la demanda del usuario.

Analizando más a detalle el aspecto operativo de la nube, encontramos las siguientes definiciones:

"Todo lo que nos ofrece la Nube Computacional son propiedad o dependen de una infraestructura de software", conocida como middleware, "que es invisible para la comunidad de investigación" (Nurmi, 2009)

Lo que introduce el concepto de middleware a nuestro proyecto, siendo un middleware definido de la siguiente forma:

"Un middleware es un tipo de software que permite manejar la complejidad y la heterogeneidad de los recursos. Es una capa de software que está encima del Sistema Operativo, pero debajo de la capa programas o aplicaciones, que provee una abstracción en los sistemas distribuidos" (Bakken, 2009)

Dada la importancia que ha tomado el concepto de Nube Computacional y las oportunidades de negocio que se han generado a su alrededor, todas o casi todas las grandes empresas IT tanto de hardware como de software han demostrado su interés a través de estrategias a largo plazo para la implementación de nubes Computacionales bajo su marca: IBM, Intel,

Microsoft, Oracle, Hewlett-Packard, Cisco Webex, Juniper, etc. Entre los usuarios que han aprovechado de las bondades de la nube están con fines comerciales están Rackspace, Cloudera y Amazon AWS. Las nubes computacionales han ayudado a que el correo electrónico pueda ser leído y archivado a distancia en Google Mail, Yahoo Mail, Microsoft Mail. También es posible subir, ver y descargar videos YouTube; o escuchar cualquier tipo de música en línea. Otro ejemplo de la aplicación de nube computacional es el software de gestión de clientes CRM (Customer Relation Management) en línea de Salesforce.com

En Latinoamérica la empresa MercadoLibre.com ha migrado todos sus servidores hacia su propia nube computacional disminuyendo sus costes de operación. (Comisario, 2013)

Así mismo y poco a poco se puede encontrar instalaciones de nubes computacionales en grandes empresas, universidades, entidades gubernamentales, que requieren sus propios centros de datos para ponerlos a disposición de sus empleados, investigadores y estudiantes con el objetivo de disminuir los tiempos de implementación de servicios.

1.4.2. Características:

Veamos las características más importantes asociadas a Cloud Computing:

Según el NIST, el modelo de la nube computacional tiene cinco características esenciales, así como también tres modelos de servicio y cuatro modelos de implementación. (Peter Mell, 2011)

Servicios a la carta.

El usuario puede acceder a capacidades de computación en la nube de forma automática a medida que las vaya necesitando sin necesidad de una interacción humana con su proveedor.

Esta característica incluye el tema tanto de logística de la implementación de los recursos así como el pago que corresponde su utilización.

Múltiples formas de acceder a la red.

Los recursos son accesibles a través de la red y por medio de mecanismos estándar que permiten su utilización por cualquier tipo de equipos heterogéneos por ejemplo desde los computadores de escritorio hasta teléfonos móviles o tabletas.

Esta característica denominada ubicuidad permite que múltiples sistemas puedan acceder a un mismo servicio desde cualquier ubicación física (siempre que cuenten con acceso a la dicha red), y es una de las principales bondades que aporta la Nube Computacional.

Con el objetivo de proporcionar dicha ubicuidad, los proveedores de la “nube” cuentan con la infraestructura física o virtual y el ancho de banda necesarios para dar cabida a los requisitos de los diferentes equipos que se subscriben. Además, los proveedores disponen de rutas redundantes y balanceadas en las redes de comunicaciones de acceso a sus servicios, lo cual ofrece mayor garantía en el equilibrio de la carga de datos por esas rutas,

reduciendo la posibilidad de que las redes se sobrecarguen y que los servicios ofrecidos se retrasen, fallen o se corten.

Compartición de recursos.

Los recursos computacionales tales como almacenamiento, memoria, ancho de banda, capacidad de procesamiento, máquinas virtuales, etc. de los proveedores son compartidos bajo el modelo multi-usuarios, a los que se van asignando capacidades tanto físicas como virtuales de manera dinámica según sus requerimientos. Los usuarios pueden desconocer la ubicación de los recursos a los que acceden y en algunos casos solo podrán seleccionar algunos aspectos tales como el Centro de cómputo donde se aloja la infraestructura.

Siendo la virtualización el método por el cual es posible una optimización respecto al aprovechamiento de los recursos instalados ya que permite que **las aplicaciones sean independientes del hardware** en el que se alojan, es decir que varias aplicaciones pueden ejecutarse en una misma máquina o una aplicación puede usar varios equipos al mismo tiempo, dependiendo de la necesidad del usuario.

Consideraciones referentes a la seguridad de la información de cada usuario son necesarias al momento de implementar entornos virtuales en los que los recursos de almacenamiento son compartidos, por lo que es necesario establecer controles adecuados de acceso y gestión segura de la información en cada uno de los niveles informáticos mediante el cifrado de los datos.

Rapidez y Elasticidad.

Por elasticidad de la Nube computacional se refiere a la capacidad de asignar y liberar los recursos computacionales físicos o virtuales en algunos casos hasta de manera automática y que su crecimiento y decrecimiento sea asociado con la demanda. Para el usuario, los recursos serán considerados como ilimitados y podrán ser asignados en cualquier cantidad que se encuentran siempre disponibles.

Servicio ponderado.

El controlador de la Nube Computacional tiene la capacidad y las herramientas para medir, a determinado nivel, el servicio y que sea efectivamente entregado a cada usuario, de menara que para el Proveedor y para el Usuario sea clara y conocida la información relacionada al consumo real de los recursos, lo que posibilita el pago por el uso efectivo de los servicios.

1.4.3. Términos económicos

En términos económicos, se dice que ambas características, virtualización y escalabilidad aumentan la elasticidad y la ponderación de la Nube computacional, ya que los costes asociados al uso de los recursos se adaptan mejor a las necesidades y a los requerimientos del momento. Las principales formas de pago asociadas a la Nube son:

1.4.3.1. Pago por una suscripción al servicio.

En esta forma de pago se concierta un precio predefinido durante un periodo de tiempo en el que se hace uso de determinados recursos contratados. La suscripción se puede realizar sobre diferentes parámetros:

- **Por número de usuarios:** se establece el pago en función del número de usuarios que acceden del servicio durante un cierto periodo de tiempo.
- **Por tipo de funcionalidad:** el pago se realiza en función del número de funcionalidades, ofrecidas como servicio, consumidas durante un periodo de tiempo prestablecido.
- **Por consumo ilimitado con tarifa fija:** permite un uso ilimitado de ciertos recursos durante un periodo de tiempo y a cambio de un valor establecido.

1.4.3.2. Pago por uso

La forma de pago en este caso se basa en los recursos utilizados, por ejemplo, en función de la cantidad de información transmitida por las redes de datos, o el pago por cada gigabyte de información almacenada, o por recurso de procesamiento o en memoria RAM. **(IBM, 2009)**

1.4.4. Aplicaciones y beneficios

La nube computacional presenta grandes oportunidades para los usuarios en general, sean particulares o pertenecientes a una empresa. Para determinar el alcance de dichos beneficios es necesario clasificarlos de acuerdo las ventajas competitivas que ofrece:

1.4.4.1. Ventajas estratégicas

Las ventajas que la Nube Computacional ofrece a los usuarios, en este caso los alumnos del Departamento de Eléctrica y Electrónica de la ESPE, desde un punto de vista estratégico se pueden agrupar en las siguientes:

- **Fiabilidad**

La Escuela Politécnica del Ejército puede aprovechar las características de escalabilidad, ubicuidad y virtualización relacionadas a la Nube Computacional para disponer de sistemas replicados que reducen la posibilidad de pérdida de información o del servicio prestado en caso de un desastre o sabotaje, ofreciendo a su vez una mayor disponibilidad gracias a un servicio más balanceado en el caso de que el uso de los sistemas sea mucho mayor del previsto por parte de los usuarios.

- **Productividad**

La nube Computacional permite el acceso a sus servicios desde cualquier ubicación física. De tal manera que los usuarios, alumnos e investigadores pueden acceder a las aplicaciones, documentos y correos electrónicos almacenados en la nube desde cualquier lugar con acceso a su Red mediante un enlace VPN a través del Internet, trabajando con ellos *en línea* o desconectados y con posibilidad de sincronizarlos posteriormente como ya los hacen algunos proveedores como SugarSync.com. Esto aumenta la flexibilidad y la capacidad de trabajar a distancia o en grupo, y como resultado la mejora en la productividad reflejada en el avance de los proyectos en desarrollo.

- **Contribuciones múltiples**

El uso de programas y aplicaciones a través de la red de datos permite la interacción entre profesores y alumnos para que puedan trabajar a la vez en un mismo documento en tiempo real, por ejemplo la empresa Woodwings admite la elaboración y edición de libros permitiendo tanto a los autores, diseñadores y editores ver el mismo documento en tiempo real, con ello se fomenta la productividad y la comunicación entre los diferentes estamentos de trabajo.

- **Nuevas Oportunidades**

La reducción de infraestructura computacional que resulta de migrarse a la “nube” permitiría a la ESPE crear productos que anteriormente no eran posibles o que representaban una alta inversión que en su momento no era posible realizar. La ventaja de utilizar a la nube computacional como plataforma de operación está en el hecho de que ahora se pueden llevar a cabo muchas ideas de proyectos que anteriormente exigían grandes niveles de potencia de procesamiento, una capacidad de ampliación rápida o una orientación de trabajo distinta.

- **Tercerización**

El permanente y acelerado cambio en la infraestructura tecnológica implica que cualquier recambio sobre los servicios ya implementados sea demasiado costoso. Para evitar esto, la Nube Computacional permite a los usuarios realizar alianzas con proveedores de servicios en la nube probablemente con mayor capacidad de adaptación a las innovaciones y

recambios, como por ejemplo los servicios de Amazon AWS o Rackspace, permitiéndoles adaptarse rápidamente a cualquier nueva necesidad y contar con opciones más novedosas y atractivas tanto en infraestructura así como en aplicaciones.

- **Inversiones inteligentes**

La migración hacia la nube computacional de las actividades operacionales de una empresa, permite que se pueda dedicar de manera más efectiva a centrar sus esfuerzos en las tareas propias de su negocio, sin distraer los presupuestos en proyectos tecnológicos cuyo tiempo de implementación significa una alta demanda de recursos económicos sin tener la certeza de las retribuciones. Se puede aprovechar esta ventaja competitiva para ofrecer una mayor inversión en innovación y desarrollo investigativo, haciendo uso de otros recursos que pueda ofrecer la nube computacional sin necesidad de implementar físicamente los equipos.

1.4.4.2. Ventajas técnicas

Analizando las bondades técnicas de la nube computacional en comparación con los sistemas tradicionales y tratando de enfocarlos hacia su aplicación en la ESPE, podemos determinar los siguientes:

- **Establecimiento de nuevos servicios**

Al momento de implementar un nuevo servicio, los proyectos tecnológicos computacionales tradicionales que pueden requerir varias semanas o meses para adquirir, configurar y poner en marcha los recursos asociados a nuevos servicios, el uso de la Nube Computacional permite adoptar en mucho menos

tiempo la infraestructura necesaria para proveer del nuevo servicio. Esto tiene un impacto fundamental en la agilidad del Departamento para responder a los crecientes y cambiantes requerimientos de los miembros, y permite reducir los problemas asociados con los retrasos por parte del proveedor ya que la mayoría de equipos son importados.

- **Flexibilidad**

Permite la asignación democrática de los recursos de la ESPE bajo el estilo de pago por consumo o uso, lo que permite el acceso a todos los alumnos o investigadores de manera equivalente a la infraestructura o servicios con los que cuenta el departamento, con mayor rapidez que los servicios de subcontratación tradicionales relacionados a la adquisición de equipos.

- **Capacidad de recuperación ante fallos**

Los proveedores de la “nube”, en este caso el Departamento, ofrecen tanto soporte frente a problemas en cualquier momento del año a través de la implementación de sistemas redundantes para asegurar una mayor disponibilidad de la información que se administra.

1.4.4.3. Ventajas económicas

Existen importantes ventajas económicas que es necesario tener presente referentes a la nube Computacional; la adopción de este modelo reducirá drásticamente los gastos asociados a la compra de nuevos sistemas informáticos o licencias de aplicaciones informáticas instaladas, el mantenimiento de esos sistemas y los gastos de horas/hombre encargado de

su administración. Todo esto puede suponer por tanto un gran ahorro económico y un impacto muy positivo en el presupuesto de la Universidad.

1.4.5. Análisis de la arquitectura de la Nube Computacional

1.4.5.1. Modelos de implementación de la Nube.

El Instituto NIST clasifica los modelos de implementación de la Nube Computacional en:

- *Nubes privadas*
- *Nubes públicas*
- *Nubes comunitarias*
- *Nubes híbridas*

(Peter Mell, 2011)

- **Nubes Privadas**

En las nubes privadas, la infraestructura de la nube está provista para uso exclusivo de la organización y sus miembros. Puede estar gestionada por una única organización, ya sea directamente o a través de terceros; y puede estar implementada en la infraestructura de la organización o fuera de ella.

En una nube privada, la organización establece un entorno de *virtualización* en sus propios servidores para cada usuario, en cualquiera de sus propios centros de datos o en los de un proveedor de servicios.

La estructura de nube privada es útil para empresas que o bien tienen inversiones o costes significativos de sus sistemas computacionales o

consideran que deben tener un control total sobre los diferentes aspectos de infraestructura.

La ventaja principal de las nubes privadas es el control sobre su infraestructura y se aprovechan todas las ventajas de la virtualización y de la nube computacional en sus propios Centros de Datos, entre las ventajas más notables están las de las interfaces de administración y la medición del uso de los recursos. (Buyya Rajkumar, 2011)

- **Nubes públicas**

La denominación de Cloud Público o Nube Pública se refiere al conjunto de servicios e infraestructura abiertos al público en general y que puede ser administrada por una Empresa, el gobierno, una Universidad o la combinación de ellos. En este caso la nube se encuentra alojada en las instalaciones del prestador de servicios que pone a disposición de cualquier usuario en Internet su infraestructura de forma gratuita o mediante el pago de cierta cantidad relacionada con la magnitud o tiempo de uso de los servicios. (Buyya Rajkumar, 2011)

El uso de nubes públicas permite ampliar fácilmente los recursos necesitados, ya que éstas suelen tener más tamaño que las nubes privadas normalmente implantadas en una única locación.

Como aspectos necesarios a considerar en el caso de este tipo de implementación son:

• La ubicación de los datos de los usuarios puede ser incierta inclusive para el proveedor ya que la nube puede llegar a ser tan automatizada a tal punto de distribuir la información de acuerdo a variables retroalimentadas como por ejemplo el estado físico de los discos de almacenamiento.

• No hay exclusividad en los espacios de almacenamiento por lo tanto la información es guardada en discos compartidos por varias organizaciones a la vez, en este caso se debe solicitar al Proveedor acuerdos en los que se detalla:

o Un acuerdo de Nivel de Servicio (SLA) que describa las garantías sobre posibles pérdidas o falta de disponibilidad de información.

o Control de la propiedad intelectual de la información a través de la implementación estándares de seguridad tales como códigos de cifrado y encriptación y consideraciones relacionadas a la ubicación geográfica de las mismas.

o Condiciones para que el usuario pueda auditar o inspeccionar su información en cualquier momento.

Algunos ejemplos de nubes públicas son Amazon Elastic Compute Cloud, Rackspace, IBM Blue Cloud, Sun Cloud, Google AppEngine y Microsoft Windows Azure Services Platform.

- **Nubes Compartidas**

Se refiere a la infraestructura de nube suministrada para el uso exclusivo de una comunidad en específico, que puede estar conformada por varias

organizaciones que comparten los mismos intereses: Puede estar administradas por todos o algún miembro de la comunidad o por terceros que a su vez proveen la infraestructura para el uso del resto de miembros.

En este tipo de nubes, los temas de interés de los miembros pueden estar relacionados con los objetivos tecnológicos, políticas y necesidades relativas a la seguridad informática o consideraciones sobre el cumplimiento de metas comunes.

- **Nubes Híbridas**

El término se aplica cuando una nube privada ha sido complementada con recursos de una nube pública que se mantienen como entidades separadas pero que a su vez se encuentran unidas por la tecnología estandarizada, como por ejemplo el uso de APIs de código abierto o que a su vez forme parte de algún tipo de desarrollo propietario de algún proveedor en específico, cuyo objetivo es el de proporcionar portabilidad de datos y aplicaciones entre ellas.

Un caso muy frecuente de este tipo de nubes se produce al momento de alquilar temporalmente la capacidad de para manejar picos de carga y se denomina como “cloud bursting”. (Buyya Rajkumar, 2011)

En la actualidad, las nubes públicas y privadas se pueden considerar como subconjuntos del Internet basados en la representación abreviada de la compleja red de conexiones y dispositivos interconectados que forman la red de datos y sus servicios en función de sus relaciones entre sí y con las organizaciones.

Los conceptos público y privado de la informática en nube deben facilitar las interconexiones entre los proveedores y los clientes cumpliendo el estándar general en el nivel de servicio mediante acuerdos que garanticen el desempeño y la estabilidad en general. Como lo mencionamos anteriormente, un ejemplo es el uso de software *Open Source* (código abierto) o software libre, y los estándares abiertos; este es el caso de Xen en el entorno AWS de Amazon y Openstack de Rackspace que permiten la creación de nubes privadas de bajo costo pero cumpliendo los estándares de la industria y que su vez pueden ser parte de una nube privada y pública.

1.4.6. Arquitectura de la capa de servicios

Para comprender las bondades y capacidades que la nube computacional puede ofrecer es necesario dividirla en las siguientes capas o layers:

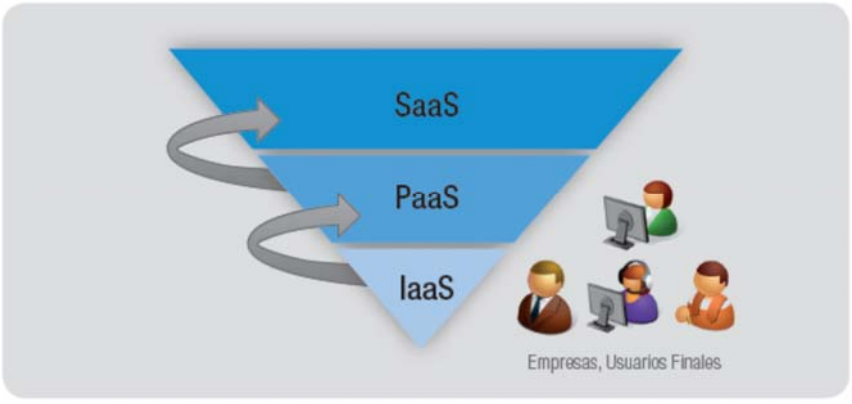


FIGURA 2: Capa de Servicios

1.4.6.1. Software como Servicio (SaaS)

Corresponde a la capa más alta y consiste en la entrega de una aplicación completa como un servicio. En este caso el proveedor de SaaS dispone de una aplicación estándar desarrollada, operada y mantenida en algunos casos por él mismo y con la que brinda servicio a una masa de usuarios a través de la red, sin que estos tengan que instalar ningún programa adicional en sus equipos.

Los proveedores de SaaS son responsables de la disponibilidad y funcionalidad de sus servicios sin omitir las necesidades de los usuarios que finalmente son los que se beneficiarán del software.

Las actividades son gestionadas desde alguna ubicación central, en lugar de hacerlo desde cada lugar o equipo, permitiendo a los usuarios el acceso remoto a las aplicaciones mediante la Red. Igualmente, las actualizaciones son centralizadas, eliminando la necesidad de descargar parches por parte de los usuarios finales para sus equipos.

Se pueden clasificar de acuerdo al tipo de servicio que ofrecen:

TABLA 1 : Tipos de Servicios Cloud Computing

Tipo de Servicio	Ejemplo de Proveedor
Aplicaciones como sitios Web	Box.net Facebook LinkedIn Twitter MySpace – Micosoft Google Maps
Colaboración y aplicaciones de Oficina	Cisco WebEx Weboffice Google Docs Google Talk Jungle Disc Gmail Microsoft Hotmail Yahoo Mail
Servicios de Pago	Amazon FPS Paypal
Software basado en Web pero Integrables a otras aplicaciones	Evernote Google Calendar API Salesforce.com Yahoo! Maps API Amazon AWS

Ventajas

Ventajas comparativas con respecto a las aplicaciones tradicionales son las siguientes:

- *Inversión inicial:* Las implementaciones tradicionales tienen un alto costo relacionado con la adquisición de licencias, en el caso de SaaS el pago se realiza por el tiempo de uso de la aplicación.
- *Multiplataforma:* El acceso a las aplicaciones en SaaS se realiza mediante el navegador de Internet por lo tanto es independiente del sistema operativo o de la plataforma utilizada por el usuario.
- *Actualizaciones:* En este caso el proveedor de SaaS se encargará de gestionar los aspectos referentes a las mejoras en las aplicaciones, ofreciendo de tal manera al usuario acceso a las últimas versiones de los programas.
- *Logística:* Generalmente las aplicaciones tradicionales implican que exista un departamento de Sistemas o IT por detrás que gestione y opere la plataforma en la que se ejecutan, con SaaS se disminuye la gestión de las aplicaciones así como también el tiempo que se destina al soporte y mantenimiento, permitiendo al departamento de IT direccionar mayor tiempo y recursos hacia el giro de negocio de la empresa.

Consideraciones

Al momento de elegir un proveedor de SaaS se recomienda considerar los siguientes aspectos:

Técnicos: Los enlaces de datos pueden resultar en un cuello de botella al momento de utilizar el SaaS, hay que realizar los cálculos previos para verificar que el enlace corporativo permitirá un uso adecuado. Así mismo otro tema a considerar es la personalización del SaaS, en algunos casos las organizaciones requieren de un programa específico para su necesidad, el

SaaS personalizado puede resultar más costoso que una aplicación alojada localmente.

Económicos: Muy probablemente la inversión inicial al implementar el SaaS será menor que la inversión en una plataforma tradicional, sin embargo hay que analizar a largo plazo ya que SaaS necesariamente requiere pagos mensuales por el uso.

Legales: La sensibilidad de la información es un factor a considerar al momento de elegir el proveedor de SaaS, los acuerdos de servicio antes mencionados deben especificar aspectos relacionados a la seguridad de los datos de los usuarios.

1.4.6.2. Plataforma como Servicio (PaaS)

En orden descendente, PaaS (Platform as a Service) es la siguiente capa. Básicamente su objetivo se centra en un modelo en el que se proporciona un servicio de plataforma con todo lo necesario para dar soporte al ciclo de planteamiento, desarrollo y puesta en marcha de aplicaciones y servicios a través de la Red

El proveedor es el encargado de los recursos en caso de que la aplicación lo requiera, del rendimiento adecuado de la plataforma y de la seguridad de acceso. Para desarrollar los programas se requieren de bases de datos, herramientas de desarrollo y en ocasiones servidores y redes. Con PaaS el usuario únicamente se enfoca en desarrollar, depurar y probar ya que la herramienta necesaria para el desarrollo es proporcionada a través de la Red

Con el uso de PaaS, se separa del hardware al usuario, lo cual es interesante para muchos desarrolladores, y es probable que llegue a remplazar a las empresas de alojamiento tradicionales.

Incluso, también a los administradores de sistemas ya que no se necesita controlar ninguna infraestructura ni hay optimización posible más allá del código y sus algoritmos.

TABLA 2 : Proveedores PaaS

Tipo de Servicio	Ejemplo de Proveedor
Plataformas de desarrollo	Amazon EC2 Google App Engine Microsoft Midori
Bases de Datos	Amazon SimpleDB Microsoft Azure Database
Cola de Mensajes	Amazon Simple Queue Service
Servidores de Aplicaciones	NetSuite Business Operating System Rackspace Cloud Servers Google Apps

Ventajas.

Ventajas comparativas con respecto a las aplicaciones tradicionales son las siguientes:

Entorno de desarrollo y producción: En los procesos tradicionales se tienen al menos dos o tres entornos diferentes para la programación, pruebas y puesta en operación de un programa. En PaaS se puede conseguir que todo el proceso de desarrollo y operación se encuentren en la misma plataforma, lo que disminuye la posibilidad de errores por incompatibilidad de plataformas.

Consideraciones.

Al momento de elegir un proveedor de PaaS se recomienda considerar los siguientes aspectos:

Técnicos: La interoperabilidad de las plataformas de desarrollo y producción debe ser analizada detenidamente, en algunos casos PaaS solo ofrece plataformas basadas en lenguajes de programación de acceso libre por lo que restringe la gama de posibilidades de desarrollo de aplicaciones para el usuario.

Económicos: Se puede negociar con el proveedor de PaaS de un tiempo de prueba gratuito que posibilita al usuario de la oportunidad de adquirir la plataforma luego de verificar las bondades de la misma.

Legales: Se debe especificar un acuerdo en el que se detalle las condiciones de uso de la PaaS de tal forma que el usuario no haga uso indebido de la misma y el proveedor tarife un valor conocido por ambas partes tanto por espacio de almacenamiento así como de recursos computacionales (procesador, memoria y ancho de banda).

1.4.6.3. Infraestructura como Servicio (IaaS)

IaaS corresponde a la capa más baja. La idea básica es la de tercerización de equipos servidores en busca de espacio en disco, sistemas de base de datos, routers, switchs y/o tiempo de procesamiento en lugar de tener un servidor local y toda la infraestructura y personal necesarios para el soporte, mantenimiento y conectividad.

Las IaaS permiten desplazar al proveedor la mayor parte de los factores relacionados con la gestión de los equipos con el concerniente ahorro al pagar solo por lo consumido y olvidarse del manejo y administración de equipos así como su mantenimiento. Por otro lado, IaaS permite una escalabilidad automática o semiautomática, según los acuerdos preestablecidos, de forma que se puedan obtener más recursos según se requiera.

Para hacer una distinción respecto a las PaaS, las IaaS se presentan como una propuesta con mucho más flexibilidad para el uso que el usuario la tenga en mente, pero también requieren mucho más involucramiento por parte del usuario en lo que a instalación, configuración y mantenimiento del software se refiere.

TABLA 3 : Ejemplo de Soluciones IaaS

Tipo de Servicio	Ejemplo de Proveedores
Procesamiento	Amazon Elastic Compute Cloud Sun Network.com Elastic Host Eucalyptus Openstack Open Nebula
Distribución de contenido a través de servidores virtuales	Akamai Amazon cloudFront
Almacenamiento	Amazon S3 Jungle Disc Microsoft Skydrive ICloud -Apple Youtube Openstack
Admin. de Sistemas	Elastra Engine Yard Rackspace Savvis
Admin. de Alojamiento	Godaddy Layered Technologies Digital Realty Trust Rackspace

Ventajas

Ventajas comparativas con respecto a las aplicaciones tradicionales son las siguientes:

Rendimiento: el usuario se encarga de la gestión de los recursos asignados de acuerdo a sus necesidades incrementando el rendimiento de sus aplicaciones de acuerdo a la demanda interna, por otro lado el proveedor que gestione la plataforma de tal manera que los recursos son notablemente mejor aprovechados, beneficiando con un menor precio al usuario.

Infraestructura: Los equipos son propiedad del proveedor de IaaS por lo tanto la responsabilidad de mantenimiento y renovación ya no le corresponden al usuario. Sin embargo se debe considerar al momento de elegir un proveedor de IaaS que demuestre su planificación respecto a la permanencia del servicio en el transcurso del tiempo, lo que menos desea el usuario es que su proveedor desaparezca repentinamente.

Fiabilidad: La inversión que implica la redundancia de sistemas en muchos casos no es posible asumirla por parte del usuario, bajo la modalidad de IaaS es muy posible que las pequeñas organizaciones accedan a este tipo de beneficios incluidos en el servicio contratado por defecto.

Consideraciones

Al momento de elegir un proveedor de IaaS se recomienda considerar los siguientes aspectos:

Técnicos: Puede ser un limitante la cantidad de Sistemas Operativos y aplicaciones tales como las de bases de datos o alojamiento web que el proveedor puede ofrecer. Dicha limitación será traspasada al usuario de tal manera que debe considerarse al momento de planificar alguna nueva

implementación. Por otro lado como aspecto favorable está la posibilidad de que los usuarios puedan acceder a la infraestructura arrendada desde cualquier lugar a través de una conexión de datos fiable.

Económicas: IaaS ofrece al usuario varios modelos de pago entre los que se detalla; pago por consumo de recursos utilizados en un periodo de tiempo acordado, pago por recursos reservados de forma exclusiva o pago por recursos utilizados en un pico de carga.

Legales: Como se mencionó anteriormente, es importante que en el acuerdo de servicio se determine la duración del servicio a fin de evitar sorpresivos cierras durante el periodo de operación.

1.4.7. Síntesis:

En el siguiente cuadro se puede apreciar en resumen los conceptos analizados anteriormente bajo la definición elaborada por el Instituto NIST de Estados Unidos

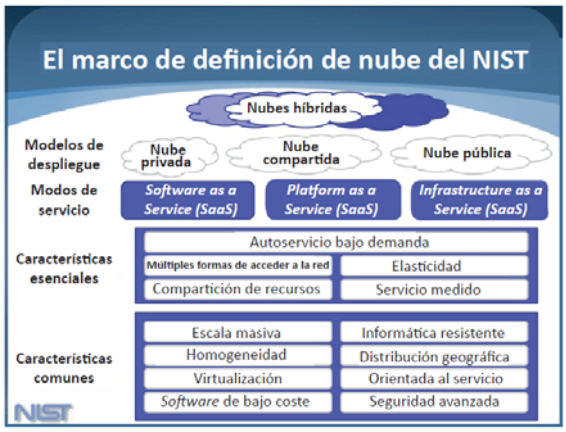


FIGURA 3 : Conceptualización de Nube Computacional

Temas a considerar sobre la nube computacional y el desarrollo hacia el futuro.

El mercado mundial de servicios *Cloud Computing* tuvo en 2009 un valor superior a 17.000 millones de dólares, estando previsto un crecimiento anual superior al 27% durante los siguientes cuatro años, hasta alcanzar los 44.200 millones de dólares en 2013 (IDC). El área de mayor potencial de crecimiento es la de almacenamiento, dentro de la modalidad *IaaS - Infraestructure as a Service*. Según las previsiones del IDC en 2014, la cuota de mercado de las tecnologías *cloud* habrá alcanzado el 12% del mercado de tecnología, con un crecimiento anual cinco veces superior al de las tecnologías *in-house.* Para el 2020 se estima que la mayoría de las personas realizaran sus labores en aplicaciones basadas en el Internet (Project, 2010)

Según Gartner, *Cloud Computing* será una de las diez tecnologías estratégicas de los próximos años, tanto en la modalidad de nubes privadas como de nubes públicas, con cerca de un 40% de las organizaciones incluyendo en sus presupuestos el gasto en el 2017. La madurez de las nubes privadas conlleva que las nubes híbridas sean el siguiente salto tecnológico. (Babcock, 2013)

Cisco Networks considera que el crecimiento del tráfico será de hasta un 35% anual hasta el 2017 llegando a niveles de los 5,3 zettabytes (1 zettabyte corresponde a 1 billón de terabytes) y que la mayoría de este tráfico, hasta un 69%, será originado por el uso de las aplicaciones provistas por las nubes computacionales. (Cisco, 2013)

1.4.8. Certificaciones para Nubes Computacionales

Aunque se trata de una cuestión reciente y aún por desarrollar plenamente, cabe mencionar que aparte de los Acuerdos de Nivel de Servicios establecidos como parámetros a seguir entre los proveedores, existen ciertos estándares que permiten evaluar la capacidad y calidad del proveedor de la Nube Computacional y que pueden constituir un factor de confianza a la hora de adquirir sus servicios. Entre estos estándares destacan:

• SAS 70 - estándar internacional que proporciona una guía para que un auditor independiente emita una opinión sobre la descripción de controles de la organización a través de un Reporte de Servicio del Auditor.

• SysTrust - Una auditoria bajo los principios de SysTrust permite obtener un informe sobre la fiabilidad del sistema atendiendo a criterios como la Disponibilidad, Seguridad, Integridad y Confidencialidad de la información.

• ISO 27001 - estándar para la seguridad de la información ligado a la ISO 27002 que recoge una guía de buenas prácticas con los objetivos de control y controles recomendables en cuanto a seguridad de la información.

2. CAPÍTULO II: OPENSTACK

2.1. INTRODUCCIÓN A OPENSTACK

2.1.1. Concepto:

OpenStack es una recolección de varias tecnologías bajo licencia de código abierto Apache 2.0 orientadas a la entrega de sistemas de nube computacional a gran escala, es apoyado y soportado empresas como Cisco, Dell, Citrix, Intel, Canonical y Red Hat entre más de 120 adscritas hasta el momento. El objetivo final del proyecto es facilitar un sistema de nube computacional de código abierto y fácilmente adaptable a sistemas hardware de bajo coste, facilitando su implantación en empresas de todos los niveles. (Openstack, Openstack Org, 2013)

En sus inicios, el proyecto se encontraba a cargo de dos gigantes de la industria de aplicaciones tecnológicas, por un lado Rackspace, que ha liberado el código de sus servicios Cloud Files y Cloud Servers; y por otro NASA que ha colaborado con el código de su plataforma Nébula para potenciar su contraparte computacional. Ambos códigos forman núcleo inicial de OpenStack y han sido desarrollados utilizando Python, ofreciendo la posibilidad a usuarios o empresas de **crear sus propios servicios IaaS de Nube computacional**. Sin embargo durante el último año se han integrado a la dirección del proyecto otros gigantes como HP y Cisco para aportar sus experiencias en cada uno de sus campos de especialización. La directiva del proyecto se elige ahora por votación.

Openstack comprende una colección de varios proyectos de Licencia Abierta cuyo desarrollo y mantenimiento está a cargo de los usuarios o comunidades inscritas para ello. Al momento de la realización del presente documento se encontraban adscritas al proyecto cerca de 175 compañías y 13700 personas, datos que reflejan la importancia que tiene esta tecnología tanto desde el punto de vista científico como desde su aspecto comercial, ya que si buscamos en el Internet, es muy fácil encontrar varias implementaciones comerciales de la aplicación de Openstack como núcleo y base de los servicios ofertados. (Openstack, Openstack Org, 2013)

2.1.2. Componentes de Openstack

Openstack está formado por tres componentes principales; el componente de **infraestructura computacional**, el de **almacenamiento de objetos** e imágenes y finalmente del relacionado a los **servicios de red**. Todos ellos forman parte de las herramientas para crear y administrar los servicios en una nube de equipos computacionales, entre los que se incluye la ejecución de Instancias, la administración de los componentes de Red y los permisos de acceso a la misma.

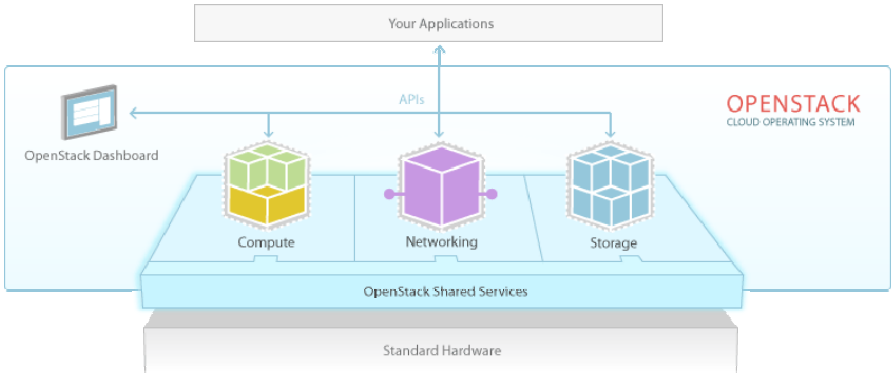


FIGURA 4 : Distribución de nodos Openstack

Se entiende por Instancias a las Máquinas Virtuales o recursos computacionales administrados por Nova, creadas por imágenes almacenadas en Glance y elegidas de varias Recetas (flavors) predeterminados o personalizados.

Al ser un proyecto en permanente desarrollo, los administradores del sistema lo han separado a su vez en diferentes sub-proyectos o implementaciones, cada una de ellas con su propósito específico y aplicación dentro de concepto general con el objetivo de favorecer la inclusión de muchos desarrolladores que aporten al proyecto de ideas frescas y mejoras significativas. En sus inicios, Openstack contaba con cerca de 130 desarrolladores y ahora superan las 10000 personas involucradas de una u otra manera en los avances de las diferentes versiones que ha salido a la luz durante los últimos 2 años de desarrollo.

Siendo el **Componente Computacional (Compute)** el que controla la nube administrando la ejecución de las instancias para cada usuario o grupo

de usuarios, también se encarga de configurar los recursos de la infraestructura asignados a cada una de ellas. Este componente se maneja bajo los proyectos denominados Nova y Glance**.**

El **Componente de Almacenamiento (Storage)** es un sistema de depósito de gran capacidad, altamente escalable y con sistema de redundancia incluido. Su uso es variado, desde un sistema sencillo de almacenamiento de archivos hasta utilizarlo como un sistema de provisión de imágenes. Puede utilizarse como una herramienta para el desarrollo de aplicaciones de almacenamiento cuya capacidad crezca de acuerdo a la demanda de las aplicaciones ejecutadas como por ejemplo los programas de bases de datos. Dentro de Openstack a este componente se desarrolla a través de los proyectos **Swift y Cinder**.

El tercer módulo principal de Openstack es el **Componente de Red (Networking)** que se desarrolla bajo el proyecto **Neutron**, cuyo objetivo es el de gestionar los recursos de interconexión y las comunicaciones generadas entre el resto de componentes y subcomponentes, así como también hacia con otro tipo de nubes computacionales. Antes de la implementación de Neutron, la administración de los recursos de red era parte del proyecto Nova bajo el API *nova-network.*

Inicialmente los componentes Nova, Glance y Swift formaban parte de la distribución de Openstack conocida como “Diablo”. Sin embargo a inicios del 2012 salió a luz pública la distribución llamada “Essex” en la que se incluyeron como un avance, especialmente orientado al usuario, los componentes

Horizon y Keystone. El primero enfocado al desarrollo de un Panel de Control y Administración de los usuarios de la Nube a través de una interfaz gráfica; y Keystone como componente administrador de las seguridades dentro de la Nube y en el que se ha denominado como el aporte fundamental dentro de los impulsares del Cloud Computing para apuntalar las implementaciones comerciales de Openstack. Neutron fue parte del desarrollo de la versión "Havana" lanzada a finales del 2013 en la que se incluyen otros dos proyectos más: **Ceilometer y Heat**

En la actualidad los proyectos que conforma el sistema de Openstack son:

- Openstack Compute - proyecto Nova
- Openstack Netwroking – proyecto Neutron
- Openstack Object Storage – proyecto Swift
- Openstack Block Storage – proyecto Cinder
- Openstack Identity – proyecto Keystone
- Openstack Image Service – proyecto Glance
- Openstack Dashboard – proyecto Horizon
- Openstack Telemetry – proyecto Ceilometer
- Openstack Orchestration – proyecto Heat

A continuación un detalle de las versiones de las diferentes distribuciones de Openstack desde sus inicios:

TABLA 4 : Listado de Proyectos Openstack

Nombre	Versión	Fecha	Proyectos incluidos
Icehouse	En desarrollo	Abril 17, 2014	
Havana	2013.2	Octubre 17, 2013	Nova, Glance, Swift , Horizon, Keystone, Neutron (Quantum), Cinder , Oslo, Heat, Ceilometer
	2013.2.1	Diciembre 16, 2013	
Grizzly	2013.1	Abril 4, 2013	Nova, Glance, Swift , Horizon, Keystone, Quantum, Cinder , Oslo
	2013.1.1	Mayo 9, 2013	
	2013.1.2	Junio 6, 2013	
	2013.1.3	Agosto 8, 2013	
	2013.1.4	Octubre 17, 2013	
Folsom	2012.2	Septiembre 27, 2012	Nova, Glance, Swift , Horizon, Keystone, Quantum, Cinder , Oslo
	2012.2.1	Noviembre 29, 2012	
	2012.2.2	Diciembre 13, 2012	
	2012.2.3	Enero 31, 2013	
	2012.2.4	Abril 11, 2013	
Essex	2012.1	Abril 5, 2012	Nova, Glance, Swift , Horizon, Keystone
	2012.1.1	Junio 22, 2012	
	2012.1.2	Agosto 10, 2012	
	2012.1.3	Octubre 12, 2012	
Diablo	2011.3	Septiembre 22, 2011	Nova, Glance, Swift
	2011.3.1	Enero 19, 2012	
Cactus	2011.2	Abril 15, 2011	Nova, Glance, Swift
Bexar	2011.1	Febrero 3, 2011	Nova, Glance, Swift
Austin	2010.1	Octubre 21, 2010	Nova, Swift

El presente proyecto de Titulación tendrá como base a la distribución “Havana 2013.2” ya que incluye a Dashboard de Horizon como una herramienta que permitirá al administrador del servicio dentro del Departamento de Eléctrica y Electrónica el suministro de los diferentes recursos de la nube computacional de acuerdo a la necesidad de los estudiantes, sin embargo no incluye los proyectos Ceilometer y Heat ya que se hallan en etapa de prueba y su implementación no se encuentra aun totalmente documentada.

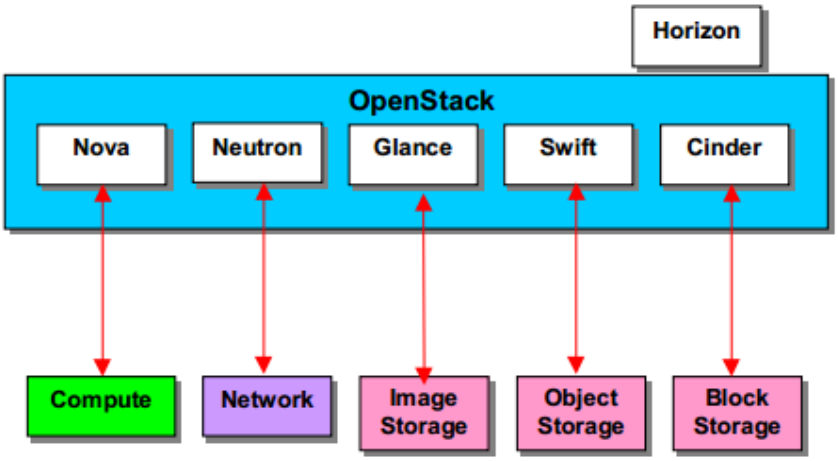


FIGURA 5: Componente vs. Aplicación

A continuación un breve resumen de los componentes y su interacción con los demás:

- **Infraestructura Computacional – Nova:** Dirige, almacena y recupera los discos virtuales y recursos asociados a Glance y les asocia los recursos de infraestructura necesarios para su ejecución.

- **Servidor de Imágenes – Glance:** Almacena y recupera los discos virtuales activos en Swift
- **Repositorio de Objetos – Swift:** Servicio de Almacenamiento.
- **Panel de Control – Horizon:** Corresponde al aplicativo gráfico de interacción entre los servicios de Openstack y los usuarios.
- **Autenticación – Keystone:** Se utiliza para autenticar la ejecución de los diferentes servicios bajo la modalidad de ID y contraseña únicos para cada instancia.
- **Red – Neutron:** inicialmente denominado como **Quantum**, maneja las redes y las direcciones IP de la nube de tal manera que no se conviertan en un cuello de botella que influya negativamente en el desempeño de los otros componentes.
- **Telemetría – Ceilometer:** Genera la información relacionada a medición y monitoreo de la nube de tal manera que dicha información sea utilizada para medir el desempeño de la misma, así como también para procesos de facturación de los recursos utilizados.
- **Orquestación – Heat:** este componente se encarga de organizar y coordinar los diferentes estamentos de nube predefiniendo patrones de uso de acuerdo a las aplicaciones requeridas por el usuario.

Técnicamente, los componentes de Openstack han sido programados usando Phyton y la intercomunicación entre ellos se realiza mediante instrucciones API bajo estándares generales y que en conjunto forman un todo

diseñado para proveer un Sistema Operativo para la Nube Computacional altamente escalable.

Los diferentes Accesos Programables que conformas los proyectos de Openstack se programan bajo la arquitectura REST. Para comprender las bondades de este API, nos referimos a su definición y alcance, siendo REST determinado de la siguiente manera: "***Transferencia de Estado Representacional*** *(Representational State Transfer) o* ***REST*** *es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.*"

Se los puede definir también como un protocolo de tipo Cliente/Servidor para acceder a los objetos modelados sin necesidad de conocer su estado previo. Se denominan como RESTful APIs a los sistemas que siguen este principio de operación. Para nuestro caso, los RESTful APIs de Openstack han sido implementados usando los comandos HTTP: GET/PUT/POST/DELETE en combinación del formato ligero para el intercambio de datos JSON

Cada uno de los APIs permiten una intercomunicación entre los usuarios y los diferentes servicios o proyectos a través de un sistema de colas de mensajes siguiendo el protocolo de mensaje AMQP (Advanced Message Queuing Protocol), en este caso Openstack utiliza la aplicación de negociación de mensajes en código abierto denominada RabbitMQ, sin embargo ya se está en marcha el proyecto Marconi propio de Openstack y especializado en el manejo de mensajería y administración de solicitudes a

gran escala. Cabe mencionar que cada uno de los APIs de los diferentes proyectos puede ser ejecutado mediante un instrucción a través de la interfaz de Líneas de Comando o CLI, lo que resulta muy conveniente y útil desde el punto de vista de desarrollo de las diferentes implementaciones de Openstack porque permite un análisis pasos a paso de las actividades que se realizan en caso de que se presente algún error.

El siguiente gráfico tomado del portal de Openstack hace una descripción de la arquitectura de Openstack versión Havana en al que se detallan los proyectos que conforman Openstack y la interrelación entre ellos.

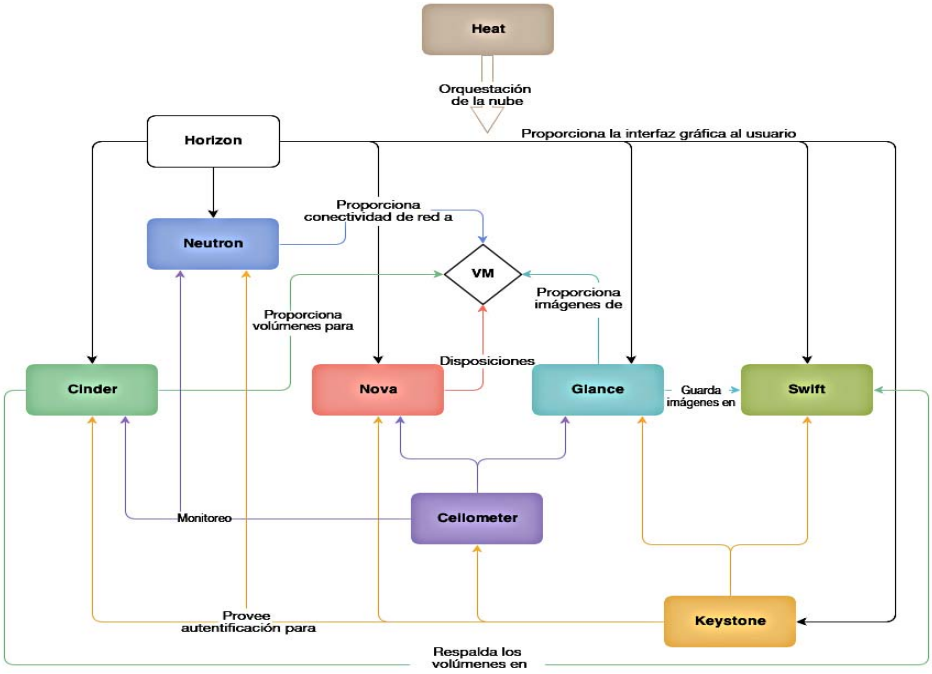


FIGURA 6 : Arquitectura Conceptual

Por otro lado, Openstack también ofrece un diagrama, pero en este caso se trata de la arquitectura lógica de la plataforma:

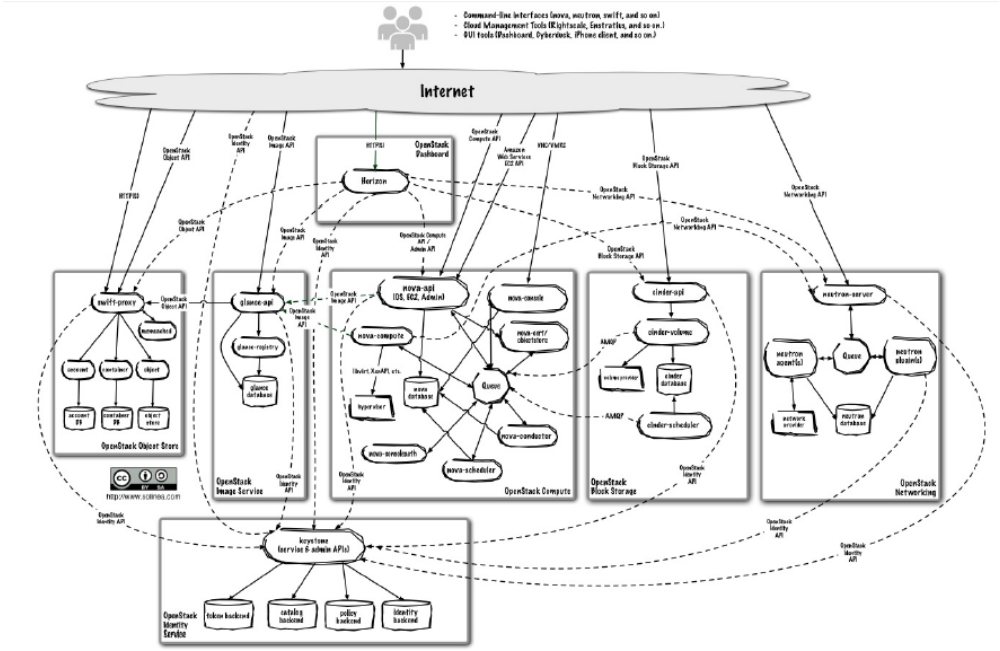


FIGURA 7 : Arquitectura Lógica

A simple vista puede resultar complicada la interpretación del gráfico anterior, pero ahí se detalla la manera en que los diferentes subcomponentes de los proyectos de Openstack se comunican entre ellos, nótese lo siguiente:

- Todos los componentes tienen un API propio para recibir instrucciones por parte de los usuarios.
- En Openstack Compute, todos subcomponentes se comunican a través de Queue o cola de mensajes (RabbitMQ).
- Todos los componentes se comunican con Keystone para solicitar las credenciales de autenticación, acceso y operación de las instancias dentro de la nube.
- Horizon recibe consultas mediante el protocolo http, lo que lo hace accesible desde cualquier consola o sistema operativo.

Este sistema de encolamiento basado en RabbitMQ (http://www.rabbitmq.com/) es el que permite un crecimiento escalado de las aplicaciones y de la infraestructura que utiliza la nube computacional de modo horizontal, es decir que podemos agregar más equipos de almacenamiento o procesamiento que se comunique a la misma cola de solicitudes de manera más rápida y sencilla. Por otro lado permite que los diferentes proyectos de Openstack puedan evolucionar y desarrollarse por separado, al momento ya hay compañías que utilizan los componentes de Swift para implementar sistemas de almacenamiento masivo sin necesidad del resto de componentes, así mismo hay otras empresas que ya ofrecen sus propias interfaces gráficas de control basados en Horizon.

Así mismo algunos autores han separado a Openstack en diferentes capas de acuerdo a su desempeño dentro de la nube. Siendo la capa superior la de Interfaz gráfica referente a la interacción con el usuario o administrador de la nube. En esta capa encontramos al proyecto Horizon.

La segunda capa es la de Servicios Elásticos donde se encuentran los componentes combinados de Openstack que organizan el funcionamiento de la nube y la administración de las máquinas virtuales, aquí tenemos a Nova y Cinder

Una tercera capa corresponde a los Servicios Compartidos, entre ellos por ejemplo el servicio de Identidad o Keystone que es solicitado por los demás componentes para generar claves de autenticación necesarios para Nova, Horizon, Cinder y Swift.

Por último, la cuarta capa que corresponde a los de Recursos y Datos, aquí se incluyen los proyectos complementarios y externos de Openstack, como por ejemplo RabbitMQ para la administración de las colas, MySQL para el manejo de las bases de Datos, KVM para la administración de las máquinas virtuales y Ceph para el almacenamiento de datos.

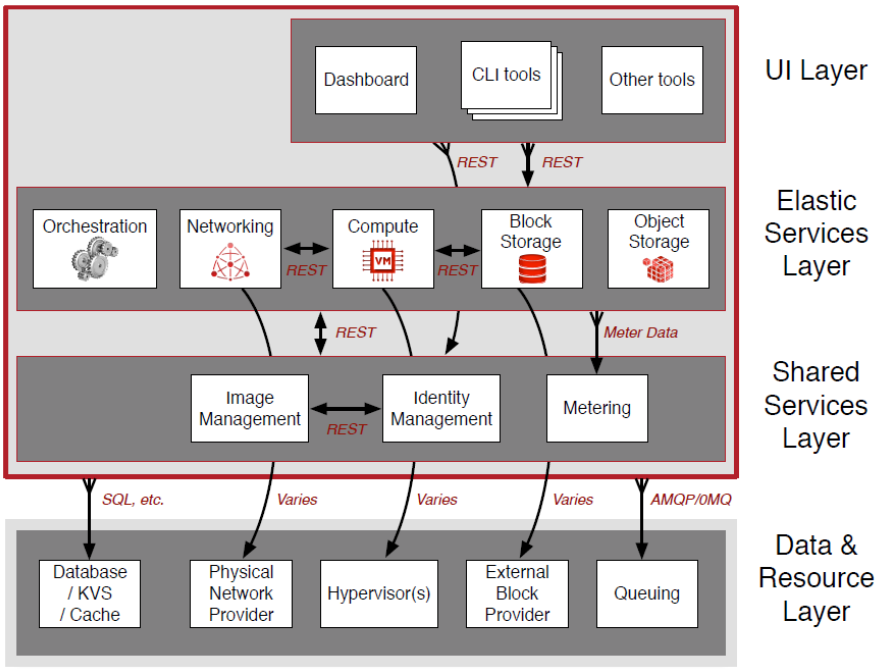


FIGURA 8 : Distribución interna de Openstack

2.1.3. Descripción de los componentes de Openstack.

En el presente capítulo haremos un análisis más a detalle de cada uno de los servicios anteriormente descritos, sin embargo debemos iniciar con la introducción del concepto de Hipervisores:

*"Un **hipervisor** (en inglés hypervisor) o **monitor de máquina virtual** (virtual machine monitor) es una plataforma que permite aplicar diversas*

*técnicas de control de **virtualización** para utilizar, al mismo tiempo, diferentes sistemas operativos (sin modificar o modificados en el caso de paravirtualización) en una misma computadora. Es una extensión de un término anterior, «supervisor», que se aplicaba a los kernels de los sistemas operativos."* (Wikipedia W. , 2013)

Así mismo la definición de virtualización:

*"..**virtualización** es la creación -a través de software- de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red.." "..Esta capa de software (VMM) maneja, gestiona y arbitra los cuatro recursos principales de una computadora (CPU, Memoria, Almacenamiento y Conexiones de Red) y así podrá repartir dinámicamente dichos recursos entre todas las máquinas virtuales definidas en el computador central. Esto hace que se puedan tener varios ordenadores virtuales ejecutándose en el mismo ordenador físico"* (Wikipedia, virtualización, n.d.)

Más adelante hablaremos de la aplicación de los Hipervisores dentro del proyecto de Titulación pero es de gran importancia haber incluido el concepto ya que ayudará a comprender la descripción de los componentes de Openstack.

2.1.3.1. Componente Computacional

Se puede implementar el componente computacional o Nova ya sea en equipos sencillos o en equipos de alto poder de procesamiento, todo depende

del número de instancias a ejecutar y del nivel de disponibilidad requerido. Opera en conjunto con múltiples Hipervisores, sin embargo la mayoría de las implementación se han con realizado utilizando KVM y XenServer.

2.1.3.1.1. Características

Las características más importantes son:

Permite el uso de hardware básico no propietario en su implementación: lo que lo hace muy versátil al momento de probarlo previo a una implementación en producción.

Administra, coloca y limita el uso de los recursos asignados al usuario: Controla la designación de recursos físicos para la ejecución de una instancia

Automatiza el aprovisionamiento y el manejo de los recursos vía API: Dirige a los diferentes componentes de Openstack, a este proceso también se lo denomina Orquestación de recursos.

Tiene acceso al administrador y a los servicios por separado vía CLI: Permite al usuario la interactuar con los diferentes componentes a través de la aplicación de diferentes instrucciones que a su vez controlan la operación de la nube.

Asigna los recursos de red de tipo Flat, Flat DHCP, VLAN DHCP e IPv6: permitiendo un modelo de red flexible y que se adapte a las necesidades de la instancia y a la vez de la aplicación que se ejecuta sobre ella.

Designa de forma automática y segura las credenciales de autenticación: Gestiona con Keystone las solicitudes de IDs para las instancias y su vez controla que no haya un exceso de consultas de dichos APIs

Permite una implementación escalable y de alto desempeño: Basada en una arquitectura distribuida y asincrónica, permite implementar soluciones de Alta Disponibilidad, repartiendo los diferentes componentes en varios nodos.

Facilita la implementación y administración de las imágenes de las máquinas virtuales: Junto a Glance administra las imágenes almacenadas en Swift.

Incrementa la productividad de la nube: Controlando el tiempo de vida de las máquinas virtuales en tiempo real, incluyendo los diferentes cambios en los recursos asignados sucedidos como resultado de la operación de máquina virtual.

Permite la asignación de direcciones IP: Controla la comunicación entre las máquinas virtuales hacia el exterior a través de IP flotantes que son devueltas una vez terminada la instancia relacionada.

Genera un rápido aprovisionamiento de las máquinas virtuales almacenadas: Controla el desempeño del programa Hipervisor.

Más detalles de las características podemos tener al analizar los proyectos que conforman al componente computacional; *Nova* y *Glance.*

2.2. NOVA

Nova es el proyecto principal del componente computacional o controlador de la nube de Openstack ya que es el lugar donde se crean las máquinas virtuales y administra de manera centralizada los demás nodos de las implementaciones mediante el envío de mensajes a los otros componentes a través de la cola de mensajería. Así mismo también administra los recursos de infraestructura que se asignan a cada instancia. Su módulo principal está programado en lenguaje Phyton.

Se puede entender a Nova como el componente más complejo de Openstack ya que ejecuta más procesos que los demás componentes de la nube e incluye los accesos programables vía RESTful API de consulta que el usuario envía a las máquinas virtuales en funcionamiento. Interactúa con el servicio Identity (Identidad – Keystone) por el tema de autenticación, con Image (Glance) para las imágenes, con Dashboard (Horizon) para la interfaz gráfica administrativa para los usuarios.

2.2.1. Características

Entre las características principales se encuentran las siguientes:

Control de las instancias: por instancia se denomina a la máquina virtual que ha sido iniciada y terminada por Nova.

Asignación de IPs públicas: Previo a la inclusión del proyecto Neutron, Nova se encargaba de la conexión de las máquinas virtuales a la red.

Agregar bloques de almacenamiento: Junto al componente Cinder, Nova agrega bloques de almacenamiento según los requerimientos de las máquinas virtuales activas.

Creación, modificación y eliminación de las seguridades de las máquinas virtuales relativas a un grupo predeterminado.

Muestra las consolas de las instancias: Interactúa con Dashboard a través de los APIs con el objetivo de transferir información en tiempo real del estado de las máquinas virtuales creadas a los usuarios.

2.2.2. Procesos

La lista de procesos que incluye Nova se detallan a continuación:

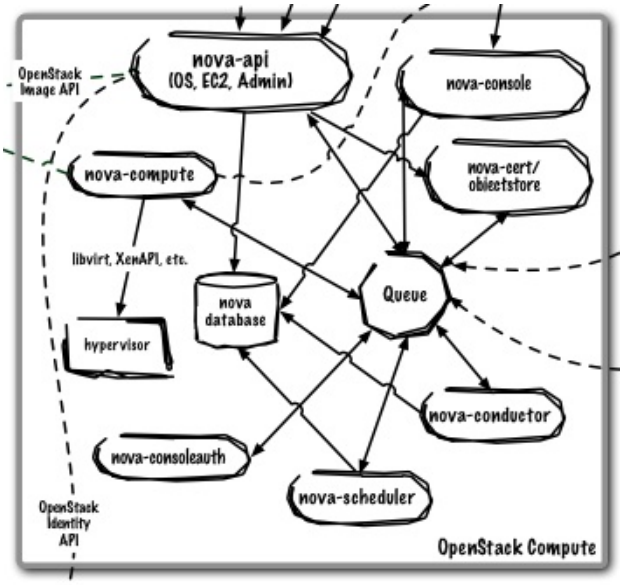


FIGURA 9 : Conceptual NOVA

- Nova-api.- compuesto por los programas ejecutados en segundo plano (deamons): nova-api, nova-api-os-compute, nova-api-ec2, nova-api-metada y nova-api-all. Todos ellos aceptan y responden a los llamados generados por el usuario vía API e inician las diferentes actividades para poner en marcha a las instancias.

- Nova-compute.- es el proceso que se ejecuta en segundo plano que se encarga de la creación y terminación de las máquinas virtuales a través del API del hipervisor utilizado, por ejemplo XenServer lo hace con el XenAPI y KVM lo hace con libvirt.

- Nova-conductor.- Actualiza los cambios en la base de datos generados por los otros procesos, por ejemplo en la creación de máquinas virtuales por parte del hipervisor, nova-conductor se encarga de actualizar el estado de las base de datos de las mismas en la nube.

- Nova-schedule.- Siendo el proceso más complejo de Nova, se encarga de organizar el estado de las solicitudes de instancias en la cola de mensajes y determina el equipo en el que alojará y se ejecutará la máquina virtual.

- Nova-console, nova-consoleauth, nova-xvpvncpproxy, nova-spicehtml5proxy.-Permiten que el usuario pueda actuar con sus instancias vía consola CLI a través de un proxy.

- Nova-network, nova-dhcpbridge.- son los procesos encargados de las tareas relacionadas a la interconexión de las máquinas virtuales a través de la red de datos, incluye el almacenamiento en la base de datos de las direcciones IP asignadas a las máquinas virtuales y su tiempo de vida. Este

proceso ha sido separado de Nova en el proyecto de Openstack Networking o Neutron

Con respecto a la base de datos que utiliza Nova, es el lugar donde se almacenan los datos relacionados al ciclo de vida de la instancia, ahí se encuentran los estados de creación y ejecución de las instancias en la nube así como también la información relacionada a las conexiones de red. La base de datos más utilizada para esta implementación es MySQL, sin embargo al momento de instalar Openstack, la base de datos SQlite es la que se encuentra activada por defecto. PostgreSQL también puede ser utilizada pero la documentación que respalda su implementación no es muy amplia.

En el siguiente gráfico generado por el grupo de Cloudscaling (Bias, Cloud Scaling, 2013) en la que se puede apreciar la relación entre los diferentes componentes de Nova más a detalle.

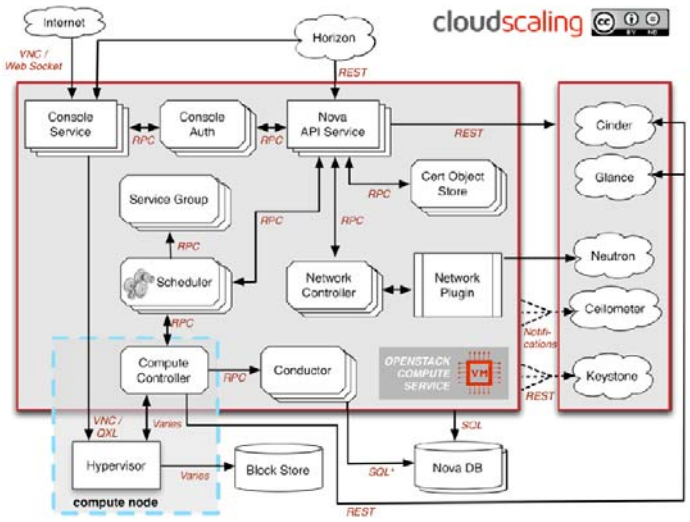


FIGURA 10 : Lógico NOVA

En ambientes o implementaciones que cuentan con más de un nodo computacional donde se ejecutan las instancias o máquinas virtuales, se puede apreciar que todos ellos se comunican periódicamente a través de la cola de mensajería con nova-scheduler transfiriendo su estatdo, en el que se incluye información relacionada con los recursos disponibles. Esta información es almacenada por nova-scheduler y es utilizada al momento de recibir una petición posterior y sea necesario determinar el host o anfitrión para la máquina virtual.

Por defecto, el componente scheduler opera como un filtro que clasifica a los anfitriones bajo el siguiente criterio:

- AvailabilityZoneFilter: Si pertenecen a la zona relacionada a la petición

- RamFilter: Si es que cuenta con suficiente RAM para tender la petición.

- ComputeFilter: Separa los anfitriones que no están disponibles para atender al requerimiento.

- ComputeCapabilitiesFilter: Verifica si el anfitrión satisface los requisitos necesarios asociados a la instancia.

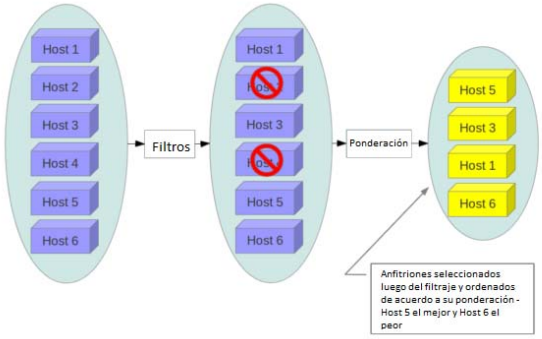


FIGURA 11 : Asignación de Host

Así mismo otro parámetro que utiliza nova-scheduler para elegir al equipo anfitrión de la instancia es el denominado weight o peso. Para ello utiliza un sistema de báscula para determinar el coste de operación del anfitrión si atendiera a la petición. Esta báscula se basa principalmente en la métrica obtenida por la clase RamWeigher que determina el peso de un anfitrión y luego le otorga un puesto de preferencia respecto a los otros hosts disponibles.

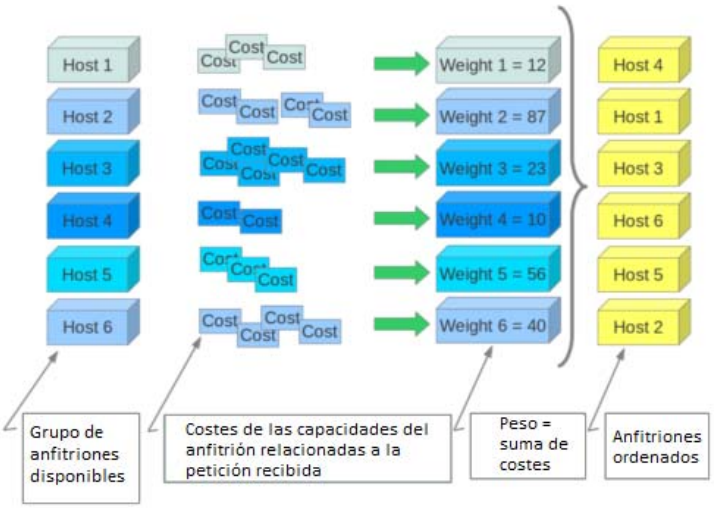


FIGURA 12 : Ordenamiento de los Hosts

2.3. KEYSTONE – Autenticación y Autorización

El proyecto de Openstack que se refiere a la administración de credenciales se denomina Keystone. El alcance de este proyecto va más allá del control de las credenciales de autenticación ya que se comunica con los demás Proyectos para transferir Políticas, Roles, Catálogos y Tokens relacionados a cada uno de ellos.

Siendo los Catálogos definidos como el listado de los componentes de Openstack disponibles, así como su información de conexión (URL, puerto y contexto).

Otra definición importante es la Proyecto o Contenedor (Tenant); y es el agrupamiento de alto nivel al que se le relacionan recursos propios.

A los Usuarios se los define como parte de un proyecto o de varios a la vez y tienen Roles asignados. Actúan en nombre del Contenedor (Tenant) para generar el aprovisionamiento de Recursos.

Por Roles se define a los privilegios relacionados a Usuarios o Servicios, siendo los roles por defecto; Admin y Member.

Las Cuotas son límites o asignaciones establecidas por los administradores de los Recursos y están asociados a los Contenedores y no a los Usuarios. Por ejemplo en Openstack Compute, como Cuotas se puede definir al tamaño, la cantidad de CPU virtual o el límite de memoria RAM utilizada por cada Instancia.

Por último los Tokens son elementos de autenticación temporales utilizados por Keystone para comunicarse con los APIs de los demás componentes de Openstack. Su validez es de hasta 24 horas y son transmitidos cada vez que los clientes hacen una llamadas a un servicio.

Es un servicio crucial para la operación de la nube ya que apuntala la autenticación y verificación entre los demás servicios de la nube de Openstack, por tal motivo es el primer componente que instalaremos en la parte práctica del proyecto.

En el proceso de autenticación con el servicio de Identidad de Openstack, se envía de regreso un Token de autorización, el mismo que es compartido por todos los servicios una vez que ha sido validado. Por lo tanto este Token es utilizado como la credencial del usuario para acceder a los demás servicios, como por ejemplo a Openstack Storage o Compute. Justamente por lo mencionado anteriormente, es primordial la instalación de Keystone desde el inicio de la instalación de la nube, de tal manera que podamos crear los roles apropiados para cada usuario y servicio relacionado, así como de los contenedores y los accesos a la infraestructura física y virtual.

2.3.1. Características

Entre las características más importantes de Keystone podemos detallar las siguientes:

o Autenticación de los usuarios y sus Tokens para acceder a los servicios.

- Almacena la información de los usuarios y sus roles para luego aprobar su acceso a los servicios.
- Proporciona un catálogo de los servicios proporcionados por la nube.
- Crea las políticas relacionadas a los usuarios y a los servicios.

La arquitectura de Keystone es muy sencilla de comprender:

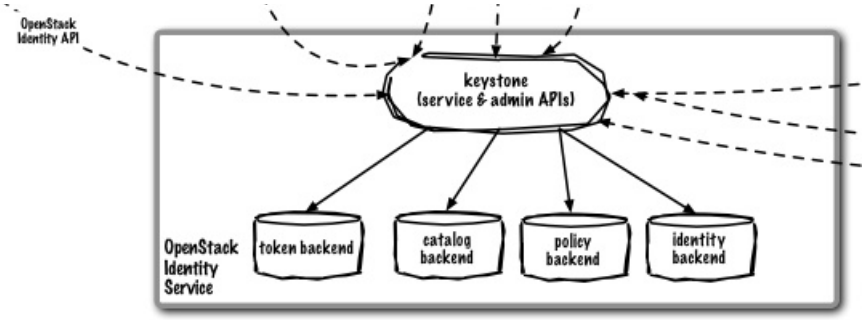


FIGURA 13 : Lógico KEYSTONE

Keystone maneja las solicitudes vía API así como también proporciona los servicios de identificación, políticas y Tokens.

Las funciones de Keystone operan con diferentes tipos de conectores, lo que facilita varios tipos de usos de sus servicios, la mayoría soporta backends estándares como LDAP, SQL y PAM.

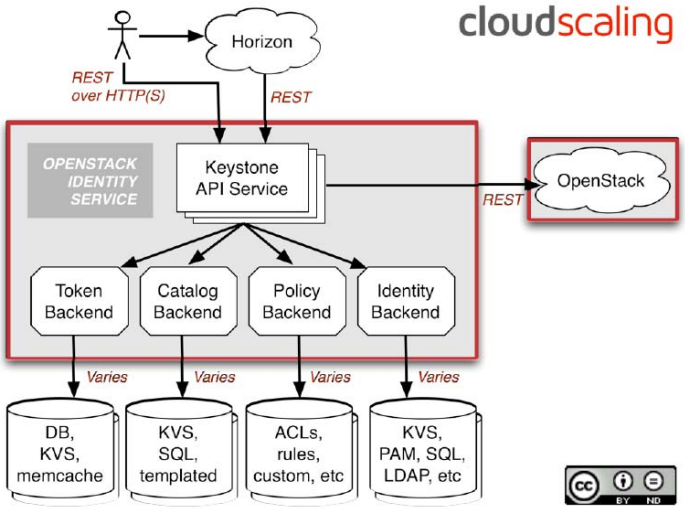


FIGURA 14 : Conceptual KEYSTONE

2.4. GLANCE – Almacenamiento de imágenes

Openstack Image Store almacena de modo centralizado las imágenes virtuales para disposición de los usuarios y de los otros componentes de la nube.

2.4.1. Características

Entre sus características se detallan:

- Almacenamiento de las imágenes públicas y privadas que el usuario puede utilizar para levantar una instancia.
- Los usuarios pueden consultar y enlistar las imágenes que se encuentran disponibles a través del API de conexión.
- Entrega las imágenes a Nova para que se ejecute la instancia solicitada por el usuario.

- Puede generar imágenes de instancias en ejecución a modo de respaldo en caso de fallos.

2.4.2. Procesos

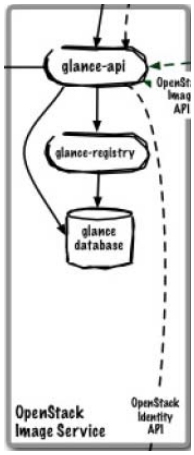


FIGURA 15 : Lógico GLANCE

La lista de procesos que incluye Glance se detalla a continuación:

- Glance-api.- acepta los llamados relacionas a la búsqueda, recuperación y almacenamiento de las imágenes.
- Glance-registry.- recupera, procesa y almacena la información de las imágenes relacionada al tamaño, el tipo, etc.

Al igual que Nova, la base de datos que utiliza Glance para almacenar la información administrada por glance-registry puede ser MySQL o SQLite

En el siguiente gráfico se puede apreciar la interacción de Glance con los otros componentes de Openstack.

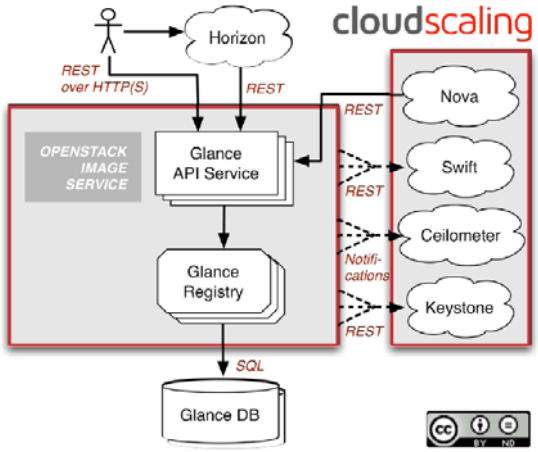


FIGURA 16 : Conceptual GLANCE

Aquí se puede ver a Glance actuando de forma recurrente con los otros componentes, acepta, a través de los APIs, las solicitudes de imágenes generadas por el componente Nova y a su vez puede almacenar los discos de las imágenes en el servicio de almacenamiento controlado por Swift.

La siguiente gráfica describe la relación entre alguno de los objetos anteriormente mencionados.

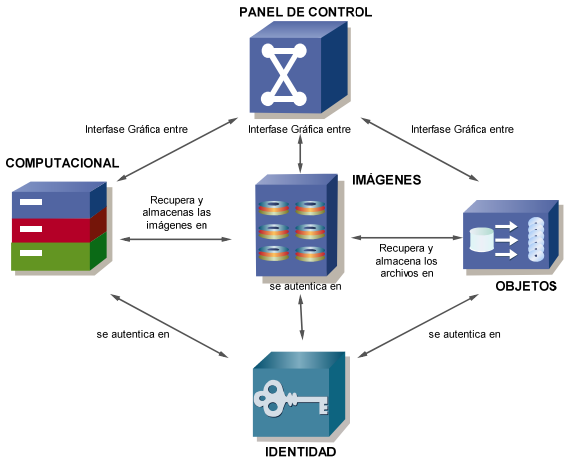


FIGURA 17 : Diagrama de Interacción GLANCE

Entre los formatos de imágenes que Glance puede administrar encontramos los siguientes:

- RAW: Formato de imagen de disco no estructurado. Es el formato básico creado y administrado a partir de herramientas como dd, parted, fdisk, mount, kpartx, etc. No está optimizado para su uso en entornos virtualizados pero es muy portable entre los hipervisores. Este es el formato por defecto que utiliza Glance al momento de crear respaldos de imágenes en línea.
- ISO: Volcado de sistema de archivos contenido en un CD/DVD.
- VDI: Formato soportado por VirtualBox.
- VMDK: Formato utilizado por VMWare en aplicaciones como VMWare Workstation.
- VHD: Utilizado general por aplicaciones como VMware, Xen, VirtualBox.
- Amazon AMI : formato utilizado por Amazon, *Amazon Machine Image*
- QCOW2: formato estándar de imagen usado por QEMU-KVM

2.5. DASHBOARD - Panel de Control

Dashboard bajo el proyecto Horizon es el servicio de Openstack encargado ofrecer una interfaz gráfica de control y administración de la nube de Openstack para el usuario bajo la modalidad de self-service. Su objetivo principal es el de facilitar la interacción de los usuarios y administradores con Openstack, evitando el uso de los APIs de cada proyecto para la creación y administración de las instancias.

2.5.1. Características

A diferencia del resto de proyectos, Horizon ha sido desarrollado utilizando el lenguaje para aplicaciones web denominado Django proporcionando una interfaz limpia y de rápido desarrollo que al mismo tiempo de ser ligera, permite un amplio rango de modificaciones.

FIGURA 18 : Panel de Control - Dashboard

Esta interfaz tiene las siguientes ventanas:

Para usuarios:

- Información relacionada a la cuota y al uso asignado a cada usuario así como los privilegios administrativos sobre sus instancias.
- Detalle de las instancias que operan de máquinas virtuales en la nube
- Administración de los discos lógicos; creación y eliminación, así como la conectividad con el almacenamiento de objetos.
- Acceso a las imágenes y la habilidad para administrarlas.

- Administración de las seguridades relacionadas a los usuarios y las instancias que han creado.

Para administradores:

En conjunto a las opciones que tiene para los usuarios, Dashboard también es una aplicación orientada al administrador de la nube permitiéndole:

- Creación del catálogo de flavors para los usuarios, asignando a cada uno de ellos los valores correspondientes a CPU, memoria RAM y espacio de disco.
- Creación de grupos virtuales de usuarios con proyectos relacionados (seguridad y privilegios)
- Acceder a una ventana informativa en la que se detallan los servicios en ejecución y las cuotas de infraestructura utilizadas.
- Detalle de la topología de la Red
- Subir o cargar imágenes a Glance para disponibilidad de los usuarios
- Agregar características extras a los flavors disponibles.
- Migración de las instancias hacia otros nodos

La arquitectura de Horizon ha sido implementada utilizando el mod_wsgi de Apache para aplicaciones desarrolladas en Python, esto quiere decir que el código en sí mismo es separado en diferentes módulos que pueden ser reutilizados para interactuar entre los diferentes APIs de los componentes de Openstack tal como se puede apreciar en el siguiente gráfico.

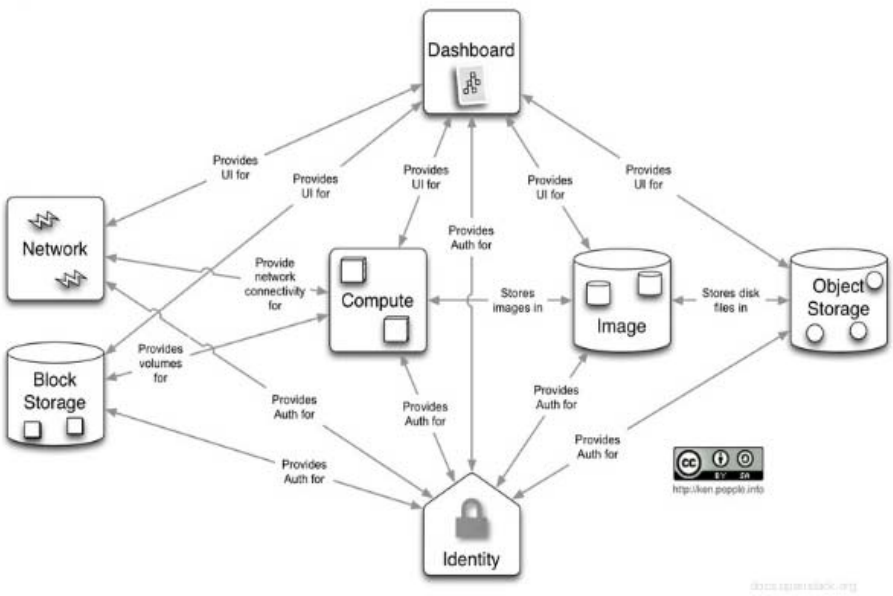


FIGURA 19 : Comunicación entre los componentes

Desde el punto de vista de la topología de red, el nodo donde se ejecuta el componente Horizon de Openstack debe ser accesible por los usuarios así como tener abierta la conectividad con los otros componentes de la nube por lo que es importante considerar estos factores al momento de su implementación.

2.6. CINDER - Bloques de almacenamiento

Los bloques de almacenamiento de volúmenes que utilizan las instancias son asignados por los APIs del proyecto Cinder. El sistema administra la creación y el relacionamiento de los bloques de almacenamiento y los servidores, así como también de administrar los diferentes tipos de volúmenes o discos lógicos.

El manejo de los bloques de almacenamiento era inicialmente parte del proyecto Nova o nova-volume por lo que hasta en las presentes versiones se puede hacer uso de este servicio de acuerdo a las necesidades del sistema.

2.6.1. Características:

- Creación, modificación y eliminación de los volúmenes.
- Respaldo de los volúmenes, incluyendo a los que se encuentran operativos o utilizados
- Consultas del estado de los volúmenes así como de sus metadatos.

2.6.2. Procesos

El proceso interno de consultas y asignación se detalla a continuación:

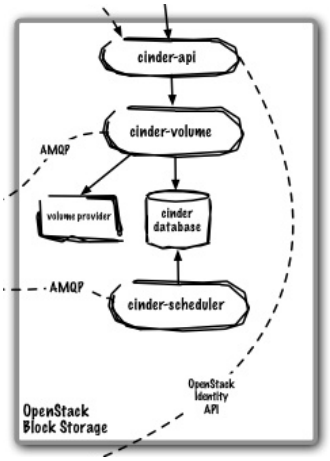


FIGURA 20 : Lógico CINDER

Cinder-api acepta las consultas y las direcciona hacia cinder-volume para proceder con la petición.

Cinder-volumen actúa siguiendo la petición mediante la lectura/escritura de la base de datos propia de Cinder determinando así el estado de la petición

e interactuando con otros procesos tales como cinder-schedule, a través de la cola de mensajes y directamente con el hardware y software de almacenamiento. Cinder-volumen puede interactuar con un variado número de proveedores de almacenamiento mediante su arquitectura de controladores genéricos y propios de cada proveedor. Entre estos controladores se encuentran los de IBM, SolidFire, Rados – Ceph, NetAPP, Windows Server 2012 iSCSI, HP, Nexenta, Zadara, Red Hat GlusterFS, EMC, Xen y Linux iSCSI.

- Cinder-sheduler es el servicio que se encarga de escoger al nodo con el bloque de almacenamiento óptimo para crear el volumen requerido sobre él

- Cinder-backup es el servicio que respalda la información del volumen usado en Swift.

Las implementaciones de Cinder utilizan la cola de mensajes para la comunicación de sus procesos, así como también para almacenar el estado de los volúmenes en su base de datos.

Como se puede apreciar en el siguiente gráfico, Cinder interactúa principalmente con Nova proveyendo de los volúmenes de almacenamiento a sus instancias.

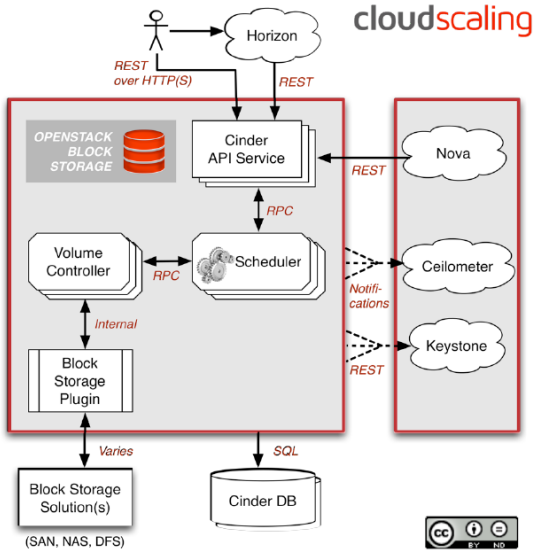


FIGURA 21 : Conceptual CINDER

2.7. SWIFT – Almacenamiento de Objetos

El almacenamiento de objetos está representado en el proyecto Switf de Openstack y se encarga de proveer de un sistema de almacenamiento de datos a gran escala y accesible a través de su API de conexión. A diferencia de los sistemas de almacenamiento tradicional, Swift es completamente distribuido permitiendo que hayan múltiples copias de un mismo objeto, agregándole a la nube alta disponibilidad y fiabilidad.

2.7.1. Características

Entre las características de Swift podemos enunciar las siguientes:

- Almacena y recuperas los archivos de los contenedores creados.
- Coloca y modifica los metadatos de los objetos
- Administra las versiones de los objetos

- Proporciona acceso a los objetos a través de consultas a través del protocolo http ya que cada uno tiene su propia dirección URL.

Los contenedores pueden ser pensados como la carpeta raíz dentro del sistema de almacenamiento que contendrá a los objetos, siendo estos últimos los archivos con la información relacionada con el usuario.

La arquitectura de Switf le permite una implementación altamente distribuida, lo que disminuye la posibilidad de fallas en la infraestructura, así como también le permite un crecimiento escalable y horizontal. (Swiftstack, 2013)

2.7.2. Procesos

A continuación una descripción de sus componentes:

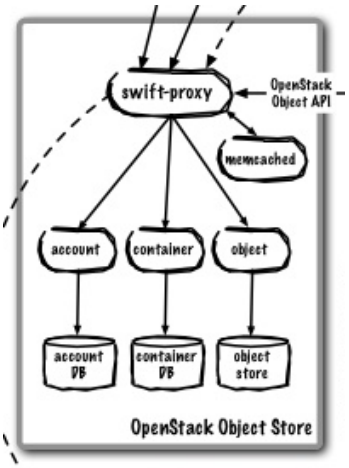


FIGURA 22 : Lógico SWIFT

o Swift-proxy: Administra las solicitudes enviada mediante el Openstack API o a través de consultas HTTP. Carga los archivos recibidos, modifica sus metadatos, así como sus contenedores. Adicionalmente utiliza una memoria

adicional tipo cache (memcache), siendo esta una unidad de almacenamiento de alta velocidad que mejora el desempeño del componente.

o Account : Administra las cuentas definidas por el servicio de almacenamiento de objetos

o Container: administra el mapeo de los contenedores o carpetas dentro del servicio de almacenamiento de objetos

o Object: Administra los objetos o archivos que se encuentran en los nodos de almacenamiento

También hay otros servicios que se encuentran activos dentro del proceso de Switf, un ejemplo de estos es el servicio de replicación que asegura la consistencia y disponibilidad de los objetos en el arreglo de discos duros.

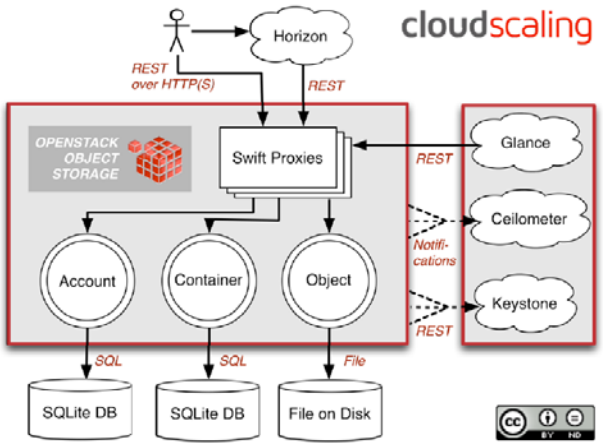


FIGURA 23 : Conceptual Swift

Como se puede apreciar en la gráfica anterior, el acceso los contenedores y objetos está controlada por el proceso de Keystone

2.7.3. Accesos a Swift

Swift puede ser utilizado no solo por los componentes de Openstack, a continuación enumeramos algunas aplicaciones que utilizan a Swift a través de su API de comunicación para controlar el almacenamiento de objetos:

- Cyberduck
- Cloudfuse
- jCloud SwiftBlobStore() de Java

Por tal motivo no solo es utilizada por Openstack, hay otras aplicaciones que implementan a Swift como su backend o programa base para desarrollar un propio producto, entre los más reconocidos están:

- Swiftstack
- Nasuni
- Panzura
- Riverbed
- FUSE

2.8. NEUTRON – Redes de datos

En Openstack se pueden implementar tres topologías de Redes de Datos, cada una con sus debilidades y fortalezas que iremos detallando más adelante. Enumerándolas de acuerdo a su complejidad están; primero la denominada como Flat DHCP Networking, luego VLAN manager y como la topología más recientes está al denominada como Software Defined Networking (SDN) o Red Definida por Programación, siendo esta última una

herramienta para la gestión de Redes donde al administrador de la Red y el operador de la Nube pueden definir los servicios de conexiones virtuales a través de programación. El componente SDN de Openstack se denomina Neutron

Conocido anteriormente como el proyecto Quantum, a Nova se lo puede definir como *"Network Connectivity as a Service"* ya que se encarga de la conectividad de las máquinas virtuales generadas por los otros proyectos de Openstack. Se cambió el nombre del proyecto porque la palabra Quantum ya se encontraba registrada anteriormente.

Inicialmente era parte del proyecto Nova bajo el componente *nova-network*, luego de la versión Grizzly pasó a ser desarrollado de forma autónoma ya que despertó gran interés en los programadores de la comunidad de Openstack así como en las compañías privadas, tales como Cisco y Juniper, que vieron en el proyecto la oportunidad de desarrollar controladores de red virtuales con mayor capacidad de procesamiento y mejor protección contra fallos.

Con SDN se puede implementar redes más complejas en un ambiente multi-contenedor más seguro ya que supera las debilidades de las topologías anteriores. Por ejemplo en la implementación bajo la topología Flat DHCP todas las instancias convivían bajo la misma subred IP independientemente de su contenedora, abriendo así una brecha de seguridad que posibilite la intromisión de una instancia sobre otra. Luego se implementó la topología con Redes virtuales o VLAN manager ya que separaba las redes de datos con el

identificados propio de la VLAN pero a su vez estaban limitadas por el número de Identificadores posibles, 4096, lo que la hacía imposible su implementación en redes más grandes, así mismo el usuario estaba limitado al rango de direcciones IPs propios de su subred. Sin embargo para vencer estos inconvenientes, se ha desarrollado la aplicación de Reglas Grupales de Seguridad para aislar las redes y sus aplicaciones logrando un entorno de operación más seguro.

2.8.1. Características

Entre las características de Neutron, podemos detallar las siguientes:

- Los usuarios puede crear sus propias redes y adjuntar sus máquinas virtuales a ellas.
- Tiene una arquitectura basada en plugin o complementos que permiten a los usuarios la utilización de equipos de red de tipo propietario y no-propietario. Lo que amplía el espectro de aplicación de las redes basadas en Neutron.
- Contiene extensiones que le permiten ofrecer servicios adicionales tales como el de balance de carga de la red.

Ampliando más a detalle las características de sus diferentes plugins o complementos, vale la pena destacar que los mismos pueden acomodarse a diferentes proveedores de hardware y software de red por lo que su aplicación puede variar ampliamente.

2.8.2. Procesos

En el siguiente gráfico vamos a detallar la operación Neutron

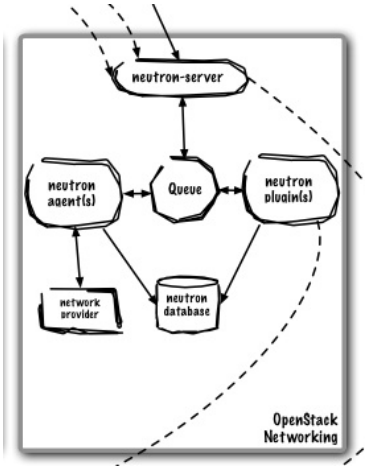


FIGURA 24 : Lógico NEUTRON

- Neutron-server acepta las solicitudes vía API y las direcciona al complemente adecuado a través de la cola de solicitudes.

- Los procesos Neutron-plugins y Neutron-agent son los que se encargan de crear las conexiones de red, asignar las subredes y las direcciones IP a las máquinas virtuales. Estas conexiones se generan de acuerdo al fabricante de los equipos que se utilizan en la red, por lo que hay plugins para los switchs físicos y virtuales de Cisco, otros provistos por NC para sus productos OpenFlow, lo mismo para Open vSwitch y Linux bridging. Todos ellos operando en Capa 3 y proveyendo de servicios tales como DHCP y otros servicios propios de cada fabricante.

- La mayoría de las instalaciones de Neutron hacen uso de la cola de mensajes para direccionar la información entre el proceso de Neutro-server y

los distintos plugins. Así mismo utilizan la base de datos para almacenar el estado de la red bajo la petición de los plugins de algún fabricante específico.

En el siguiente cuadro podemos observar que Neutron interactúa principalmente con Nova para proveer conectividad a las instancias generadas en Nova

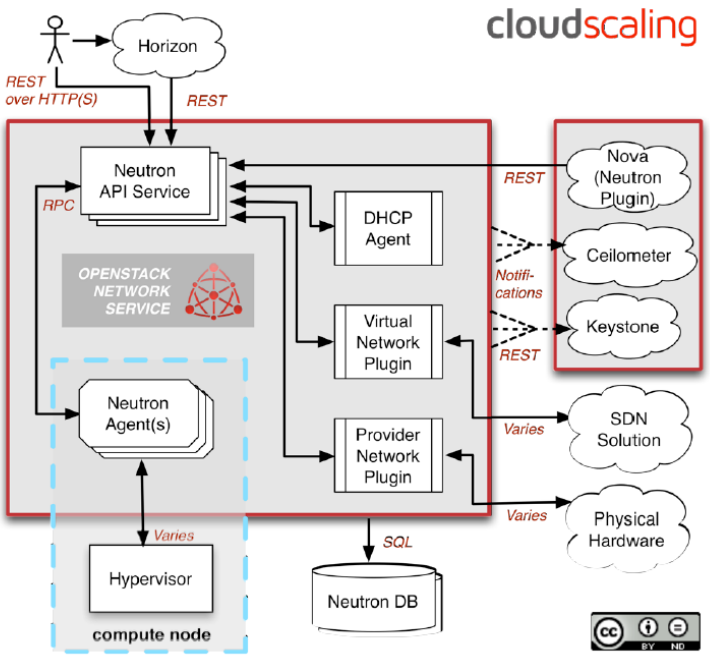


FIGURA 25 : Conceptual NEUTRON

El componente Nova instruye al Hypervisor a través de su módulo nova-compute para crear la Máquina Virtual, en este momento el agente de Nova se comunica con Neutron-server para recibir los atributos de la red para la máquina virtual. Luego el agente de Neutron informa a su plugin, en este caso VSwitch, para que configure la red virtual para la instancia y finalmente Neutron comunica las entradas de la tabla de ruteo solicitadas por vSwitch.

2.9. KVM – Hipervisor

Openstack utiliza varios hipervisores entre los que se incluyen; KVM, LXC., QEMU (Quick Emulator), UML, VMWare ESX/ESXi, Xen, PowerVM, Hyper-V.

Probablemente el factor más importante al momento de elegir el hipervisor que será utilizado se basa en el nivel de conocimientos previos y experiencia que se cuente sobre alguno de ellos. Otros aspectos a considerar son las características y bondades de cada uno y la documentación así como el soporte técnico que haya disponible.

Para nuestro caso, vimos que KVM es el hipervisor recomendado por la comunidad de Openstack y que ha sido mayormente utilizado en las diferentes implementaciones que hemos encontrado, sin embargo también había aplicaciones con Xen y VMware y Hiper-V aunque la documentación no era muy descriptiva o utilizaban una versión de Openstack anterior a Grizzly o Havana.

Vale la pena mencionar que en el documento *Openstack Configuration Reference* (Openstack, Hypervisor configuration basics, 2013) menciona que es posible ejecutar varios hipervisores en la misma implementación usando arreglos de nodos computacionales, sin embargo para nuestro caso

contaremos con un solo nodo Computacional por lo tanto solo podremos ejecutar un hipervisor.

2.9.1. Descripción

A continuación una descripción más detallada del hipervisor KVM

KVM son las iniciales de Kernel-based Virtual Machine y corresponde a la categoría de hipervisores Tipo 1 ya que se integra en un sistema operativo existente, en este caso Linux, aunque también se ha agregado a FreeBSD e Illumos. Desde sus inicios fue concebido como un proyecto de Código Abierto bajo el desarrollo de la empresa Qumranet que en el 2008 pasó a Red Hat para convertirse en la solución impulsada por la empresa desde la versión RHEL 6. Así mismo se ha convertido en el hipervisor de virtualización oficial del kernel de Linux desde la versión 2.6.20 facilitando su implementación en otras versiones y aprovechando de los aportes de importantes compañías (KVM, 2013). Actualmente es utilizado por SUSE y por Ubuntu en sus versiones oficiales.

Proporciona de virtualización en arquitecturas x86 que cuentan con extensiones Intel VT (Virtualization Techonology) o AMD-V (Virtualization) motivo por el cuál previo a su implementación, es necesario verificar si el equipo está capacitado para su ejecución mediante el siguiente comando:

 # Kvm-ok

Si la respuesta es:

INFO: /dev/kvm exists

KVM acceleration can be used

Significa que podemos continuar, caso contrario se debe verificar en el BIOS del equipo si es posible habilitar la virtualización del procesador.

Cuando se carga el módulo KVM, Linux se convierte en un hipervisor capaz de ejecutar las máquinas virtuales aisladas y las hospeda como procesos, por lo que cada una puede beneficiarse de todas las características del kernel de Linux, incluyendo las redes, seguridades y almacenamiento.

Así mismo todos los avances que se hagan al kernel de Linux como sistema Operativo, también se aplican a KVM, como por ejemplo (Tholeti, 2011):

- Las mejoras en el manejo de los núcleos de CPU y de la memoria RAM de un equipo por Linux, también serán trasladadas a las máquinas virtuales de KVM.
- Bajo KVM, la gestión de las máquinas virtuales se la mira como un proceso, por lo tanto si el kernel es mejorado en esa área, la gestión de KVM también se verá mejorada.
- Las imágenes de los discos de las máquinas virtuales se administran como cualquier archivo en Linux por lo tanto su almacenamiento puede realizarse bajo cualquier sistema soportado: NAS, iSCSI, SAN.
- KVM utiliza cualquier dispositivo que sea a su vez soportado por Linux.
- KVM también puede aprovecharse del modelo de seguridad de Linux, el programa Vagrant (http://www.vagrantup.com/) que crea “sandboxes” o

contenedores que encierran a las máquinas virtuales basados en los lineamientos de seguridad que aíslan las máquinas virtuales entre ellas. (Jackson, 2013)

Linux y KVM utilizan las librerías **libvirt** (http://libvirt.org/) y **libguestfs** (http://libguestfs.org/) para estandarizar sus comunicaciones a través de sus APIs, base para la gestión de las máquinas virtuales y de las imágenes.

Cabe mencionar que libvirt puede gestionar también a otros hipervisores como Xen, VMWare ESX, OpenVZ, Hyper-V, lo que lo colabora con la versatilidad de Openstack.

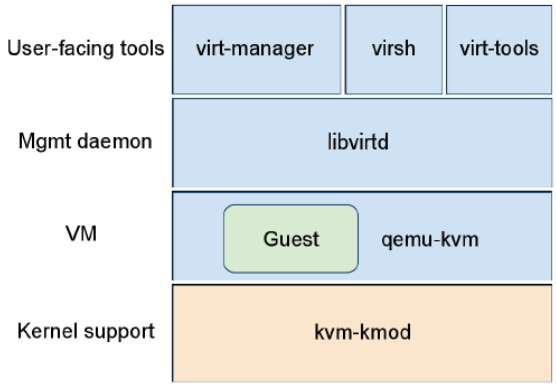


FIGURA 26 : Arquitectura de KVM

(IBM, IBM DeveloperWorks, 2010)

Con respecto al manejo de imágenes, el formato nativo de KVM es QCOW2, el cual permite la realización de snapshots, compresión y cifrado de las máquinas virtuales en ejecución, esta características es utilizada por el componente Glance de Openstack para generar respaldos en línea. De esta manera KVM se relaciona con el proyecto de código abierto QEMU, un

sistema de monitoreo de máquinas virtuales, ya que QCOW2 es un formato de archivo para imágenes de discos propio de QEMU con características avanzadas como la expansión dinámica, copy-on-write, snapshots, cifrado y compresión entre otros.

Así mismo QEMU se utiliza para proporcionar los dispositivos de entrada y salida que las máquinas virtuales utilizan.

Los hipervisores y las tecnologías de virtualización han ido creciendo y expandiendo su influencia en varios campos de las Tecnologías de la Información. Uno de las evoluciones lógicas ha sido su aplicación como elemento básico en el entorno de las nubes computacionales. Varios proyectos utilizando KVM como hipervisor, entre ellos están Openstack, CloudStack de Apache y OpenNebula.

2.9.2. Componentes

Los paquetes básicos de KVM se instalan mediante la siguiente instrucción:

 # sudo apt-get install qemu-kvm libvirt-bin bridge-utils

siendo:

- qemu-kvm: Sistema KVM. Incluye el módulo del kernel
- libvirt-bin : kit de herramientas programadas en C que se utilizan para la interacción con el hipervisor
- bridge-utils: Kit de herramientas que se utilizan para la configuración de los puentes Ethernet entre las máquinas virtuales y el hipervisor.

KVM cuenta con más instrucciones de tal manera que controle todo el proceso de creación y administración de las máquinas virtuales y sus comunicaciones. Sin embargo los proyectos de Openstack controlan el resto de las características de las instancias creadas en la nube.

2.9.3. Proceso de creación de las instancias

Las instancias se ejecutan en el nodo Computacional bajo la dirección del componente Nova y con el soporte del resto de componentes, cada uno de ellos con su correspondiente responsabilidad en el proceso. A continuación describimos los pasos seguidos en el ciclo de puesta en alta de una máquina virtual (Openstack, Compute Node, 2014):

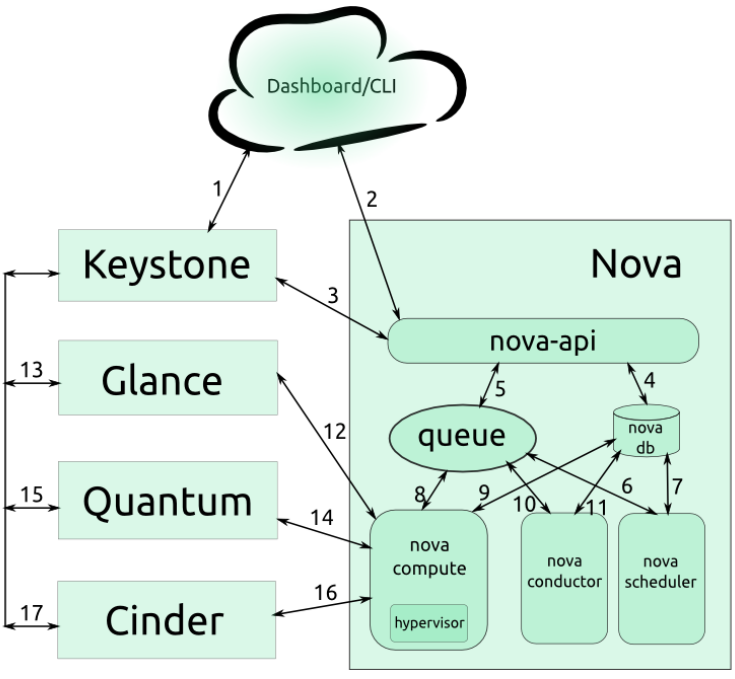


FIGURA 27 : Generación de Máquinas Virtuales

1. El usuario autentica sus credenciales con Keystone vía RESTful API y le retorna un auth_token relacionado a la solicitud y que será utilizado por lo demás componentes

2. Horizon convierte la petición de la instancia en una orden de ejecución que se envía a nova-api

3. Nova-api recibe la solicitud y valida el auth_token y los permisos de acceso con Keystone, quien a su vez responde con la confirmación de los permisos y roles asignados

4. Nova-api interactúa con nova-database creando la base relacionado a la petición de instancia.

5. Nova-api envía una consulta a nova-scheduler para proceder con la petición.

6. Nova-scheduler recoge la petición de la cola

7. Nova-scheduler interactúa con nova-database para encontrar al anfitrión adecuado utilizando el procedimiento de ponderación de filtros y pesos. La respuesta incluye el identificador del anfitrión asignado (host_id) y de inmediato nova-scheduler envía la solicitud a nova-compute para que ejecute la instancia en dicho anfitrión.

8. Nova-compute recupera la petición de la cola de mensajes

9. Nova-compute envía la solicitud a nova-conductor para que busque la información tal como el identificador del anfitrión (host_id) y las características físicas de la instancia (flavor)

10. Nova-conductor recupera la solicitud de la cola de mensajes

11. Nova-conductor interactúa con nova-database para recuperar la información relacionada a la instancia

12. Nova-compute envía la autorización auth_token a glance-api y solicita la URI de la imagen de acuerdo al su identificador.

13. Glance-api valida el auth_toekn con Keystone

Mientras tanto nova-compute recibe los metadatos de la imagen y envía una solicitud al Neutron-api para que se asigne la información de red a la instancia

14. Neutron-server valida el auth_token con Keystone y remite la información de red solicitada por nova-compute

15. Nova-compute remite el auth_token a Cinder solicitando la asignación de un disco virtual a la instancia.

16. Cinder-api valida el auth_token con Keystone mientras que nova-compute recupera la información del bloque de almacenamiento.

17. Nova-compute genera los datos para iniciar el driver del hipervisor así como también su puesta en marcha de la instancia en él, a través de libvirt o el API.

Red Hat realizó una presentación en el REDHAT SUMMIT en junio del 2013 en el que se mostró el siguiente gráfico del proceso de creación de una instancia más sencillo de comprender:

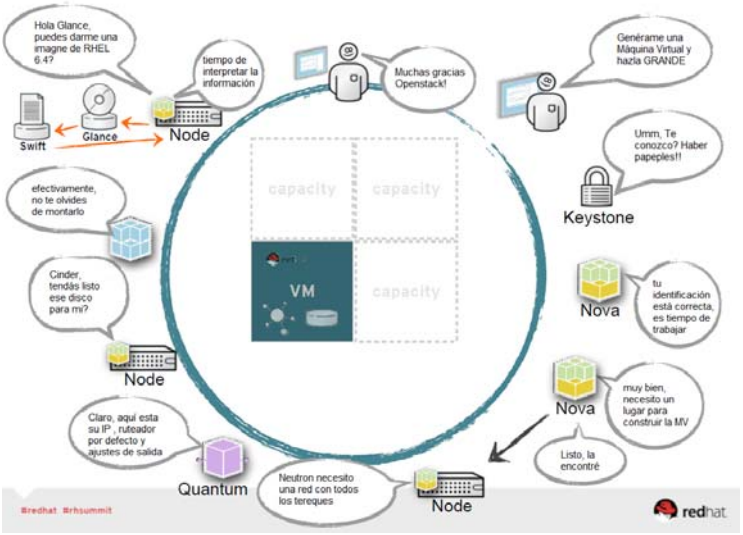


FIGURA 28 : Ciclo de creación de una Máquina Virtual

3. CAPITULO III: IMPLEMENTACIÓN

3.1. Consideraciones para la instalación

Openstack ha sido bien recibido por la comunidad de desarrolladores a nivel mundial, su crecimiento ha permitido que muchos programadores particulares y empresas involucradas se junten y desarrollen métodos y programas de instalación menos complejos y más rápidos. Consideremos que en sus versiones iniciales y previas al lanzamiento de Horizon, la puesta en marcha de la plataforma Openstack incluía la instalación paso a paso de los diferentes componentes así como de los diferentes paquetes asociados.

Como le mencionamos anteriormente, la arquitectura de Openstack admite un crecimiento horizontal, lo que permite que todos los servicios sean distribuidos en una infraestructura computacional física reducida y con la

ventaja de poder incrementarla conforme sea requerida por los usuarios y bajo el punto de vista técnico de los administradores que evalúan el desempeño de la nube instalada.

Desde el punto de vista operacional, al momento de planificar la implementación de Openstack, hemos considerado los siguientes aspectos que a su vez pueden ser trasladados a implementaciones de mayor capacidad;

- Número de Usuarios simultáneos y ocasionales
- Requisitos de Máquinas Virtuales
- Costes de implementación
- Elección del proveedor de equipos
- Planificación de la infraestructura a largo plazo
- Complejidad y control de las actualizaciones

Los aspectos mencionados anteriormente nos permiten visualizar la flexibilidad de Openstack ya que de cierta manera puede adaptarse para cumplir con los diferentes requerimientos de la implementación.

3.2. Modelo de Implementación

Otro factor a considerar es el modelo de implementación por el que se elegirá, en el caso de nuestro análisis encontramos dos tipos de opciones:

- Multi-Nodo
- Multi-Nodo con Alta disponibilidad

Para el caso de nuestro proyecto optaremos por la implementación de tipo Multi-Nodo, comprendido por los nodos de Computación o *Compute*, el de Red o Networking y el de Control o *Controller*.

3.2.1. Multi modo con alta disponibilidad

Ya en entornos de producción donde el concepto de Alta Disponibilidad o High Availability (HA) es un requisito fundamental, se puede implementar Openstack con al menos dos o tres nodos de Control de tal manera que se reduzca los posibles puntos de fallo que afecten al desempeño de la nube. Este tipo de implementaciones son posibles gracias a que los servicios de Openstack están comunicados a través de los RESTful APIs mediante protocolo HTTP y en los mensajes basados en AMQP usando el protocolo RPC. Además de las ventajas propia de Openstack, la implementación HA incluye el uso de otras herramientas de código abierto disponibles, por ejemplo la solución Pacemaker (ClusterLabs, 2013) que es un administrador de recursos escalables y agrupados de alta disponibilidad, tomando la información desde el portal Web de Pacemaker podemos anotar las siguientes características sobre el software:

- Servicio de detección y recuperación de un equipo o aplicación que haya fallado.
- Suporta prácticamente cualquier configuración de redundancia.
- Cuenta con un sistema de estrategias configurables en caso de que muchas máquinas fallen a la vez.
- Controla la ejecución de aplicaciones en el mismo equipo

- Controla las aplicaciones que requieren ser ejecutadas en varios equipos simultáneamente.

Más detalles de la aplicación se pueden encontrar en http://clusterlabs.org/

Otra aplicación que se utiliza para las implementaciones HA es Galera de Codership, se trata de un programa que permite la clusterización o arreglo de las bases de datos generadas en MySQL, así mismo tomando la información desde el portal web del desarrollador podemos anotar las siguientes características que lo hacen útil para Openstack:

- Permite la replicación sincrónica de las bases de datos
- Posee una topología multi-master activa
- Puede leer y escribir en cualquier nodo de arreglo de bases de datos
- Control automático de los miembros, cualquier nodo que falle es retirado del arreglo.
- Inclusión de nodos relacionados automáticamente.
- Conexiones directas basadas en MySQL nativo.

Más información al respecto se encuentra en http://codership.com/

Una implementación redundante o HA de Openstack apoyada en sus APIs utiliza una combinación de direcciones IP virtuales administradas por el componente Neutron, más la aplicación de los programas anteriormente descritos donde la replicación de las bases de datos MySQL están a cargo de Galera, la administración de las colas de mensajes se realiza mediante la

aplicación de las capacidades de cluster que tiene el mismo RabittMQ y donde la orquestación del arreglo de servidores está a cargo de Pacemaker. Mirantis y PistonCloud, empresas especializadas en implementaciones comerciales de Openstack con redundancia, han definido la arquitectura de alta disponibilidad bajo el siguiente diagrama que incluye el componente *HAProxy* que en este caso es Pacemaker.

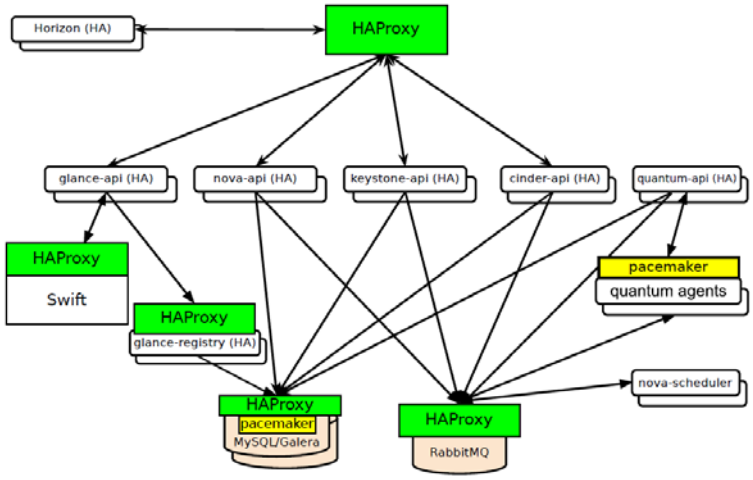


FIGURA 29 : Arquitectura de Nodos HA

(Mirantis, 2013)

Como se puede apreciar en la gráfica, este tipo de implementación se orienta más a Centro de Datos que cuentan con infraestructura suficiente para replicar los componentes de la Openstack en varios equipos.

3.2.2. Multi modo sin alta disponibilidad

Este tipo de arquitectura resulta ideal para pequeñas y mediana empresas que quieren dar sus primeros pasos en el mundo “Cloud”. El sistema multi-nodo pero sin Alta Disponibilidad se refiere a la implementación en la que cada

nodo será único en su función y en los componentes que alberga, sin embargo permitirá un crecimiento uniforme tanto en almacenamiento como en procesamiento ya que se podrán agregar horizontalmente más equipos que operen como Nodos Computacionales con el objetivo de incrementar el poder de procesamiento y a su vez el número de máquinas virtuales que operan de manera simultánea.

El siguiente diagrama de la topología de red muestra este tipo de implementación:

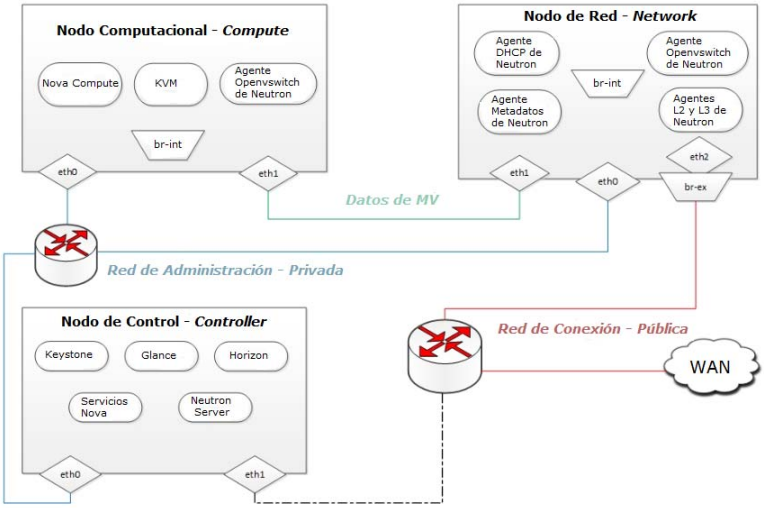


FIGURA 30 : implementación multi nodo sin redundancia

3.2.2.1. Descripción:

Cada nodo representa a un equipo, hemos dividido en tres sistemas independientes para comprender la escalabilidad de Openstack tanto en la capacidad de procesamiento y almacenamiento así como en la fiabilidad que implica tener un Nodo de Control que puede duplicarse a modo de respaldo.

Nodo Computacional: Este nodo alberga a las máquinas virtuales por lo tanto debe instalarse los siguientes procesos:

- Nova-compute
- Neutron Agente (plugin) de Openvswitch
- Hipervisor KVM

Aquí aprovechamos para ampliar el concepto de que su crecimiento es horizontal ya que al momento de configurar los sus componentes, en particular nova.conf, se determina la dirección IP del nodo de Control que contiene la cola de mensajería de tal manera que su adhesión a la nube sea directa y sin dependencia otro nodo similar.

Por otro lado el proceso Nova-compute también regula las políticas de seguridad de las instancias

Nodo de Red: Contendrá todos los procesos referentes a Neutron y los que le permitan conectarse al Nodo de Control. Este nodo se encarga de manejar la conexión de las máquinas virtuales hacia el exterior generando rutas independientes e implementan políticas de seguridad a través de su firewall interno.

Nodo de Control: Es el encargado de orquestar la operación de la nube, contiene los componentes de autenticación, imágenes y el dashboard de acceso, así como también los componentes de administración del nodo computacional Nova-api. Aquí se instala la base de datos MySQL y el sistema de control de la cola de mensajería RabbitMQ.

Red de Datos: La red que hemos elegido para el diseño de la nube se basa en tres segmentos:

- *Red de Administración:* Este segmento comunica a los tres nodos permitiendo que los API puedan intercambiar las instrucciones de control y ejecución de las máquinas virtuales. Así mismo cabe mencionar que es privado por lo tanto sus direcciones IP no se encuentran publicadas y para nuestro diseño serán de tipo estáticas.
- *Red de Conexión:* Este segmento corresponde a la conexión de los componentes del nodo Computacional hacia una red exterior, que puede ser el internet, brindando un acceso al Dashboard más amplio a los usuarios. Así mismo permite que el nodo de Red acceda al internet y direccione el tráfico generado por las máquinas virtuales albergadas en el nodo Computacional.
- *Red de Datos:* esta red comunicará al nodo Computacional y al nodo de Red canalizando el tráfico generado por las máquinas virtuales hacia el exterior pasando a través de sus firewall de protección y control.

3.2.3. Infraestructura inicial.

Al momento de determinar la infraestructura sobre la que correrá una nube, debemos considerar las diferentes opciones que hay disponibles tanto en el área de hardware así como de software.

Siendo el hardware la base física sobre la que se alojará la nube, hay que cuantificar el poder de procesamiento así como la cantidad de memoria RAM con la que se debe iniciar pero considerando un crecimiento a futuro.

Poder de Procesamiento –CPU

Depende del número de máquinas virtuales que se planea implementar en el ambiente de la nube y del valor de CPU asignado a cada máquina virtual

Memoria

Depende de la cantidad de RAM asignada a cada máquina virtual así como al nodo Computacional

Almacenamiento

Depende de la asignación de discos físicos a cada máquina virtual. Openstack tiene dos tipos de discos, el de resguardo permanente y los de resguardo temporal, llamados efímeros (ephimeral) porque la información se elimina una vez que se borra la máquina virtual a la que está ligada.

Redes de Datos

Depende la arquitectura de la nube que se planea implementar, así como del consumo de ancho de banda de cada máquina virtual y del sistema de almacenamiento en la red.

Para determinar la escalabilidad de la nube y cómo mejorar los niveles de rendimiento se requiere manejar todas las variables anteriormente mencionadas, sin embargo se puede partir considerando el número de máquinas virtuales que se espera ejecutar simultáneamente y la cantidad de espacio de almacenamiento requerido de acuerdo a los diferentes *flavors.*

En el Internet hay varios portales que detallan más sobre las consideraciones iniciales, sin embargo nos permitimos recomendar:

https://www.mirantis.com/openstack-services/bom-calculator/

ya que sugiere los equipos en base a un cálculo del número de máquinas virtuales que se ejecutarán simultáneamente.

4. CAPÍTULO IV : INSTALACIÓN DE OPENSTACK

Para realizar la implementación de la plataforma Openstack, hemos determinado el siguiente equipo:

4.1. Plataforma

4.1.1. Detalle Físico

Servidor Dell Power Edge 2950 para montar en rack de 2 U con las siguientes características:

- Dos procesadores Intel Xeon ® de 4 núcleos de 2.66 GHz y 64 bits.
- 32 GB de RAM DDR-2 de 667 MHz para servidor
- 2 Puertos de red 1 Gbps.
- 4 Discos duros de 1 TB.
- 2 Fuentes de poder redundantes
- 1 Lector de DVD
- Controlador RAID de 6 unidades SATA
- Tarjeta iLO dedicada para administración

4.1.2. Descripción Lógica

Los disco duros están dispuestos en dos arreglos; el primero para el sistema operativo configurados en espejo. Los otros dos discos en mono nativo.

Dejando una capacidad total de 3 TB disponibles para el proyecto.

Para la implementación, hemos utilizado el sistema operativo Linux en la versión Ubuntu Server 12.04.03 LTS para procesadores de 64 bits.

La versión de Openstack puede ser instalada en un arreglo de nodos de procesamiento bajo las arquitecturas antes mencionadas o instalando en un solo nodo que a su vez alberga varios equipos virtualizados.

En nuestra investigación, encontramos que las plataformas de virtualización más utilizadas son Vagrant y VMware. Hay documentación disponible en el Internet con muchos detalles sobre los pasos a seguir para instalar Openstack en este tipo de ambientes virtualizados. Sin embargo no es recomendado para entornos de producción por el alto consumo de recursos inmersos solamente en poner en marcha la aplicación, sin embargo la ventaja fundamental es que permite realizar una implementación con más nodos de los que podemos utilizar en ambientes de hardware exclusivamente.

Para la elección de la implementación, nosotros consideramos que la realización de la implementación en ambientes virtuales puede reducir la capacidad que el equipo afectando a su vez el desempeño de las posibles aplicaciones que los estudiantes pueden requerir.

Por otro lado, la implementación de Openstack en varios nodos, mínimo tres como lo mencionamos anteriormente; nodo Controlador, nodo Computacional y nodo de Red resulta muy exigente para el hardware en el que se instala e incluye entre otras cosas a equipos de red, tales como Switchs de capa L2/L3 con soporte para VLANs.

En el portal de Openstack hay varias recetas de instalación, así como otras en el internet. Para nuestro proyecto hemos optado por la instalación de la plataforma en dos equipos; el primer equipo que incluya al componente de Red y de Almacenamiento y un segundo equipo que contenga al componente computacional donde se albergará a las instancias.

4.1.3. Máquinas Virtuales

De acuerdo al hardware propuesto vamos a calcular el número de máquinas virtuales que podremos ejecutar de manera simultánea en el nodo Computacional. Para ello utilizamos la siguiente fórmula tomada del libro "Openstack *Operations Guide"* (Tom Fifield, Diane Fleming, 2013) :

$$\#n\acute{u}cleos = \frac{\#\ VM\ x\ Vel\ de\ VM}{Vel\ \ Procesador}$$

Donde:

- #núcleos corresponde a la cantidad de núcleos del Procesador
- # VM corresponde a las Máquinas Virtuales que operaran simultáneamente

- Vel de VM corresponde a la velocidad en GHz que asignaremos a cada instancia, en nuestro caso este valor será de 1,5 GHz

Resolviendo la ecuación:

$$\#VM = \frac{4\ x\ 2.66}{1.5}\ ;$$

$$\#VM = 7.01$$

Por lo tanto el número de instancias que se podrán ejecutar simultáneamente es de siete máquinas virtuales, considerando que no exista la posibilidad de sobre utilizar los recursos existentes. Sin embargo si se activa la opción de Hyper-threading o de Múltiples hilos en el servidor podemos incrementar el número de instancias en 1.3 veces el cálculo inicial obteniendo nueve instancias que se pueden ejecutar conjuntamente.

4.2. Procedimiento de instalación:

No importa si vamos a realizar la instalación en uno o más equipos dentro de la plataforma, todos ellos deben ser configurados de manera común siguiendo el siguiente procedimiento:

- Configurar direcciones de red estáticas en las interfaces disponibles
- Modificar el archivo /etc/hostname con el nombre de cada equipo
- Modificar el archivo /rtc/host con los nombres de los equipos que participan en la plataforma y la dirección IP correspondiente, este archivo será el mismo para todos los nodos

- Instalar el servicio NTP y definir al equipo será encargado de ser el servidor principal o de control
- Crear la tabla de contraseñas necesarias para la instalación de los diferentes servicios, esta debe tener un formato prestablecido para su fácil recordación:

TABLA 5 : LISTADO DE CLAVES

PROCESO	CLAVE
Admin	adminfie
keystone	keystonefie
nova	novafie
mysql	mysqlfie
rabbitmq	rabbitfie

Con respecto a la red de datos, hemos configurado el siguiente esquema de red:

- Red interna: utilizada por las máquinas virtuales: 10.11.12.0/24
- IP Flotantes: 192.168.0.224/27

4.2.1. Preparación inicial del software

- Actualizar el paquete base, actualizar el sistema operativo conforme a la distribución.
- Instalar el paquete python-software-properties
- Instalar el repositorio cloud-archive:havana de Ubuntu
- Reiniciar para que los cambios tengan efecto.

Para la instalación de los paquetes, por favor refiérase a nuestros anexos. Sin embargo a continuación encuentre un resumen del procedimiento seguido:

- Instalar MySql server
  - apt-get install mysql-server
- debemos realizar instalación segura para eliminar los usuarios anónimos y el acceso remoto
- Instalar python-mysqldb
  - apt-get install python-mysqldb
- Instalar el Messaging server Rabbit
  - apt-get install rabbitmq-server
- Debemos cambiar el password por defecto del usuario guest
- Instalar el servicio de Indentidad Keystone
  - apt-get install keystone
- Instalar el servicio Glance
  - apt-get install glance
- Instalar el cliente Glance
  - apt-get install python-glanceclient
- Instalar los servicios de nova
  - apt-get install nova-novncproxy novnc nova-api nova-ajax-console-proxy nova-cert nova-conductor nova-consoleauth nova-doc nova-scheduler python-novaclient
- Instalar el Dashboard

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

- Se debe remover el tema Ubuntu del entorno del sitio Dashboard
  - apt-get remove --purge openstack-dashboard-ubuntu-theme

- Instalación de Cinder
  - apt-get install cinder-api cinder-scheduler

En el anexo I se detallan los pasos seguidos para la instalación de todos los componentes de Openstack, sin embargo debido a que es muy posible cometer algunos errores de sintaxis durante el procedimiento, recomendamos utilizar paquetes instaladores previamente configurados tales como el proyecto FUEL patrocinado por la empresa Mirantis, disponible en http://www.mirantis.com que instala la versión más reciente disponible de Openstack utilizando la misma arquitectura y plataforma anteriormente descrita.

5. CAPÍTULO V: ACTIVIDADES DE LABORATORIO

Una vez instalada la plataforma, podemos realizar las siguientes actividades con el objetivo de familiarizarse con los comandos y el Panel de control Dashboard.

5.1. Actividad 1: Keystone 1

Objetivo: En este ejercicio vamos a configurar un equipo para que acceda a la implementación de Openstack

Requisitos: Openstack instalado y operativo, una cuenta de Administrador – Admin con acceso al componente Horizon

Instrucciones:

Usando el cliente de Línea de Comandos – CLI

Se asume que el estudiante está familiarizado con el uso de Terminal de Linux o cualquier otro cliente CLI

Instalar las variables de entorno

Antes de realizar cualquier instrucción a través del uso de los clientes API de Openstack, es necesario configurar las variables de entorno en el equipo. Las mismas que son necesarias para autenticar la sesión en el Nodo Controlador.

Los estudiantes podrán descargar el archivo openrc.sh desde el Dashboard de Horizon siguiendo la siguiente dirección

A continuación vamos a crear usuarios utilizando los comandos en línea de keystone, por ejemplo:

Enlistamos los usuarios: keystone user-list

Y confirmamos con el detalle de usuarios mostrado en el Panel de Control.

De manera similar realizamos con los siguientes comandos:

- Keystone user-create
  - Generar un usuario se lo realiza de la siguiente forma:

- $ keystone user-create --name <nombre> --tenant-id <tenant-id> --pass <contraseña> --email <correo>
- Verifique lo que ha generado en : Admin -> Proyectos -> Usuarios

5.2. Actividad 2: Administración y Cuentas

Objetivo: En esta práctica, vamos a revisar Keystone, la CLI y Administración de usuarios.

Procedimiento:

Configuración del sistema: Trabajo en OpenStack deployment. Una cuenta de administrador con acceso a horizon deployment.

Enlace a Horizon:

http://192.168.0.190/

Su usuario y password

Lista de Proyectos

$ keystone tenant-list

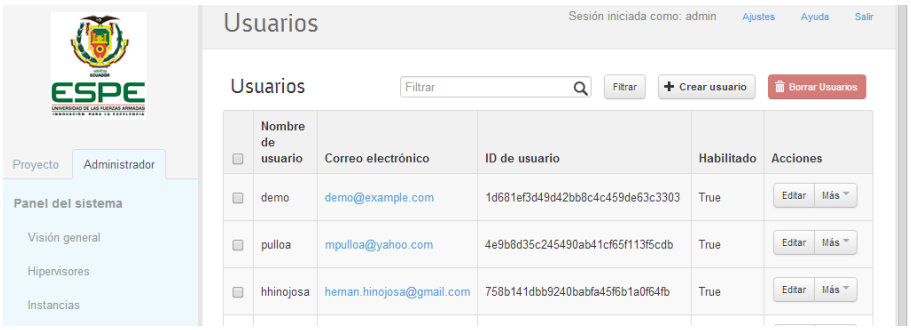
Tenga en cuenta que el Comando utiliza "tenant" en lugar de "proyecto".

Lista de usuarios: $ keystone user-list:

Agregar nuevo usuario al proyecto: $ keystone user-create

Vía Dashboard

- Admin -> Usuarios -> Crear usuario



Ahora probamos a través de línea de comandos

- $ Keystone user-create --name<user-name> [--tenant-id<tenant-id>] [--pass<password>] [--email <email>] [--enable<true|false>]

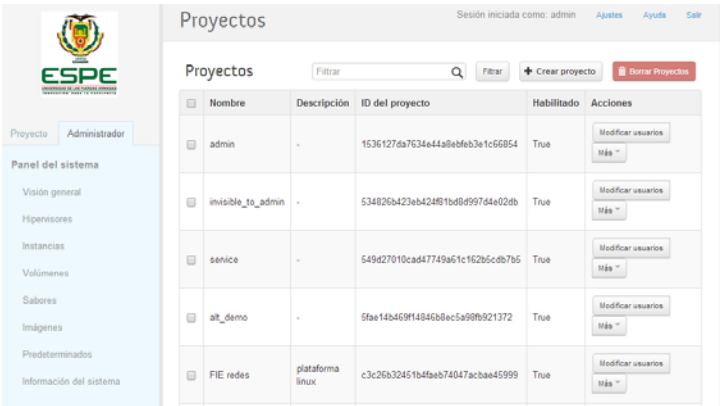
Agregar un usuario existente al proyecto Test

Vía Dashboard

Admin - > Proyectos - > Modificar usuarios

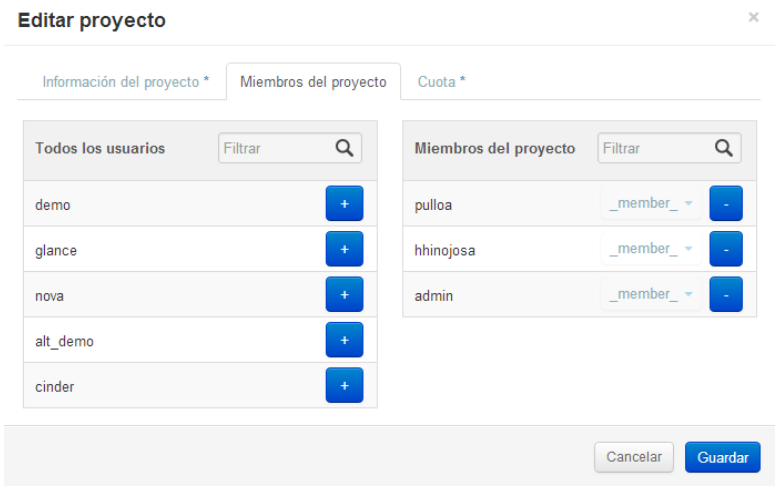
Ahora a través de línea de comandos

- $ Keystone user- role- add --user-id<user-id> --role-id <role-id> [--tenant-id <tenant-id>]



Vaya a Admin -> Proyectos -> Proyecto FIE redes y compruebe que el

usuario está en el proyecto apropiado con el role que le corresponde

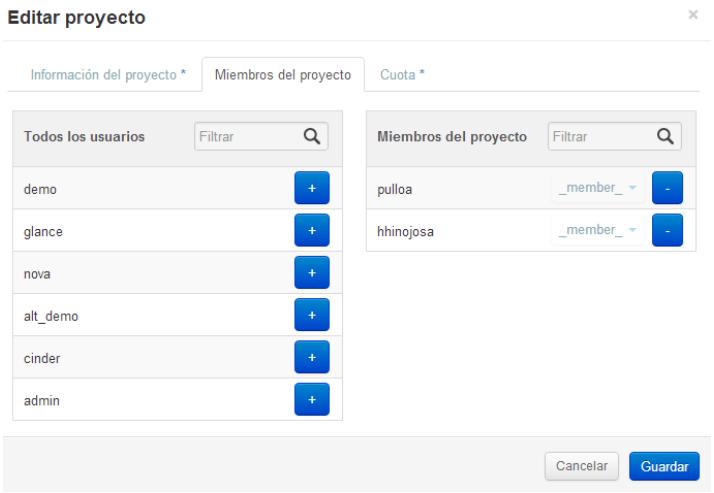


Inicie la sesión como el nuevo usuario y navegue a través del Dashboard

Confirme que el nuevo usuario tiene derechos de miembro, y no a los derechos de administrador.

Elimine el usuario de Project

Vía Dashboard eliminamos el usuario admin



A través de línea de comandos

$ Keystone user-role-remove --user-id <user-id> --role-id<role-id> [--tenant-id<tenant-id>]

TABLA 6 : COMANDOS EN KEYSTONE

Listar los comandos Keystone disponibles	**$ keystone help**
Comando para administrar Proyectos	
Crear un nuevo tenant	tenant-create
Borrar un tenant	tenant-delete
mostrar detalles de un tenant	tenant-get
Lista de todos los tenants	tenant-list
Actualizar nombre, descripción y estatus	tenant-update
Comando para administrar Roles	
Crear un nuevo rol	role-create
Borrar un rol	role-delete
Mostrar detalles de role	role-get
Listar todos los roles	role-list
Comandos para administrar Usuario	
Crear nuevo usuario	user-create
Borrar usuario	user-delete
Motrar detalles de usuario	user-get
Lista de usuarios	user-list
Actualizar password de usuario	user-password-update
añadir rol a usuario	user-role-add
Listar roles asignados a un usuario	user-role-list
Remover role a un usuario	user-role-remove
Actualizar nombre de usuario, email y estatus	user-update

Ejemplos

Lista de Proyectos Disponibles

$ keystone tenant-list

Mostrar Información especifica de un proyecto

$ keystone tenant-get <tenant_id>

Ejemplo

$ keystone tenant-get fa3700a42da8446f9d989dddbb2fbe0b

Crear un Nuevo Proyecto

$ keystone tenant-create --name <tenant-name>

Opciones

[--description <tenant-description>]

[--enabled <true|false>]

Actualizar Project Metadata

$ keystone tenant-update <tenant-id>

Opciones

[--name <tenant_name>]

[--description <tenant-description>]

[--enabled <true|false>]

Ejemplo

$ keystone tenant-update fa3700a42da8446f9d989dddbb2fbe0b

--name Hhinojosa --description Prueba --enabled true

Enlistar los usuarios registrados

$ keystone user-list

Crear un Nuevo usuario de autenticación

$ keystone user-create --name<user-name>

Opciones

[--tenant-id <tenant-id>]

[--pass <password>] [--email <email>]

[--enabled <true|false>]

Ejemplo

$keystone user-create --name pulloa --pass patricio --email prueba@ejemplo.com --enabled true

Actualizar un usuario

$ keystone user-update <user-id>

Opciones

[--name <user-name>]

[--email <email>]

[--enabled <true|false>]

Ejemplo

$keystone user-update --name Hhinojosa --description Prueba -- enabled false 2a7081224a2a4aeab83bd6f188e1e753

Eliminar un usuario

$ keystone user-delete <user-id>

Ejemplo

$ keystone user-delete 2a7081224a2a4aeab83bd6f188e1e753

5.3. Actividad 3 : Instancias y Migraciones

Objetivo: En esta práctica, vamos a crear las instancias desde el dashboard y la interfaz de línea de comandos (CLI). Vamos a migrar las máquinas virtuales a diferentes hosts.

Requisitos:

- Ambiente de trabajo OpenStack.
- Una cuenta de administrador con acceso a Horizon.
- Un computador configurado para acceder al ambiente de trabajo de OpenStack.

Enlace a Dashboards :

http://192.168.0.190/

Procedimiento:

Crear y migrar las instancias

En esta parte, vamos a crear instancias y migrar a diferentes hosts.

Crear un keypair

1 . Haga clic en Acceso y Seguridad

2 . Haga clic en el botón Crear keypair en la sección keypair

3 . Rellene el campo keypair name

4 . Lea la descripción del keypair para conocer su propósito

5 . Haga clic en Crear Keypair.. Tenga en cuenta que el keypair privada se ha descargado en su computadora.

Cambie los privilegios de ese archivo a 600 (por ejemplo chmod 600 keypair.pem)

Crear una instancia utilizando el Dashboard (Método # 1)

6 . Haga clic en Instancias en el panel Administrador

7 . Haga clic en el botón arranque de Instancia

8 . Seleccione la imagen como Instancia de fuente

9 . Seleccione Image Ubuntu 12.04 en el menú desplegable de imagen

10 . Escriba un nombre en el campo Nombre de instancia

11 . Elija un flavor (m1.small) en el menú desplegable

12 . En instancias de campo, tipo 1

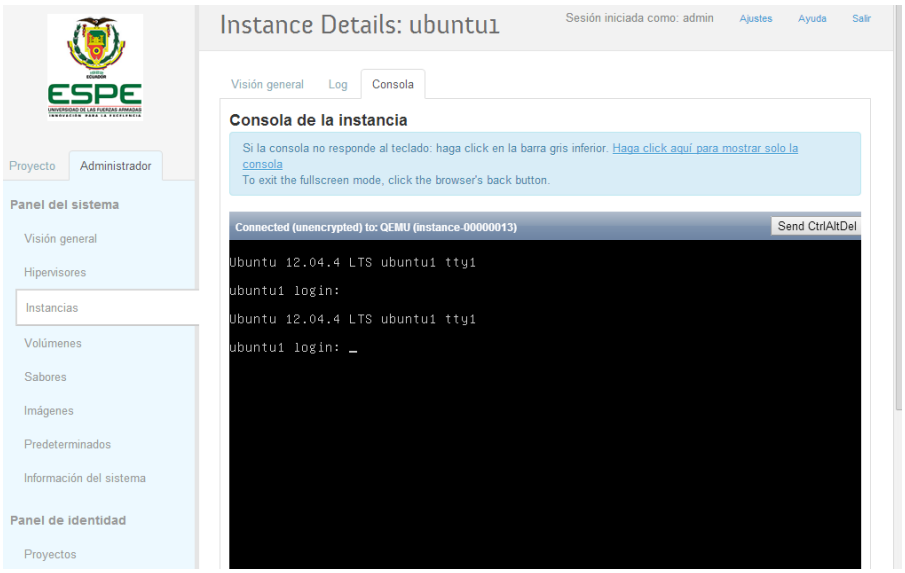
13 . Haga clic en el enlace de acceso y seguridad en la parte superior de la ventana

14 . En la ventana, seleccione la keypair generadas en el paso 5.

15 . Haga clic en Iniciar. La instancia se generó y el estado pasará de Build a Activo.

16 . Seleccione la casilla junto a la nueva instancia y haga clic en el botón rojo Finalizar las veces.

17 .Haga clic en Finalizar instancias cuando la casilla de confirmación aparezca. Vamos a crear otra instancia en la siguiente sección.



Crear una instancia utilizando el Dashboard (Método # 2)

18 . Haga clic en Imágenes y Snapshots

19 . Haga clic en Iniciar al lado de la imagen Ubuntu 12.04 en la sección Imágenes

20 . Seleccione imagen como Instancia de fuente

21 . Seleccione Image Ubuntu 12.04 en el menú desplegable de imagen

22 . Escriba un nombre en el campo Nombre de instancia

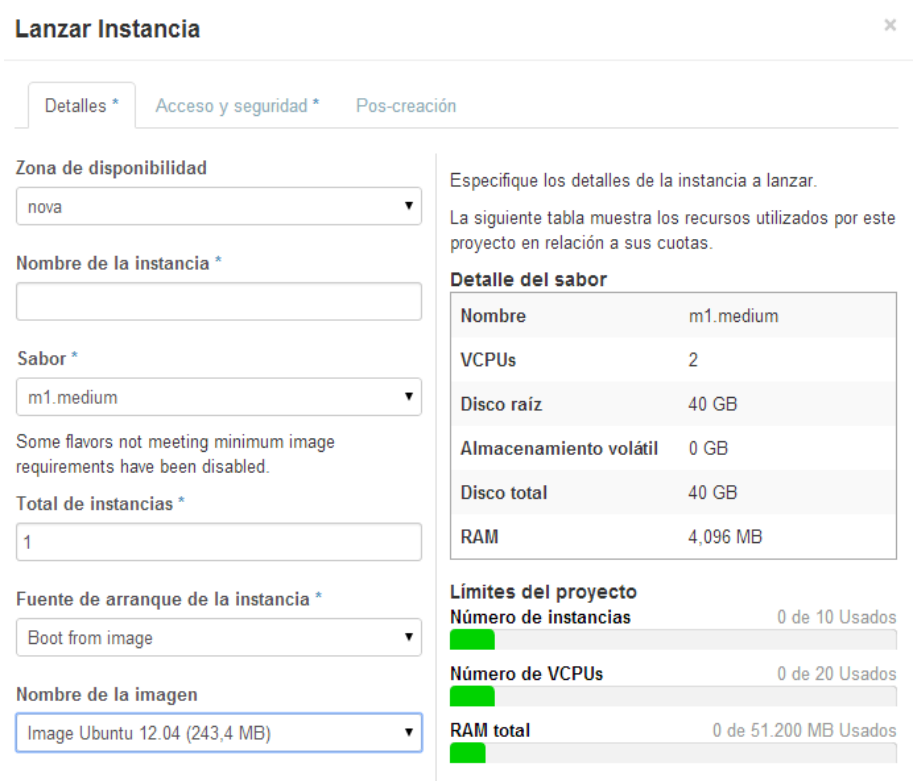
23 . Elija un flavor (m1.small) en el menú desplegable

24 . En instancias de campo escoja, tipo 1

25 . Haga clic en el enlace de acceso y seguridad en la parte superior de la ventana

26 . En la ventana desplegable, seleccione la keypair generadas en el paso 5.

27 . Haga clic en Iniciar. La instancia se generó y el estado pasará de Build a Activo.



Crear una instancia (Command Line Interface)

28 . Autenticar, haga clic en Configuración en la esquina superior derecha del Dashboards

29 . Haga clic en OpenStack API

30 . Haga clic en Descargar archivo RC. Este archivo se descargará en su equipo.

31 . Escriba source openrc.sh para autenticarse.

32 . Para asegurarse de que se haya autenticado, hagamos una lista de las instancias. Escriba 'nova list'.

33 . Crear un keypair. Escriba ' nova keypair-add keypair_name> keypair_name.pem' el símbolo '>' anterior es una redirección. El comando

creará la keypair y creara el archivo .pem que se utilizará cuando ingresemos por SSH a la instancia.

34 . Ahora vamos a lanzar la instancia. Escriba 'nova help boot' para ver una lista de opciones para el boot nova

35. Escriba 'nova boot –flavor m1.tiny --image<imagen ID> --key-name <keyname> <instance name>

Nota: Puede obtener el ID de imagen escribiendo ' nova image-list' y el nombre de clave, escriba ' nova keypair-list'

36. Cree una IP flotante por lo que la instancia puede ser de acceso público.

Escriba 'floating-ip-create'

37. Agregue la IP flotante a la instancia

Escriba ' nova add-floating-ip <Server name> <IP address>

La dirección IP es la IP flotante creada anteriormente. Escriba 'nova list' para obtener el nombre del servidor.

38. Conectarse mediante ssh a la instancia

ssh -i <file.pem> root @ floating_ip

Migre la instancia utilizando el Dashboard

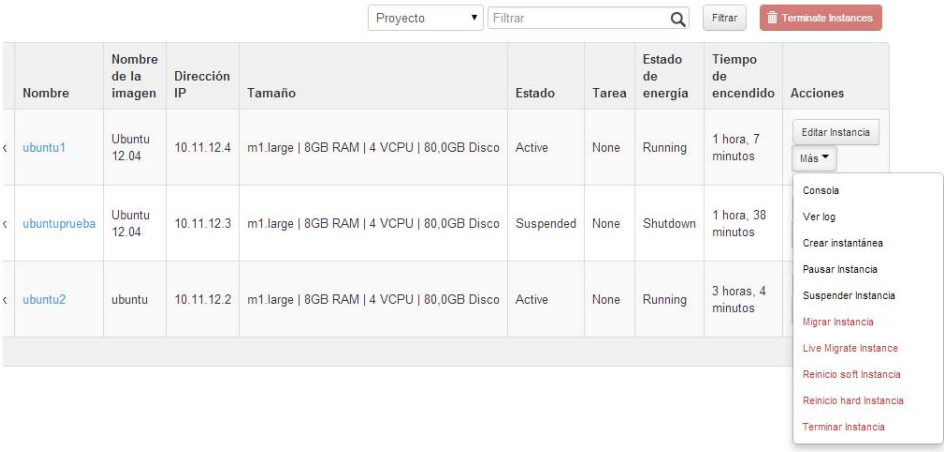
39. Haga clic en el panel Proyecto

40. Haga clic en Instancias

41. En Acciones, haga clic en la flecha para abrir el menú desplegable

42. Seleccione Migrar

43. En el menú desplegable, seleccione un host y haga clic en Migrar.



Migre la instancia mediante la CLI

44. Escriba ' nova host-list' para listar las direcciones IP de host

45. Escriba 'nova-list' para listar el ID de instancia

46. Migre la instancia

Escriba 'nova gc-migrate --dest <destino host> <InstanceID>

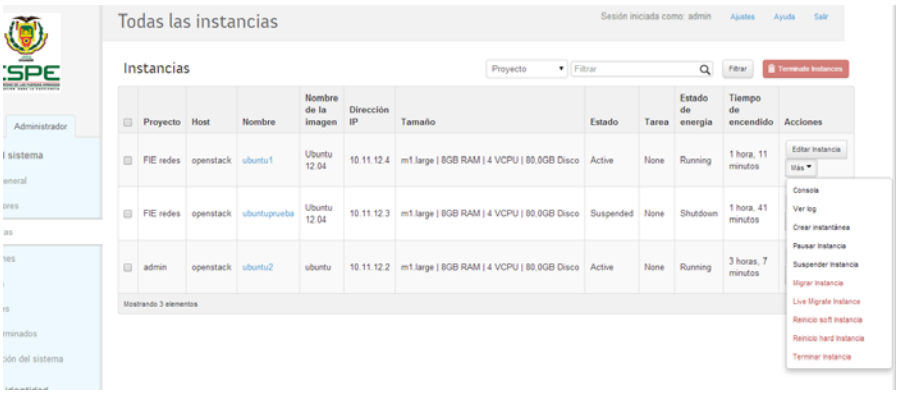
Escriba 'nova list' para ver el cambio de estado de ejemplo para migrar

2. Crear instantáneas de instancias

Crear una instantánea desde el Dashboard

47. Haga clic en Instancias

48. Haga clic en botón de Crear instantánea



Crear una instantánea desde la CLI

49. Escriba 'nova list' para listar las instancias en ejecución

50. Escriba ' nova image-create <server> <snapshot_name>

TABLA 7 : COMANDOS EN NOVA

Listar los comandos Nova disponibles	**$ nova help**
Comando para administrar Proyectos	
Crear un nuevo flavor	flavor-create
Crear un nuevo flavor	flavor-create
Borrar un flavor especifico	flavor-delete
Configurar o desconfigurara extra_spec para un flavor	flavor-key
muestra una lista flavors disponibles	flavor-list
Show detalles sobre el flavor.	flavor-show
Comandos de IP Flotante	
Asignar una ip flotante al tenant.	floating-ip-create

CONTINUA →

Desasignar una IP flotante.	floating-ip-delete
Listar las IPs flotantes para un tenant	Floating-ip-list
Comandos Keypair	
Crea una nueva keypair para usar con una instancia	keypair-add
Borra la keypair	keypair-delete
Presenta una lista de keypairs para un usuario	keypair-list
Comandos Image	
Crea a nueva imagen tomando una snapshot	image-create
Borra una imagen	image-delete
Presenta una lista de imagenes para bootear	image-list
Agrega o elimina metadata de una image	image-meta
Show detalles sobre una imagen	image-show
Comandos Volume	
Agrega un volume a un server	volume-attach
Agrega un nuevo volume	Volume-create
Elimina un volume.	volume-delete
Desconectar un volume de un server.	volume-detach
Lista todos los volumes.	volume-list
Muestra detalles de un volume	volume-show
Agrega un nuevo snapshot.	volume-snapshot-create
Elimina un snapshot.	volume-snapshot-delete
Lista todos los snapshots	Volume-snapshot-list
Muestra detlles sobre una snapshot.	volume-snapshot-show

6. CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

Bajo la propuesta descrita en los objetivos de nuestro proyecto, diseño e implementación de una nube computacional basados en Openstack, nuestras conclusiones son las siguientes:

Referentes al diseño:

- Openstack es un producto que tiene la flexibilidad necesaria para acoplarse a diferentes infraestructuras, en nuestro caso probamos la solución en varios ambientes de hardware y en todos los casos operó de acuerdo a lo esperado.
- En el diseño de la nube notamos que mínimo se deben considerar dos segmentos de red para su operación porque separa su red de administración de la de acceso público.

Referentes a la implementación.

- Openstack opera en plataformas de 64 bits exclusivamente, probamos con un nodo de 32 bits pero no fue posible agregarlo a la plataforma.
- Los costes de implementación de las pruebas se concentraron en el hardware ya que todas las aplicaciones utilizadas durante el proyecto son de Código Abierto.

- La implementación de la plataforma mediante el uso de del paquete FUEL resulta más sencilla de realizar y el soporte brindado por la comunidad hace que su mantenimiento sea constante.

Aspectos generales:

- La curva de aprendizaje para implementar es demasiado larga con respecto a otras soluciones de Código Abierto debido, fundamentalmente, a los continuos cambios en la documentación de soporte que brinda la Comunidad, en algunos casos llegamos a identificar que publicaban los manuales a diario.

6.2. RECOMENDACIONES

Siendo Openstack una solución soportada por un Comunidad con amplio espectro y fruto de nuestro contacto con soluciones de Código Abierto en general, hemos querido que nuestras recomendaciones hagan mención de la experiencia resultante de la investigación e implementación de la plataforma realizada por Nosotros, pero también agregando algunas observaciones realizadas por Miembros de la Comunidad que aportan y enriquecen las mismas. Recomendaciones realizadas bajo la óptica de los errores más frecuentes al momento de realizar una implementación cuyo objetivo apunta prevenirlos, identificarlos y de alguna forma a resolverlos.

Los errores más frecuentes han sido determinados por los desarrolladores y por los usuarios de Openstack con experiencia desde las versiones iniciales,

en nuestro caso desde la versión Essex. Hay que considerar que antes del proyecto Horizon, todo el ciclo de vida de las Máquinas Virtuales se administraba mediante comandos CLI usando principalmente el RESTful API de NOVA para su control, por lo que había una alta posibilidad de equivocarse ya sea por motivos de sintaxis de la instrucción o por desconocimiento a fondo de los comandos disponibles. Luego ya con el desarrollo de los otros proyectos, como por ejemplo Dashboard de Horizon, los puntos de posibles fallos se trasladaron originalmente ubicados en la operación y manejo hacia el proceso de instalación ya que el incremento en la cantidad de proyectos la hacían más compleja.

Sintetizando nuestra experiencia así como también lo mencionado por CloudScaling (Bias, State of the Stack, 2013), Mirantis y la Comunidad de Openstack en sus respectivos portales de consulta y soporte, realizamos las siguientes recomendaciones:

Openstack tiene Licencia de tipo Open Source y gratuito, por lo tanto no hay que preocuparse de los presupuesto. Se recomienda en este caso considerar aspectos tales como la infraestructura sobre la que se instala el programa, así como los constes de mantención de la misma. No por ser gratuito debe ser inestable o inseguro, siendo estos dos aspectos los de mayor atención en el software libre, hay que dedicar el presupuesto necesario para realizar las pruebas que garanticen un el paso previo hacia la puesta en producción definitiva. Hay que considerar también presupuestos para cubrir los costes de la migración de la plataforma anterior hacia Openstack

cualquiera que esta haya sido y luego los de operación y mantenimiento, más aún si queremos una arquitectura de ALTA DISPONIBILIDAD.

- *Hágalo Usted mismo*. Openstack es un proyecto en permanente evolución y desarrollo, cada 6 meses surge una nueva actualización o versión renovada que algunas veces incluye un componente totalmente nuevo, como por ejemplo para la versión Icehouse, planificada para Abril del 2014, se propone la integración de los componentes Ceilometer, Heat y Marconi. Por lo que se recomienda al momento de realizar una implementación de Openstack, el apoyo de un socio estratégico especializado y con mayor experiencia en este tipo de proyectos que colabore con los recursos operativos y logísticos que permitan concretar el objetivo de manera efectiva y a tiempo. Conforme ha ido evolucionando Openstack, algunos miembros de la Comunidad de Desarrolladores han formado empresas dedicadas a la implementación y soporte de la solución, otro ejemplo es la alianza *Openstack Foundation* que agrupa todas las compañías del sector privado que apuestan comercialmente por Openstack principalmente ofertando sus propios procesos de instalación y soporte. Entre los más conocidos están PistonCloud, RedHat, Canonical, Mirantis y CloudScaling,

- *Terminología, todos la entienden*: Openstack es el resultado de la permanente colaboración de una Comunidad desplegada a nivel mundial, la Openstack Community. Cada uno de los servicios de Openstack se desarrolla bajo la dirección de un Coordinador que se encarga de gestionar desde el modelo de programación, las pruebas de funcionamiento previas y la difusión del resultado final hacia el resto de la Comunidad. Sin embargo la adopción

de Openstack por parte de las empresas involucradas en el negocio de las Tecnologías de la Información, entre las que podemos nombrar a las orientadas al procesamiento como por ejemplo Intel; de networking y almacenamiento, entre ellas Intel, HP, Cisco, NetApp, y otras más que han generado su propios para referirse a un mismo tema, un ejemplo es que PistonCloud y CloudScaling definen como *Project* a los que la Comunidad Openstack define como *Tenant*. Esta diversidad de términos obliga un esfuerzo extra al momento de investigar sobre el funcionamiento de una u otra solución, por lo que recomienda ir creando un glosario de términos relacionados.

- *Los sistemas anteriores son incompatibles*. Siendo Openstack considerado como un middleware, puede instalarse sobre cualquier infraestructura existente. La flexibilidad de Openstack le permite iniciar con una implementación pequeña y luego ir aumentado los nodos de procesamiento, almacenamiento o de cómputo conforme sea el requerimiento de los usuarios. Esta escalabilidad hace que Openstack pueda operar a la par de los sistemas anteriores, ya sean en hardware o las plataformas visualizadas, por lo que recomendamos realizar las pruebas de la implementación usando los equipos existentes para asegurar su funcionamiento previo a la adquisición de infraestructura nueva siempre que cumpla por lo menos con el requisito de la virtualización.

- *No es necesario contactar a los desarrolladores:* El error más frecuente cuando se realizar una implementación de una solución bajo código abierto es la falta de un canal de contacto con los desarrolladores de la misma, por lo

que recomendamos involucrarse de una u otra manera en el desarrollo de OpenStack de tal manera que sea posible reportar bugs o fallas en la operación y que ayuden a los programadores a generar su inmediata reparación con el objetivo de conseguir un sistema más estable y confiable. Así mismo la Openstack Community incluye a muchas personas listas a resolver cualquier duda que ayude a realizar una instalación menos complicada.

- *No hay que especializarse, al final es solo Linux:* Openstack puede instalarse en distintas plataformas Linux, entre ellas Ubuntu, Red Hat y Suse, sin embargo su programación está realizada en Python por lo que se recomienda un conocimiento básico previo, tanto del entorno de operación así como del entorno de programación con el objetivo de realizar las modificaciones al código que permitan afinar el sistema de acuerdo a cada necesidad. En nuestro caso la elaboración del presente Proyecto significó un reto no solo con respecto a la aplicación de Openstack sino que también nos impulsó al estudio de Linux y MySQL.

- *La Nube se mantiene y repara por sí sola.* El nivel de automatización y la fiabilidad de una nube debe ser un aspecto importante a considerar, si bien es cierto que la nube puede configurarse bajo la modalidad HA, Alta Disponibilidad, de tal manera que si se presenta algún inconveniente en uno de sus componentes, automáticamente otro nodo ingrese al sistema y el servicio continua sin ser afectado, sin embargo Openstack es un proyecto al que permanentemente se le agregan nuevos componentes que incrementan

su complejidad , por lo que se recomienda implementar un cronograma de entrenamiento continuo de los encargados de su administración.

- *Equivocarse no es opción:* Los cambios generalmente tiene obligatoriamente una curva de aprendizaje, más aún cuando el cambio incluye tecnologías en permanente desarrollo por lo que las probabilidades a equivocarse son mayores. En nuestro caso tuvimos que dedicar más tiempo a la instalación del sistema de lo que teníamos planificado, justamente debido a las modificaciones diarias que habían en los manuales de instalación con respecto a las versiones de los componentes disponibles en los repositorios de Canonical. Por tal motivo recomendamos llevar una bitácora interna que referencie las modificaciones realizadas durante los diferentes procesos de instalación de Openstack.

Ya para finalizar nos preguntamos entre nosotros si recomendaríamos el uso de Openstack y nuestra respuesta es afirmativa pero considerando los puntos anteriormente mencionados.

7. REFERENCIAS BIBLIOGRÁFICAS

Babcock, C. (1 de 12 de 2013). *InformationWeek.* Obtenido de http://www.informationweek.com/infrastructure/cloud-infrastructure/gartner-50--of-enterprises-use-hybrid-cloud-by-2017/d/d-id/1111769?

Bakken, D. (2009). *MIDDLEWARE*. Obtenido de Washington State University: http://www.eecs.wsu.edu/~bakken/middleware.htm

Bias, R. (2013). *Cloud Scaling.* Obtenido de http://www.cloudscaling.com/resources/

Bias, R. (09 de 11 de 2013). *State of the Stack.* Obtenido de Openstack Foundation: http://www.youtube.com/watch?v=S8Q8ASLQ3t8

Buyya Rajkumar, J. B. (2011). *Cloud Computing - Principles and Paradigms.* Wiley.

Cisco. (2013). *Cisco Global Cloud Index: Forecast and Methodology, 2012–2017.* Obtenido de http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns1175/Cloud_Index_White_Paper.html

ClusterLabs. (2013). *Peacemaker*. Obtenido de http://clusterlabs.org/

Comisario, A. (2013). *DataCenter Dynamics*. Obtenido de http://www.youtube.com/watch?v=4oqm3IQNoKc

IBM. (2009). *IBM Perspective on Cloud Computing. The 'next big thing' or 'another fad'?* Obtenido de http://www.itcio.es/cloud-computing

IBM. (2010). *IBM DeveloperWorks*. Obtenido de https://www.ibm.com/developerworks/

IBM. (s.f.). *Cloud computing for the enterprise: Part 1: Capturing the cloud*. Obtenido de Cloud computing for the enterprise: Part 1: Capturing the cloud: http://www.ibm.com/developerworks/websphere/techjournal/0904_amrhein/0904_amrhein.html

Jackson, K. (2013). *Openstack Cloud Computing Cookbook.* PACKT Publishing.

KVM. (2013). *Kernel Based Virtual Machine*. Obtenido de http://www.linux-kvm.org/page/Main_Page

Mirantis. (2013). *Mirantis Openstack reference Architecture.* http://www.mirantis.com.

Nurmi, D. (2009). *https://www.eucalyptus.com/.* Obtenido de https://www.eucalyptus.com/

Openstack. (2013). *Hypervisor configuration basics.* Obtenido de http://docs.openstack.org/trunk/config-reference/content/hypervisor-configuration-basics.html

Openstack. (2013). *Openstack Org*. Obtenido de http://openstack.org

Openstack. (2014). *Compute Node*. Obtenido de http://docs.openstack.org/training-guides/content/associate-computer-node.html

Peter Mell, T. G. (2011). *National Institute of Standars and Techonology.* Obtenido de http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Pipe, K. (s.f.). *Openstack Grizzly.* http://www.slideshare.net/emaganap/open-stack-overview-meetups-oct-2013.

Project, P. R. (2010). *The future of the Internet.* Obtenido de http://pewinternet.org/Reports/2010/The-future-of-cloud-computing.aspx

RAE. (26 de 10 de 2013). *Real Academia de la Lengua Española*. Obtenido de http://buscon.rae.es/drae/srv/search?val=paradigma

Swiftstack. (2013). *https://swiftstack.com.* Obtenido de https://swiftstack.com/openstack-swift/architecture/

Tholeti, B. (2011). *Hypervisors, virtualization, and the cloud: Dive into the KVM hypervisor*. Obtenido de http://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare-kvm/index.html

Tom Fifield, Diane Fleming. (2013). *Openstack Operations Guide.* O´REILLY.

Wikipedia. (2013). Obtenido de http://es.wikipedia.org/wiki/Token_(inform%C3%A1tica): http://es.wikipedia.org/wiki/Token_(inform%C3%A1tica)

Wikipedia. (2013). Obtenido de http://es.wikipedia.org/wiki/Remote_Procedure_Call

Wikipedia. (2013). *Wikipedia.* Obtenido de http://es.wikipedia.org/wiki/Complemento_(inform%C3%A1tica)

Wikipedia. (s.f.). *virtualización*. Obtenido de http://es.wikipedia.org/wiki/Virtualizaci%C3%B3n

Wikipedia, W. (2013). *Hipervisor*. Obtenido de http://es.wikipedia.org/wiki/Hipervisor

WSGI.org. (2013). *http://wsgi.readthedocs.org/en/latest/what.html*. Obtenido de http://wsgi.readthedocs.org/en/latest/what.html

8. ANEXOS

8.1. ANEXO I - Proceso de Instalación Inicial

8.1.1. Instalación del Servicio de Identidad Keystone

apt-get install keystone
agregar al archivo keystone.conf
...
[sql]
connection = mysql://keystone:keystonefie@controller/keystone
...

Eliminar la Base de Datos SQLite

Delete the keystone.db

Crear la base de Datos keystone en SQL server administrada por el usuario keystone

mysql -u root -p
CREATE DATABASE keystone;
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'keystonefie';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'keystonefie';

Poblar la base de datos creada con las respectivas tablas
keystone-manage db_sync
Generar un Token randomico hexadecimal
openssl rand -hex 10
2582bbf28532dd81ffb3

Cambiar en /etc/keystone/keystone.conf Admin_Token por el resultado del comando

[DEFAULT]
# A "shared secret" between keystone and other openstack services
admin_token=2582bbf28532dd81ffb3
...
Reiniciar el servicio de Identidad para que los cambios se hagan efectivos

service keystone restart

Generar manualmente las variables de entorno

export OS_SERVICE_TOKEN=e7b974991fa2eb5be952
export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0
Crear el Proyecto admin con la descripción Proyecto Admin

keystone tenant-create --name=admin --description="Admin Tenant"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          Admin Tenant            |
|   enabled   |              True                |
|      id     | b005fda2e968407b90b6ce5ffc8ed859 |
|     name    |              admin               |
+-------------+----------------------------------+
```

Crear el proyecto Service con la descripcion Proyecto Service

keystone tenant-create --name=service --description="Service Tenant"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          Service Tenant          |
|   enabled   |              True                |
|      id     | 1d193d48fd8a443495b806ce2c5044e3 |
|     name    |              service             |
+-------------+----------------------------------+
```

Crear el usuario admin con sus atributos

keystone user-create --name=admin --pass=adminfie --email=admin@example.com

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |        admin@example.com         |
| enabled  |              True                |
|    id    | cc11060497cf4e84a43e4be5bea3e2aa |
|   name   |              admin               |
+----------+----------------------------------+
```

Crear el role Admin

keystone role-create --name=admin

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 62f8c52c21a14f03b33a5bc7fceec431 |
|   name   |              admin               |
+----------+----------------------------------+
```

Agregarle al usuario admin del proyecto admin el role admin

keystone user-role-add --user=admin --tenant=admin --role=admin

Crear el Service keystone de tipo identity descripción Keystone Identity Service

keystone service-create --name=keystone --type=identity --description="Keystone Identity Service"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |    Keystone Identity Service     |
|      id     | 496bb9b131f048ada00c278a2f490cdc |
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
```

b3cabd8946eb4277b6bd06e3589d1564

Crear el endpoint para el servicio Keystone Identity Service

keystone endpoint-create --service-id=b3cabd8946eb4277b6bd06e3589d1564 --publicurl=http://controller:5000/v2.0 --internalurl=http://controller:5000/v2.0 --adminurl=http://controller:35357/v2.0

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |   http://controller:35357/v2.0   |
|      id     | 89a930ea58844ea88e2748963decde12 |
| internalurl |   http://controller:5000/v2.0    |
```

publicurl	http://controller:5000/v2.0
region	regionOne
service_id	496bb9b131f048ada00c278a2f490cdc
+-------------+----------------------------------+

Verificar Identity Service instalado

eliminar las variables de entorno creadas manualmente

unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
Se verifica si el servicio de Identidad está disponible
keystone --os-username=admin --os-password=adminfie --os-auth-url=http://controller:35357/v2.0 token-get

+----------+
| Property |
Value
|
+-------+
| expires |
2014-02-19T18:09:59Z
|
| id |
MIIC8QYJKoZIhvcNAQcCoIIC4jCCAt4CAQExCTAHBgUrDgMCGjCCAUcGCSqGSIb3DQEHAaCCATgEggE0eyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNC0wMi0xOFQxODowOTo1OS43MzAxNjMiLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE5VDE4OjA5OjU5WiIsICJpZCI6ICJwbGFjZWhvbGRlciJ9LCAic2VydmljZUNhdGFsb2ciOiBbXSwgInVzZXIiOiB7InVzZXJuYW1lIjogImFkbWluIiwgInJvbGVzX2xpbmtzIjogW10sICJpZCI6ICJjYzExMDYwNDk3Y2Y0ZTg0YTQzZTRiZTViZWEzZTJhYSIsICJyb2xlcyI6IFtdLCAibmFtZSI6ICJhZG1pbiJ9LCAibWV0YWRhdGEiOiB7Imlzx2FkbWluIjogMCwgInJvbGVzIjogW119fX0xggGBMIIBfQIBATBcMFcxCzAJBgNVBAYTAlVTMQ4wDAYDVQQIDAVVbnNldDEOMAwGA1UEBwwFVW5zZXQxDjAMBgNVBAoMBVVuc2V0MRgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb20CAQEwBwYFKw4DAhowDQYJKoZIhvcNAQEBBQAEggEAW5IVxBQ-7FqiMIADn7F933d6zOWwb2Y2B36-gDrUTmQmTCTOXlsiMDL5U19AKj5WlEZtihb-kLthuMdedaB9grKQFgbeW9RI2qGq4mJj0VgWwxgA7o4tnf6IKn8WAMoBurxOGczVwP1dWHI8m2f-4oUUc2P2loeSR9lQdm9e8iFc+V5vsQ0-2PTCKmy34YnKx4ovxHv7vk+IWmiHwmmnXeHgqypJL84+8gduMDZrYAIhGh6fPf3ScQgSkASkaorKudlFrR5cZgzv-X5WAsJCIun2ulUMTynzCRhksU+VaHfiDEHrENge+X8GLfRFXO2T0REilRTQ+TTLG2+8MUCqwA== |
| user_id |
cc11060497cf4e84a43e4be5bea3e2aa
|
+----------+

keystone --os-username=admin --os-password=adminfie --os-tenant-name=admin --os-auth-url=http://controller:35357/v2.0 token-get

| expires |
2014-02-19T18:12:05Z
|
| id |
MIIEsgYJKoZIhvcNAQcCoIIEozCCBJ8CAQExCTAHBgUrDgMCGjCCAwgGCSqGSIb3DQEHAaCCAvkEggL1eyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNC0wMi0xOFQxODoxMjowNS44ODAxNjAiLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE5VDE4OjEyOjA1WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIkFkbWluIFRlbmFudCIsICJlbmFibGVkIjogdHJ1ZSwgImlkIjogImIwMDVmZGEyZTk2ODQwN2I5MGI2Y2U1ZmZjOGVkODU5IiwgIm5hbWUiOiAiYWRtaW4ifX0sICJzZXJ2aWNlQ2F0YWxvZyI6IFt7ImVuZHBvaW50cyI6IFt7ImFkbWluVVJMIjogImh0dHA6Ly9jb250cm9sbGVyOjM1MzU3L3YyLjAiLCAicmVnaW9uIjogInJlZ2lvbk9uZSIsICJpbnRlcm5hbFVSTCI6ICJodHRwOi8vY29udHJvbGxlcjo1MDAwL3YyLjAiLCAiaWQiOiAiNWY1ZWExNDc4ZjdhNDUxZjlhMDQ3MmQyNjljMDgyNjciLCAicHVibGljVVJMIjogImh0dHA6Ly9jb250cm9sbGVyOjUwMDAvdjIuMCJ9XSwgImVuZHBvaW50c19saW5rcyI6IFtdLCAidHlwZSI6ICJpZGVudGl0eSIsICJuYW1lIjogImtleXN0b25lIn1dLCAidXNlciI6IHsidXNlcm5hbWUiOiAiYWRtaW4iLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogImNjMTEwNjA0OTdjZjRlODRhNDNlNGJlNWJlYTNlMmFhIiwgInJvbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9XSwgIm5hbWUiOiAiYWRtaW4ifSwgIm1ldGFkYXRhIjogeyJpc19hZG1pbiI6IDAsICJyb2xlcyI6IFsiNjJmOGM1MmMyMWExNGYwM2IzM2E1YmM3ZmNlZWM0MzEiXX19fTGCAYEwggF9AgEBMFwwVzELMAkGA1UEBhMCVVMxDjAMBgNVBAgMBVVuc2V0MQ4wDAYDVQQHDAVVbnNldDEOMAwGA1UECgwFVW5zZXQxGDAWBgNVBAMMD3d3dy5leGFtcGxlLmNvbQIBATAHBgUrDgMCGjANBgkqhkiG9w0BAQEFAASCAQBn2bIkFGv2Jn8jfE6QswzXh1LRFCitqSEp662hQL1kI46u6C4N+QUi34rgKrnHOeVGQHORpYuGNq3FL-RG7+JFmCAAxx+F9nhDAadrs+n4HoXg-Ewq0nhuFLR9sCKiieb5FZrG-Z02MusamFDAK+C4m6TVk7i53hQEzgJPk4W0jP9Ak9U-2NDFRilonFWHg821wRGTWosiuUks3KvNs2YTGH7si3gNgFJmsJYVV5d959kR-elm9Z2TcAONL8k++OjFcOrAF94Ux7tp+bKgEqMV78k-Ug2HQLXLZQwaFiyA1H-E+p45AQb75JtBFY2FmtDBhCsMhsu6s7U2IbejJhqC |
| tenant_id |
b005fda2e968407b90b6ce5ffc8ed859
|
| user_id |
cc11060497cf4e84a43e4be5bea3e2aa
|

Para configurar la variables de entorno forma menos manual creamos el archivo

keystonerc

vi keystonerc en /home/openstack/keystonerc

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

Ejecutamos para configurar las variables de entorno

source keystonerc

Realizamos un ejemplo para solicitar un token

keystone token-get


```
+--------+
| Property |
Value
|
+----+
| expires |
2014-02-19T18:16:26Z
|
|  id  |
MIIEsgYJKoZIhvcNAQcCoIIEozCCBJ8CAQExCTAHBgUrDgMCGjCCAwgGCSqGSIb3DQEH
AaCCAvkEggL1eyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNC0wMi
0xOFQxODoxNjoyNi42MDI2OTQiLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE5VDE4OjE2OjI2
WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIkFkb
WluIFRlbmFudCIsICJlbmFibGVkIjogdHJ1ZSwgImlkIjogImIwMDVmZGEyZTk2ODQwN2I
5MGI2Y2U1ZmZjOGVkODU5IiwgIm5hbWUiOiAiYWRtaW4ifX0sICJzZXJ2aWNlQ2F0Y
WxvZyI6IFt7ImVuZHBvaW50cyI6IFt7ImFkbWluVVJMIjogImh0dHA6Ly9jb250cm9sbG
VyOjM1MzU3L3YyLjAiLCAicmVnaW9uIjogInJlZ2lvbk9uZSIsICJpbnRlcm5hbFVSTCI6ICJ
odHRwOi8vY29udHJvbGxlcjo1MDAwL3YyLjAiLCAiaWQiOiAiNWY1ZWExNDc4ZjdhND
UxZjlhMDQ3MmQyNjljMDgyNjciLCAicHVibGljVVJMIjogImh0dHA6Ly9jb250cm9sbGV
yOjUwMDAvdjIuMCJ9XSwgImVuZHBvaW50c19saW5rcyI6IFtdLCAidHlwZSI6ICJpZGVu
dGl0eSIsICJuYW1lIjogImtleXN0b25lIn1dLCAidXNlciI6IHsidXNlcm5hbWUiOiAiYWRta
W4iLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogImNjMTEwNjA0OTdjZjRlODRhNDNlNG
JlNWJlYTNlMmFhIiwgInJvbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9XSwgIm5hbWUiOiAiY
WRtaW4ifSwgIm1ldGFkYXRhIjogeyJpc19hZG1pbiI6IDAsICJyb2xlcyI6IFsiNjJmOGM1M
```

mMyMWExNGYwM2IzM2E1YmM3ZmNlZWM0MzEiXX19fTGCAYEwggF9AgEBMFww
VzELMAkGA1UEBhMCVVMxDjAMBgNVBAgMBVVuc2V0MQ4wDAYDVQQHDAVVbnNl
dDEOMAwGA1UECgwFVW5zZXQxGDAWBgNVBAMMD3d3dy5leGFtcGxlLmNvbQIBA
TAHBgUrDgMCGjANBgkqhkiG9w0BAQEFAASCAQCXiEqeBvgWzTzXLhATei8j5L4aukJLn
zTtyj3v9mnKXtRfKE0rB69ZYUgPbjTJGIVgiB4si0qO0wJEd87RDMZSHfnO31GSk7vP4M
REfuYE7YlvWllSfI98HXP66YWQK5jV7rPj6JiHrJssVuO1NqSb0GJpFs1+OhPUwi6W51w-
u5s0dzUv68ZBEutj-exnhvGd0PWbhylARbySq5MlFbEppuh6e1bR3-XaLX7r-
lNyn8Zv1w88BTN3ayVFmBb-
K236Isv0uwyG6LHAuzhECTqOGazhg24IDDpv2NVG2BUFnBIJLZZR99Foyy+zUxGGicXa
wFOq3s+wEFTi6ATp9kw7 |
| tenant_id |
b005fda2e968407b90b6ce5ffc8ed859
|
| user_id |
cc11060497cf4e84a43e4be5bea3e2aa
|
+----------+

Verificamos la lista de usuarios de keystone

keystone user-list

```
+----------------------------------+-------+---------+-------------------+
|                id                |  name | enabled |       email       |
+----------------------------------+-------+---------+-------------------+
| cc11060497cf4e84a43e4be5bea3e2aa | admin |   True  | admin@example.com |
+----------------------------------+-------+---------+-------------------+
```

8.1.2. Instalación del Servicio de Imágenes Glance

apt-get install glance python-glanceclient

Agregamos la conexion a la base de datos SQL

Edit /etc/glance/glance-api.conf y /etc/glance/glance-registry.conf

```
...
[DEFAULT]
...
# SQLAlchemy connection string for the reference implementation
```

```
# registry server. Any valid SQLAlchemy connection string is fine.
#See:http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/connections.html#
sqlalchemy.create_engine
sql_connection = mysql://glance:glancefie@controller/glance
...
```

Eliminamos la base SQLite

Delete the glance.sqlite

Creamos la base de datos glance en SQL server administrada por el usuario glance

```
mysql -u root -p
CREATE DATABASE glance;
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY
'glancefie';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'glancefie';
```

Poblamos la base de datos creada

```
glance-manage db_sync
```

Creamos el usuario glance en Keystone

```
keystone user-create --name=glance --pass=glancefie –email=glance@example.com

+----------+----------------------------------+
| Property |             Value                |
+----------+----------------------------------+
|  email   |       glance@example.com         |
| enabled  |              True                |
|    id    | 309b06e1ae9a4863b637df04a896ac49 |
|   name   |             glance               |
+----------+----------------------------------+
```

Asignamos el role admin al usuario creado

```
keystone user-role-add --user=glance --tenant=service --role=admin
```

Agregamos los parametros de autenticacion del servicio glance

Edit the /etc/glance/glance-api.conf y /etc/glance/glance-registry.conf

```
[keystone_authtoken]
```

```
...
auth_uri = http://controller:5000/v2.0
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = glancefie

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

Agregamos las credenciales para autorización de token

Add the credentials to the /etc/glance/glance-api-paste.ini and /etc/glance/glance-registry-paste.ini

[filter:authtoken]

```
paste.filter_factory=keystoneclient.middleware.auth_token:filter_factory
auth_host=controller
admin_user=glance
admin_tenant_name=service
admin_password=glancefie
```

Creamos el servicio glance con descripción Glance Image Service

```
keystone service-create --name=glance --type=image --description="Glance Image Service"

+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Glance Image Service       |
|      id     | 84b94cdc0c38419eb3d8c8a44c32389e |
|     name    |              glance              |
|     type    |             image                |
+-------------+----------------------------------+
```

Creamos el endpoint correspondiente a Glance

```
keystone endpoint-create --service-id=63656a318b4849d2891ec597b1757160 --publicurl=http://controller:9292 --internalurl=http://controller:9292 --
```

adminurl=http://controller:9292

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |      http://controller:9292      |
|      id     | c31c691de6e54a619e9da12944c74b50 |
| internalurl |      http://controller:9292      |
|  publicurl  |      http://controller:9292      |
|    region   |           regionOne              |
|  service_id | 63656a318b4849d2891ec597b1757160 |
+-------------+----------------------------------+
```

Reiniciamos el servicio para efectuar los cambios

service glance-registry restart
service glance-api restart
creamos el directorio images en /home/openstack y descargamos la imagen de cirrOS
mkdir images
cd images/
wget http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img

glance image-create --name="CirrOS 0.3.1" --disk-format=qcow2 --container-format=bare --is-public=true < cirros-0.3.1-x86_64-disk.img

agregamos la imagen de CirrOS al servicio Glance

cd /var/lib/glance/images

glance image-create --name="CirrOS 0.3.1" --disk-format=qcow2 --container-format=bare --is-public=true < cirros-0.3.1-x86_64-disk.img

```
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | d972013792949d0d3ba628fbe8685bce     |
| container_format | bare                                 |
| created_at       | 2014-02-18T22:26:51                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
```

```
| id              | 8e05afd3-fe67-4002-a1d3-0371a425b01a |
| is_public       | True                                 |
| min_disk        | 0                                    |
| min_ram         | 0                                    |
| name            | CirrOS 0.3.1                         |
| owner           | None                                 |
| protected       | False                                |
| size            | 13147648                             |
| status          | active                               |
| updated_at      | 2014-02-18T22:26:52                  |
+-----------------+--------------------------------------+
```

Verificamos que la imagen de CirrOS se encuentra disponible

```
glance image-list
+--------------------------------------+--------------+-------------+------------------+----------+--------
+
| ID                                   | Name         | Disk Format | Container Format | Size     |
Status |
+--------------------------------------+--------------+-------------+------------------+----------+--------
+
| 8e05afd3-fe67-4002-a1d3-0371a425b01a | CirrOS 0.3.1 | qcow2       | bare
| 13147648 | active |
+--------------------------------------+--------------+-------------+------------------+----------+--------
+
```

Realizamos una prueba de la imagen CirrOS

```
file cirros-0.3.1-x86_64-disk.img
cirros-0.3.1-x86_64-disk.img: QEMU QCOW Image (v2), 41126400 bytes
```

8.1.3. Instalación del Servicio de Procesamiento NOVA

```
apt-get install nova-novncproxy novnc nova-api nova-ajax-console-proxy nova-cert
nova-conductor nova-consoleauth nova-doc nova-scheduler python-novaclient
```

Agregamos la conexion hacia la base de datos SQL con el usuario nova

Edit the /etc/nova/nova.conf

[database]

# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:novafie@controller/nova
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = novafie

Agregamos la conexion hacia rabbitmq
configuration group of the /etc/nova/nova.conf file:
[DEFAULT]
rpc_backend = nova.rpc.impl_kombu
rabbit_host = controller
rabbit_password = rabbitfie

Eliminamos la base SQLite de nova
Delete the nova.sqlite
Creacion de la base de datos nova administrada por el usuario nova

mysql -u root -p

CREATE DATABASE nova;
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'novafie';

GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'novafie';

poblamos la base de datos creada
nova-manage db sync
Agregamos los puertos de escucha para el servicio VNC

Edit the /etc/nova/nova.conf
[DEFAULT]
...
my_ip=192.168.1.12
vncserver_listen=192.168.1.12
vncserver_proxyclient_address=192.168.1.12

Creamos el usuario nova en el Servicio Identity

keystone user-create --name=nova --pass=novafie --email=nova@example.com

+----------+----------------------------------+

| Property | Value |
+----------+----------------------------------+
email	nova@example.com
enabled	True
id	f63b73761e1342f6a40e1b4a7871d4d9
name	nova
+----------+----------------------------------+

Le damos el rol de admin al usuario nova

keystone user-role-add --user=nova --tenant=service --role=admin
Definimos la estrategia de authenticacion con Keystone

Edit /etc/nova/nova.conf

[DEFAULT]
...
auth_strategy=keystone

Agregamos el filtro para utilizar el servicio nova y keystone
Add the credentials to the /etc/nova/api-paste.ini
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000/v2.0
admin_tenant_name = service
admin_user = nova
admin_password = novafie

*Verificamos que la variable api_paste_config este correcta e*n /etc/nova/nova.conf

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

Creamos el servicio nova con la descripcion Nova Compute Service en keystone

keystone service-create --name=nova --type=compute --description="Nova Compute service"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Nova Compute service       |
|      id     | 01c88a48a95946538f01555aab29db15 |
|     name    |              nova                |
|     type    |             compute              |
+-------------+----------------------------------+
```

Creamos el endpoint para el servicio nova

```
keystone endpoint-create --service-id=62c321effd3f4468acd5702f24eb340f --
publicurl=http://controller:8774/v2/%\(tenant_id\)s --
internalurl=http://controller:8774/v2/%\(tenant_id\)s --
adminurl=http://controller:8774/v2/%\(tenant_id\)s
+-------------+-----------------------------------------+
|   Property  |                Value                    |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8774/v2/%(tenant_id)s |
|      id     |     4aaaca391fe84dd6ae333f639e6adcba    |
| internalurl | http://controller:8774/v2/%(tenant_id)s |
|  publicurl  | http://controller:8774/v2/%(tenant_id)s |
|    region   |               regionOne                 |
|  service_id |     01c88a48a95946538f01555aab29db15    |
+-------------+-----------------------------------------+
```

Reiniciamos los servicios de nova

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

Verificamos la lista de imágenes mediante nova

```
nova image-list
+--------------------------------------+--------------+--------+--------+
| ID                                   | Name         | Status | Server |
+--------------------------------------+--------------+--------+--------+
| 8e05afd3-fe67-4002-a1d3-0371a425b01a | CirrOS 0.3.1 | ACTIVE |        |
+--------------------------------------+--------------+--------+--------+
```

8.1.4. Instalación del Servicio de red Neutron

Creamos la base de datos a ser administrada por el usuario neutron

```
mysql -u root -p
CREATE DATABASE neutron;
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY
'neutronfie';
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'neutronfie';
```

Poblamos la base de datos creada
Revisamos la lista de usuarios keystone creados

```
keystone tenant-list

+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| b005fda2e968407b90b6ce5ffc8ed859 |  admin  |   True  |
| 1d193d48fd8a443495b806ce2c5044e3 | service |   True  |
+----------------------------------+---------+---------+
```

Revisamos los roles existentes en keystone

```
keystone role-list
+----------------------------------+----------+
|                id                |   name   |
+----------------------------------+----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| 62f8c52c21a14f03b33a5bc7fceec431 |  admin   |
+----------------------------------+----------+
```

Creacion del usuario neutron

```
keystone user-create --name=neutron --pass=neutronfie --
email=neutron@example.com
```

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |       neutron@example.com        |
| enabled  |               True               |
|    id    | 7a1d41bcf324454c861b09193259e685 |
|   name   |             neutron              |
+----------+----------------------------------+
```

Creación del service neutron con descripción Open Stack Networking Service

keystone service-create --name=neutron --type=network --description="OpenStack Networking Service"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |   OpenStack Networking Service   |
|      id     | 6cff826cd0864ca78bfa57338baae56b |
|     name    |             neutron              |
|     type    |            network               |
+-------------+----------------------------------+
```

Creación del endpoint correspondiente al Servicio Openstack Networking

keystone endpoint-create --service-id=6cff826cd0864ca78bfa57338baae56b --publicurl=http://controller:9696 --adminurl=http://controller:9696 --internalurl=http://controller:9696

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |      http://controller:9696      |
|      id     | 3bf61599e6984677bd6477e20e13d9dd |
| internalurl |      http://controller:9696      |
|  publicurl  |      http://controller:9696      |
|    region   |            regionOne             |
|  service_id | 6cff826cd0864ca78bfa57338baae56b |
+-------------+----------------------------------+
```

apt-get install neutron-server

Agregamos la forma de autenticar el usuario neutron con Keystone

Edit /etc/neutron/neutron.conf

auth_host = controller
admin_tenant_name = service
admin_user = neutron
admin_password = neutronfie
auth_url = http://controller:35357/v2.0
auth_strategy = keystone

Agregar la forma de usar la herramienta rabbitmq cola de mensajes

rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_port = 5672
rabbit_password = rabbitfie
Agregamos la conexión a la base de datos SQL
[database]
connection = mysql://neutron:neutronfie/neutron

Instalación del Servicio Nova Compute

keystone user-create --name=nova --pass=novafie --email=nova@example.com

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |        nova@example.com          |
| enabled  |              True                |
|    id    | a0ccb8690bf440c7a0ee90b4f8029585 |
|   name   |              nova                |
+----------+----------------------------------+
```

Creación del Servicio Nova con descripción Nova Compute Service

keystone service-create --name=nova --type=compute --description="Nova Compute Service"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Nova Compute Service       |
|      id     | 1587c41953414e95a96de22bcfa0acd7 |
|     name    |              nova                |
|     type    |             compute              |
+-------------+----------------------------------+
```

Creación del endpoint correspondiente al Servicio Nova

keystone endpoint-create --service-id=1587c41953414e95a96de22bcfa0acd7 --publicurl=http://controller:8774/v2 --internalurl=http://controller:8774/v2 --adminurl=http://controller:8774/v2

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |    http://controller:8774/v2     |
```

```
|     id     | 92ccf887e98f423c8216ba08290993a9 |
| internalurl |    http://controller:8774/v2     |
|  publicurl  |    http://controller:8774/v2     |
|    region   |          regionOne               |
|  service_id | 1587c41953414e95a96de22bcfa0acd7 |
+-------------+----------------------------------+
```

Instalación de Nova

apt-get install nova-novncproxy novnc nova-api nova-ajax-console-proxy nova-cert nova-conductor nova-consoleauth nova-doc nova-scheduler python-novaclient

8.1.5. Instalación del Componente Dashboard

apt-get install memcached libapache2-mod-wsgi openstack-dashboard
apt-get remove --purge openstack-dashboard-ubuntu-theme
Personalizamos los valores siguientes en local_settings.py
Edit /etc/openstack-dashboard/local_settings.py:
ALLOWED_HOSTS = ['localhost', 'my-desktop']
OPENSTACK_HOST = "controller"
TIME_ZONE = "UTC"
Reiniciamos los servicios Apache Web Server y Memcached

service apache2 restart
service memcached restart
Creación de la base de datos dash en MySQL que será administrada por el usuario dash
mysql -u root -p

GRANT ALL ON dash.* TO 'dash'@'%' IDENTIFIED BY '*dashfie*';
GRANT ALL ON dash.* TO 'dash'@'localhost' IDENTIFIED BY 'dashfie';
Poblar la base de datos creada

usr/share/openstack-dashboard/manage.py syncdb

syncdbCreating tables ...
Creating table django_content_type
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_groups
Creating table auth_user_user_permissions
Creating table auth_user

Creating table django_session
You just installed Django's auth system, which means you don't have any superusers defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'root'): dashuser
Email address: dashuser@example.com
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
Creamos el directorio .blackhole
mkdir -p /var/lib/dash/.blackhole

Reiniciamos los servicios Apache Web Server y Memcached

service apache2 restart
service memcached restart
Reiniciamos el servicio Nova
restart nova-api

8.1.6. **Instalación del Componente Cinder**

apt-get install cinder-api cinder-scheduler

Creación de la base de datos Cinder en MySQL que será administrada por el usuario cinder

mysql -u root -p
CREATE DATABASE cinder;
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'cinderfie';
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'cinderfie';

Poblar la base de datos creada

cinder-manage db sync

Configurar las variables de entorno a través del archivo creado kestonerc

source keystonerc

Creación del usuario cinder en keystone

keystone user-create --name=cinder --pass=cinderfie --email=cinder@example.com

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |       cinder@example.com         |
| enabled  |              True                |
|    id    | af03959f4ff64be28796edd97aafdef9 |
|   name   |             cinder               |
+----------+----------------------------------+
```

Rol admin para el usuario cinder

keystone user-role-add --user=cinder --tenant=service --role=admin

Agregar las credenciales para usar el servicio keystone

/etc/cinder/api-paste.ini.

```
[filter:authtoken]
paste.filter_factory=keystoneclient.middleware.auth_token:filter_factory
auth_host=controller
auth_port = 35357
auth_protocol = http
admin_tenant_name=service
admin_user=cinder
admin_password=cinderfie
```

Agregar las credenciales para usar el servicio rabbitmq cola de mensajes

```
[DEFAULT]
...
rpc_backend = cinder.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = rabbitfie
```

Creación del usuario cinder en keystone

keystone service-create --name=cinder --type=volume --description="Cinder Volume Service"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |      Cinder Volume Service       |
|      id     | e40b056d11024624a78848952430c638 |
|     name    |              cinder              |
|     type    |             volume               |
+-------------+----------------------------------+
```

Creación del endpoint correspondiente al Servicio Cinder

```
keystone endpoint-create --service-id=e40b056d11024624a78848952430c638 --
publicurl=http://controller:8776/v1/%\(tenant_id\)s --
internalurl=http://controller:8776/v1/%\(tenant_id\)s --
adminurl=http://controller:8776/v1/%\(tenant_id\)s
```

```
+-------------+-----------------------------------------+
|   Property  |                Value                    |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8776/v1/%(tenant_id)s |
|      id     |     69e83d3835094685a2f23f854654fdf9    |
| internalurl | http://controller:8776/v1/%(tenant_id)s |
|  publicurl  | http://controller:8776/v1/%(tenant_id)s |
|    region   |               regionOne                 |
|  service_id |     e40b056d11024624a78848952430c638    |
+-------------+-----------------------------------------+
```

Creación del servicio cinderv2 con descripción Cinder Volume Service V2

```
keystone service-create --name=cinderv2 --type=volumev2 --description="Cinder
Volume Service V2"
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     Cinder Volume Service V2     |
|      id     | a5c9777c3f7644fb97c41c379263de56 |
|     name    |             cinderv2             |
|     type    |             volumev2             |
+-------------+----------------------------------+
```

Creación del endpoint correspondiente al servicio Cinder Volume V2

```
keystone endpoint-create --service-id=a5c9777c3f7644fb97c41c379263de56 --
publicurl=http://controller:8776/v2/%\(tenant_id\)s --
```

```
internalurl=http://controller:8776/v2/%\(tenant_id\)s --
adminurl=http://controller:8776/v2/%\(tenant_id\)s

+-------------+-----------------------------------------+
|   Property  |                Value                    |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8776/v2/%(tenant_id)s |
|      id     |     a53f98b477ff4ee3ae7b6bffda12d01c    |
| internalurl | http://controller:8776/v2/%(tenant_id)s |
|  publicurl  | http://controller:8776/v2/%(tenant_id)s |
|    region   |              regionOne                  |
|  service_id |     a5c9777c3f7644fb97c41c379263de56    |
+-------------+-----------------------------------------+
```

Reiniciamos los servicios Cinder para que los cambios se ejecuten

```
service cinder-scheduler restart
service cinder-api restart
```

8.2. ANEXO II: Lanzar una instancia de prueba

```
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
0b:f7:43:10:d1:20:d3:06:11:f6:a2:6c:07:56:f4:41 root@controller
The key's randomart image is:

+--[ RSA 2048]----+
|     .X*E+       |
|     o =oo.      |
|    o ..+        |
|   o o . .       |
|    + o S .      |
|   . . o +       |
|       . o       |
|         .       |
|                 |
```

```
+-----------------+

root@controller:/home/openstack# cd /root/.ssh

root@controller:~/.ssh# nova keypair-add --pub_key id_rsa.pub key

root@controller:~/.ssh# nova keypair-list

+------+-------------------------------------------------+
| Name | Fingerprint                                     |
+------+-------------------------------------------------+
| key  | 0b:f7:43:10:d1:20:d3:06:11:f6:a2:6c:07:56:f4:41 |
+------+-------------------------------------------------+

root@controller:~/.ssh# nova flavor-list

+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+

root@controller:~/.ssh# nova image-list

+--------------------------------------+--------------+--------+--------+
| ID                                   | Name         | Status | Server |
+--------------------------------------+--------------+--------+--------+
| 8a5f9b3c-5364-4d59-b1c9-af6f27f6f4bf | CirrOS 0.3.1 | ACTIVE |        |
+--------------------------------------+--------------+--------+--------+

root@controller:~/.ssh# nova secgroup-add-rule default tcp 22 22 0.0.0.0/0

+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+

root@controller:~/.ssh# nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
```

```
+-------------+-----------+---------+-----------+-------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+-------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+-------------+
```

root@controller:~/.ssh# nova boot --flavor 1 --key-name key --image 8a5f9b3c-5364-4d59-b1c9-af6f27f6f4bf --security_group default cirrOS

```
+-------------------------------------+--------------------------------------+
| Property                            | Value                                |
+-------------------------------------+--------------------------------------+
| OS-EXT-STS:task_state               | scheduling                           |
| image                               | CirrOS 0.3.1                         |
| OS-EXT-STS:vm_state                 | building                             |
| OS-EXT-SRV-ATTR:instance_name       | instance-00000001                    |
| OS-SRV-USG:launched_at              | None                                 |
| flavor                              | m1.tiny                              |
| id                                  | 55b84ba8-af43-45e7-b79f-597d50fa90dd |
| security_groups                     | [{u'name': u'default'}]              |
| user_id                             | 8887fca5e04d4884be7c735deff1f0c7     |
| OS-DCF:diskConfig                   | MANUAL                               |
| accessIPv4                          |                                      |
| accessIPv6                          |                                      |
| progress                            | 0                                    |
| OS-EXT-STS:power_state              | 0                                    |
| OS-EXT-AZ:availability_zone         | nova                                 |
| config_drive                        |                                      |
| status                              | BUILD                                |
| updated                             | 2014-02-20T05:14:53Z                 |
| hostId                              |                                      |
| OS-EXT-SRV-ATTR:host                | None                                 |
| OS-SRV-USG:terminated_at            | None                                 |
| key_name                            | key                                  |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                 |
| name                                | cirrOS                               |
| adminPass                           | a49GW6B5ats3                         |
| tenant_id                           | cf8e325cc32747eba5f9b33c07916d5c     |
| created                             | 2014-02-20T05:14:53Z                 |
| os-extended-volumes:volumes_attached | []                                   |
| metadata                            | {}                                   |
+-------------------------------------+--------------------------------------+
```

root@controller:~/.ssh# nova list

```
+--------------------------------------+--------+--------+------------+-------------+----------+
| ID                                   | Name   | Status | Task State | Power State | Networks |
+--------------------------------------+--------+--------+------------+-------------+----------+
| 55b84ba8-af43-45e7-b79f-597d50fa90dd | cirrOS | ERROR  | None       | NOSTATE
|        |
+--------------------------------------+--------+--------+------------+-------------+----------+

root@controller:~/.ssh# nova list

+--------------------------------------+--------+--------+------------+-------------+----------+
| ID                                   | Name   | Status | Task State | Power State | Networks |
+--------------------------------------+--------+--------+------------+-------------+----------+
| 55b84ba8-af43-45e7-b79f-597d50fa90dd | cirrOS | ERROR  | None       | NOSTATE
|        |
+--------------------------------------+--------+--------+------------+-------------+----------+

root@controller:~/.ssh# nova show 55b84ba8-af43-45e7-b79f-597d50fa90dd

+--------------------------------------+----------------------------------------------------------------------------------------------------------------------------------
----+
| Property                             | Value
|
+--------------------------------------+----------------------------------------------------------------------------------------------------------------------------------
----+
| status                               | ERROR
|
| updated                              | 2014-02-20T05:14:53Z
|
| OS-EXT-STS:task_state                | None
|
| OS-EXT-SRV-ATTR:host                 | None
|
| key_name                             | key
|
| image                                | CirrOS 0.3.1 (8a5f9b3c-5364-4d59-b1c9-af6f27f6f4bf)
|
| hostId                               |
|
| OS-EXT-STS:vm_state                  | error
|
| OS-EXT-SRV-ATTR:instance_name        | instance-00000001
|
| OS-SRV-USG:launched_at               | None
```

```
|
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None
|
| flavor                               | m1.tiny (1)
|
| id                                   | 55b84ba8-af43-45e7-b79f-597d50fa90dd
|
| security_groups                      | [{u'name': u'default'}]
|
| OS-SRV-USG:terminated_at             | None
|
| user_id                              | 8887fca5e04d4884be7c735deff1f0c7
|
| name                                 | cirrOS
|
| created                              | 2014-02-20T05:14:53Z
|
| tenant_id                            | cf8e325cc32747eba5f9b33c07916d5c
|
| OS-DCF:diskConfig                    | MANUAL
|
| metadata                             | {}
|
| os-extended-volumes:volumes_attached | []
|
| accessIPv4                           |
|
| accessIPv6                           |
|
| fault                                | {u'message': u'No valid host was found. ', u'code': 500,
u'details': u'  File "/usr/lib/python2.7/dist-
packages/nova/scheduler/filter_scheduler.py", line 107, in schedule_run_instance |
|                                      |     raise exception.NoValidHost(reason="")
|
|                                      | ', u'created': u'2014-02-20T05:14:54Z'}
|
| OS-EXT-STS:power_state               | 0
|
| OS-EXT-AZ:availability_zone          | nova
|
| config_drive                         |
```