UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Clúster de Virtualización de bajo costo basado en OpenStack

Proyecto integrador

.

Juan Sebastián Sánchez Vaca

Ingeniería de Sistemas

Trabajo de titulación presentado como requisito
para la obtención del título de Ingeniero de Sistemas

Quito, 7 de diciembre de 2016

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE CIENCIAS E INGENIERÍAS

HOJA DE CALIFICACIÓN
DE TRABAJO DE TITULACIÓN

Clúster de Virtualización de bajo costo basado en OpenStack

Juan Sebastián Sánchez Vaca

Calificación:

Nombre del profesor, Título académico — Aldo Cassola, PhD.

Firma del profesor

Quito, 7 de diciembre de 2016

Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Juan Sebastián Sánchez Vaca

Código: 00107176

Cédula de Identidad: 1718366550

Lugar y fecha: Quito, 7 de diciembre de 2016

AGRADECIMIENTO

Primero, agradezco a mi director de trabajo de titulación Aldo Cassola quien me ayudó a consolidar el proyecto en todas sus etapas y me brindo su apoyo tanto en el área técnica como personal. Segundo, a mis profesores que me han ayudado a lo largo de mi formación académica en especial a Juan Alejandro Almendáriz que me enseñó que el conocimiento es inútil sin ética, moral y pasión. Además, quiero agradecer a Herrera Carvajal & Asociados por los recursos donados para el proyecto. Finalmente, quiero agradecer a mis compañeros y colegas de trabajo que me han guiado en ocasiones difíciles y han sido un complemento para mi formación.

DEDICATORIA

Este trabajo está dedicado a mi tío, Freddy Sánchez quien me enseño que no existe lo imposible y que las únicas limitantes para cumplir objetivos son los que uno mismo crea. Sin su apoyo incondicional, motivación y enseñanzas en mi juventud, me hubiese sido imposible tomar las riendas en el camino de la informática y tecnología.

Adicionalmente, quiero dedicar mi trabajo a mi madre cuyas raices de trabajo duro, perseverancia y honestidad han sido un modelo a seguir en mi vida. Sin su apoyo en las distintas etapas de mi vida este trabajo no se hubiese logrado. Finalmente, a mis hermanos cuyas expectativas me motivan cada día a superarme.

RESUMEN

El presente trabajo establece la estructura, funcionamiento e implementación de un clúster de virtualización basado en OpenStack. El fundamento de este trabajo consiste en la necesidad de máquinas virtuales que garanticen alta disponibilidad y bajos costos utilizando alternativas de software abierto. Para esto, se utilizaron 4 nodos base (computadoras) con sistema operativo Ubuntu 16.04. El sistema funcional es capaz de producir máquinas virtuales en tanto existan recursos disponibles y generar un acceso de red para que los clientes de la nube puedan acceder a las máquinas virtuales directamente. Primero, se explica la estructura del sistema y escalabilidad procediendo a explicar el funcionamiento tras la creación de una máquina virtual. Posteriormente, se explica el mantenimiento y descripción de las funciones del sistema así como el proceso de deployment de nodos de cómputo.

Palabras Clave: OpenStack, Cluster, Platform Virtualization, Hypervisor de tipo 1, KVM, Ubuntu.

ABSTRACT

The former paper establishes the structure, functions and deployment of a virtualization cluster based on OpenStack. The foundation of this work bases in the ongoing needs of virtual machines that guarantee high availability and low costs using open source alternatives. For this, 4 base nodes(computers) were used with Ubuntu 16.04 as operating system. The system is capable of spawning virtual machines as long as resources are available and then creates a network access so the cloud clients can interact with the virtual machines directly. First, the structure and scalability of the system is explained. Then an explanation of the behavior after the spawn of a virtual machine is given. Afterwards, the maintenance and a description of the system functions are described as well as the process of deployment of a computing node.

Keywords: OpenStack, Cluster, Platform Virtualization, Type I Hypervisor, KVM, Ubuntu.

TABLA DE CONTENIDO

INTRODUCCIÓN **10**
Antecedentes: 11
Justificación: 12
Objetivos específicos: 13
ESTRUCTURA DEL CLÚSTER **14**
Descripción: 15
Módulo Nova. 16
Módulo Neutron. 16
Módulo de Glance. 17
Módulo Keystone. 17
Módulo Horizon. 18
Infraestructura de los componentes: 18
Esquema de red: 20
INSTALACIÓN DE NODOS DE CÓMPUTO E INSTANCIAS **21**
Descripción: 21
Pasos de configuración realizados por el script de instalación. 21
Descripción de rutas y código para instalación de nuevo nodo. 23
Script para conexión remota a través de SSH Forwarding. 24
FUNCIONAMIENTO DE MÁQUINAS VIRTUALES Y ACCESO **25**
Administración de controlador y opciones adicionales: 26
Usos comunes para Administradores y Usuarios: 27
Ejemplos de funcionamiento de usuario. 28
Casos de modificación de administrador 28
RESULTADOS Y RECOMENDACIONES A FUTURO **29**
Estado de funcionamiento del sistema: 29
Evaluación de nodos de cómputo: 30
Benchmark y stress test del sistema: 31
CONCLUSIONES Y RECOMENDACIONES **32**
Conclusiones: 32
Recomendaciones: 33
REFERENCIAS BIBLIOGRÁFICAS **36**
ANEXO A: DEFINICIONES **37**

ÍNDICE DE FIGURAS

Figura 1. Estructura de sistema y componentes.....15

Figura 2. Esquema de interacción entre componentes.....19

Figura 3. Infraestructura de Red del Sistema.....20

Figura 4. Página principal del componente de administración.....25

Figura 5. Interfaz de creación de nueva instancia.....26

Figura 6.- Carga de Nodo Controlador.....29

Figura 7.- Carga de nodo de Cómputo.....30

ÍNDICE DE TABLAS

Tabla 1.- Equipos Base y Configuración.....14

1. INTRODUCCIÓN

Hoy en día, el cloud computing ha revolucionado la industria de la tecnología. Muchos de los avances más importantes como renderización de video y simulaciones de fluidos o física cuántica requieren de gran cantidad de poder de cómputo. El pronóstico de crecimiento para el cloud computing es prometedor, Forbes aproxima que en el 2016 el gasto público en infraestructura comprende 38 mil millones de dólares americanos entre servicios de hardware y software. El pronóstico de crecimiento para el 2026 es de 173 mil millones de dólares americanos (Columbus, 2016). Las aplicaciones para una nube son extensas, desde una nube de storage hasta nodos de virtualización que ofrecen PaaS. Los gigantes más grandes de tecnología ya han optado por estas alternativas tales como Netflix y Twitter. Los clústers de virtualización han tenido un auge en los últimos años y su popularidad ha aumentado dado su gran capacidad de escalabilidad, bajo costo y alta capacidad de control. Adicionalmente, provee de seguridad de riesgo al hacer una inversión rentable y reduce gastos en administración y seguridad.

El crecimiento en el área se consolidó con la formación de empresas y servicios de gigantes tecnológicos tales como Amazon y AWS (Amazon Web Services), Azure de Microsoft y Google Cloud Platform.

Por otro lado, los desechos informáticos se convierten en un problema al día de hoy. Se estima que alrededor de 9.4 millones de toneladas de basura electrónica se desecha solamente en Estados Unidos (Button, 2016). Tomando en cuenta que la ley de Moore se ha cumplido en los últimos 15 años (Simonite, 2016), claramente puede verse que una gran porción de tecnología todavía usable se la ha catalogado como basura. A pesar de que su capacidad de cómputo todavía es útil así como sus componentes, pero inútil ante los ojos de la tecnología que se renueva cada 18 Meses (Moore, 1975).

Con el fin de dar una solución eficiente a la problemática de los desperdicios electrónicos de una manera eficiente, se propone un clúster con capacidades de virtualización orientado hacia estudiantes y profesores con fines de investigación y didácticos así como también como un medio para realizar emprendimientos con bajos recursos.

1.2Antecedentes:

Los clústers de Virtualización son componentes ya existentes en el mercado y actualmente gozan de una gran aceptación. Para el caso actual se puede tomar en cuenta casos de éxito tales como Amazon Web Services el cual ha registrado crecimientos abismales desde el 2009. A pesar de esto, el problema de procesar grandes cantidades de información que se ha presentado desde principios de siglo se ha referido a software. Los sistemas actuales que permiten virtualización son muy pocos, entre los más nombrados de Hipervisores de tipo 1 son XEN Server, VSphere ESX-i y Red Hat Virtualization. Puesto que ninguno de los anteriores ofrece una solución válida para el problema, ya que presentan problemas con la heterogeneidad de software. Esto se evidencia en la lista de requerimientos básicos de cada software, donde especifica que el soporte para hardware heterogéneo es limitado(VMware, 2016).

Dentro de los hipervisores de Tipo 2 más famosos están HyperX de Microsoft y Oracle Virtual Network. A pesar de esto, ninguno ofrece un rendimiento deseado para aprovechar máquinas cuyo soporte oficial esté por terminar. Dado el caso anterior, OpenStack es el único sistema que ofrece Virtualización de hipervisor de tipo 1 y al mismo tiempo provee de interconexión entre servidores heterogéneos.

1.3Justificación:

El proyecto consiste en la creación de un sistema informático y un proyecto práctico en el cual se reciben computadoras que se encuentren en estado funcional pero sus donadores están dispuestos a entregarlas a cambio de una suscripción en el cluster por un tiempo definido previamente. De esta manera, se reducen los desechos electrónicos existentes y se les provee de un mejor uso con el fin de brindar un servicio a estudiantes que necesitan máquinas virtuales a plataformas con usos didácticos así como también funciona para dar a profesores capacidad de cómputo en materia de investigación en diferentes áreas. Es importante recalcar que de esta manera se logra que el usuario disponga de recursos de alta disponibilidad a pesar de que el hardware sea de bajo costo y se reduce la cantidad de desechos electrónicos.

Con un sistema con costos bajos se puede mantener servicios corriendo la mayor cantidad de tiempo posible en todos los nodos y por lo tanto permitiendo que operaciones que tardan mucho tiempo puedan ser realizadas tales como modelado 3D, cálculos multivariables, renderizado de video entre otros. Adicionalmente, mantener recursos siempre disponibles es un herramienta didáctica muy práctica ya que hay recursos compartidos que necesitan de un ambiente de trabajo como bases de datos compartidas en MySQL.

La solución para los desechos electrónicos ha sido el reciclaje en el pasado donde se remueven los metales valiosos de los componentes tales como oro y plata. Con el material sobrante se crean figuras o esculturas con fines artísticos entre otros lo cual no refleja necesariamente un uso óptimo de tecnología. A pesar de que muchos de estos desechos se encuentran en un periodo conocido como EOL o fin de vida, todavía pueden ser utilizados para procesamiento y carga de datos.

1.4Objetivos específicos:

Crear una infraestructura donde varios nodos de cómputo pueden ser agregados a un clúster. Este clúster estará basado en OpenStack y cada nodo que es agregado mediante el instalador que se creó tendrá una instalación desatendida en la red. Para esto crearemos un diagrama de nodos y se define la estructura de la red local. Además, debemos crear un script para la auto instalación del sistema operativo y un script para facilitar la conexión entre las máquinas virtuales y sistemas que se encuentren fuera de la red local.

Adicionalmente, varios factores se deben tomar en cuenta una vez que sea realizada la infraestructura. Esto incluye detalles tales como tiempo de vida promedio, cantidad de nodos disponibles y carga total del sistema que deben ser medidos para obtener resultados confiables. De esta manera, se pueden generar esquemas que garanticen que el sistema tiene un periodo de vida largo. Una vez establecidos estos esquemas se procede a definir la estructura de la infraestructura así como el funcionamiento e implementación.

Para evaluar el funcionamiento del sistema, se explica las tareas que se cumplieron durante el desarrollo de la infraestructura y creación de los servidores. Con esta información, se puede determinar si se cumplieron con éxito los elementos para dar a lugar

a una infraestructura funcional a largo plazo. Igualmente, se realiza una evaluación de los sistemas que se están ejecutando al momento para determinar la funcionalidad correcta.

En la siguiente sección, se describe detalladamente cómo está estructurado el sistema, así como sus componentes y funcionalidad.

2. ESTRUCTURA DEL CLÚSTER

La estructura del clúster tiene varios elementos fundamentales. OpenStack tiene componentes que hacen más fácil la separación de tareas y facilita la redundancia entre nodos esclavos o maestros. La estructuración base para el sistema se organizó de la siguiente manera. En el nodo controlador, se instalaron los componentes base Keystone, Neutron, Glance y Horizon como componente adicional. De el lado de los nodos de cómputo, se instalaron los componentes Nova y Neutron. Todos estos componentes serán descritos así como su funcionalidad más adelante. Es importante comprender que se esta estructura se creó con el fin de hacer el sistema lo más ágil y funcional posible con los recursos disponibles al momento.

Así también es válido mencionar que para el sistema se utilizaron 4 equipos base. Dentro de los equipos que conforman la base o core se encuentran:

TIPO DE NODO	SISTEMA OPERATIVO	PROCESADOR	MEMORIA
Controlador	Ubuntu 16.04	Intel Xeon 3 Ghz	2 GB
Cómputo	Ubuntu 16.04	Intel Pentium 4 2.8 Ghz	1GB
Cómputo	Ubuntu 16.04	Intel Celeron 2.3 Ghz	1 GB
Cómputo	Ubuntu 16.04	Amd Opteron 2.4 Ghz	16 GB

Tabla 1.- Equipos Base y Configuración.

A partir de la Tabla 1, se puede ver que la configuración de los nodos de cómputo es la misma en cuanto a sus componentes. Para la sección del nodo controlador solo existe uno de cada componente.

2.1Descripción:

Para cada componente existen cuatro nodos de cómputo nombrados anteriormente en la Tabla 1 y para cada nodo controlador tan solo uno. En la Figura 1, se puede apreciar los componentes así como un ejemplo de la relación. En las secciones 2.1.1 a la sección 2.1.5 se dará detalle de los componentes.

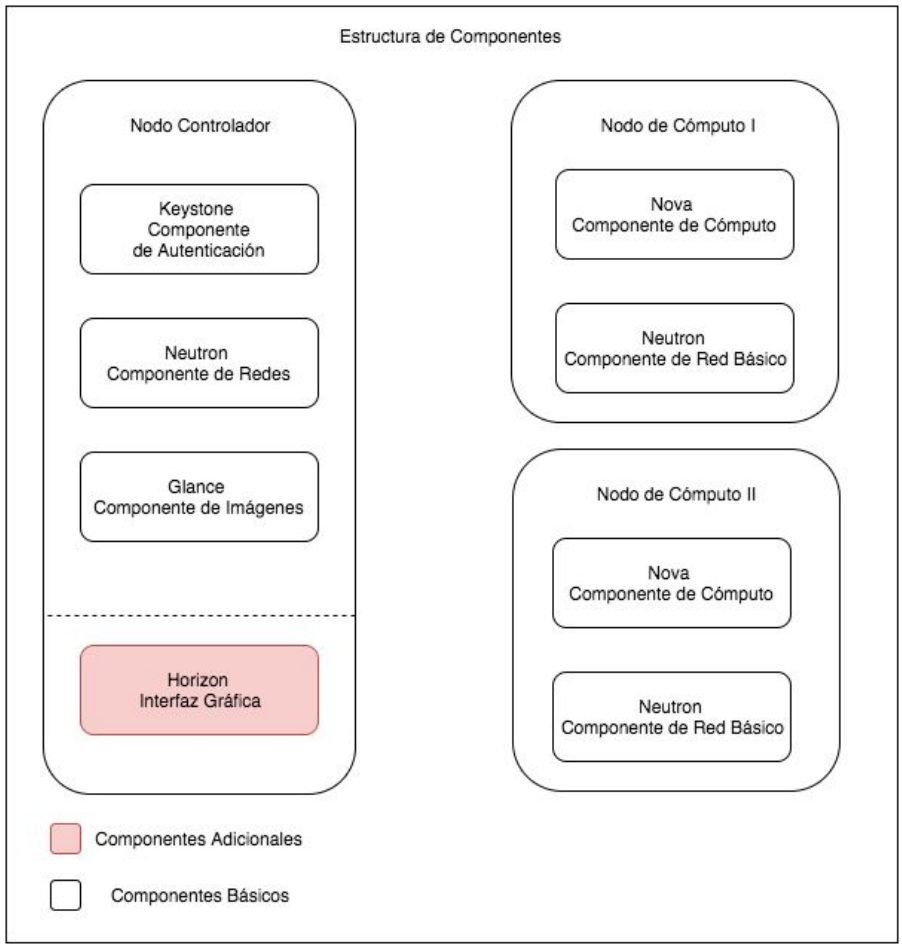


Figura 1.- Estructura de sistema y componentes.

Es importante mencionar que Keystone se encarga de la autenticación de los nodos de cómputo, Neutron de la interconexión de red y Glance funciona como servidor de imágenes para que se levanten en los nodos de cómputo. El acceso al sistema se realiza mediante Horizon que provee de interfaz gráfica.

2.1.1 Módulo Nova.

El módulo Nova se encarga de de la gestión y control de las máquinas virtuales creadas dentro de los sistemas. Este debe ser configurado en cada nodo de cómputo, ya que gracias a este se realiza la carga que se distribuye en el procesador de cada nodo de cómputo. Es válido mencionar que para este proceso se utiliza QEMU en procesadores con arquitecturas que no soporta Virtualizacion por hardware, por ejemplo VX-t. En caso de que si exista soporte se procede a realizar aceleración de virtualización por hardware mediante KVM. Es importante mencionar también que otros modelos de ejecución o virtualización pueden ser ejecutados tales como LXC, Docker, Baremetal entre otros (OpenStack,2016).

2.1.2 Módulo Neutron.

En la sección de Networking o Redes se encuentra el manejador de eventos y comunicador entre el sistema que se llama Neutron. Este debe ser instalado tanto en los nodos de control como los nodos de cómputo ya que este es el fundamento para la ejecución de máquinas virtuales, autenticación y copia de imágenes.

Neutron está basado en un componente de mensajes que en este caso fue utilizado como rabbid, que se encarga de entregar mensajes entre los distintos nodos. Adicionalmente, Neutron se encarga de manejar y crear las interfaces de red entre las máquinas virtuales creadas y de la entrega y recepción de paquetes dependiendo cual de las

redes se usen entre las redes locales o de servicio las cuales serán explicadas en detalle más adelante.

2.1.3 Módulo de Glance.

El módulo de Glance se encarga de una de las secciones vitales del cluster de virtualización: El manejo de imágenes a ser cargadas se realiza normalmente en un servidor separado. Por razones de costos para el caso particular se utilizó en el mismo nodo controlador. El componente de Glance almacena las imágenes las cuales son transferidas a los otros nodos cuando se debe crear una nueva instancia o una nueva máquina virtual.

Este componente tiene algunos tipos de archivo soportados por defecto tales como QCOW2 o IMG entre otros que facilitan la creación de nuevas instancias. Adicionalmente se puede ampliar el espectro de imágenes soportadas al instalar módulos adicionales (OpenStack, 2016).

2.1.4 Módulo Keystone.

En la sección del módulo Keystone, el módulo genera tokens para la verificación y autenticación del todos los componentes del sistema (Kupidura, 2013). Cada componente tiene sus propios tokens generados y al momento de necesitar comunicarse entre si utiliza el token previamente generado para autenticarse y permitir el traslado de mensajes o comandos.

Para el caso del sistema, se utiliza el mecanismo dentro del servidor controlador a pesar de que es recomendado para la producción separar el componente dentro de otro servidor para garantizar la seguridad de todo el sistema.

2.1.5 Módulo Horizon.

Finalmente, en el último módulo instalado dentro de la configuración es Horizon el cual provee de una interfaz gráfica al usuario mediante un servidor HTTP. Concretamente se utiliza Apache versión 2 para levantar la interfaz gráfica dentro del lado de los clientes. Para todo el acceso, se utiliza VNC lo cual posibilita el complemento de VNC + HTML que permite el control remoto de las máquinas virtuales a través de la interfaz gráfica.

2.2Infraestructura de los componentes:

La manera en la que trabajan los componentes entre sí esta diseñada de una manera en la que se manejen de manera eficaz los recursos como es demostrado en la sección 5.1, siguiendo una estructura fija de diagramación los componentes interactúan entre sí pasando mensajes hasta que se cumpla la función de lanzamiento de una instancia.

El componente de Neutron es el encargado de entregar los mensajes cifrados entre todos los componentes del sistema pasando por la red local del sistema. Para el proceso de cifrado y autenticación, el componente de Keystone genera tokens basados en la identidad de cada componente y usuario. Posteriormente, pasa con el proceso de validación del componente Horizon para que este pueda obtener todos los datos del sistema. Una vez que Horizon se valida en el sistema obtiene los datos de cada componente enviando queries simples a las bases de datos para conocer el estado del sistema y los recursos disponibles para presentarlos en la interfaz gráfica.

Es así que establecida la interfaz gráfica el usuario puede proceder a crear una nueva instancia. Para este proceso, Horizon se encarga de enviar los comandos a Neutron que se encarga de distribuirlos a Glance y Nova cuando se autentiquen con Keystone. Una vez llegado el mensaje a Nova de una nueva instancia, este le pide a Glance que le envie la

imagen base para la instancia. Cuando el servidor de Nova ya tiene la imagen por parte de Glance procede a lanzar una nueva instancia usando los componentes de nova.

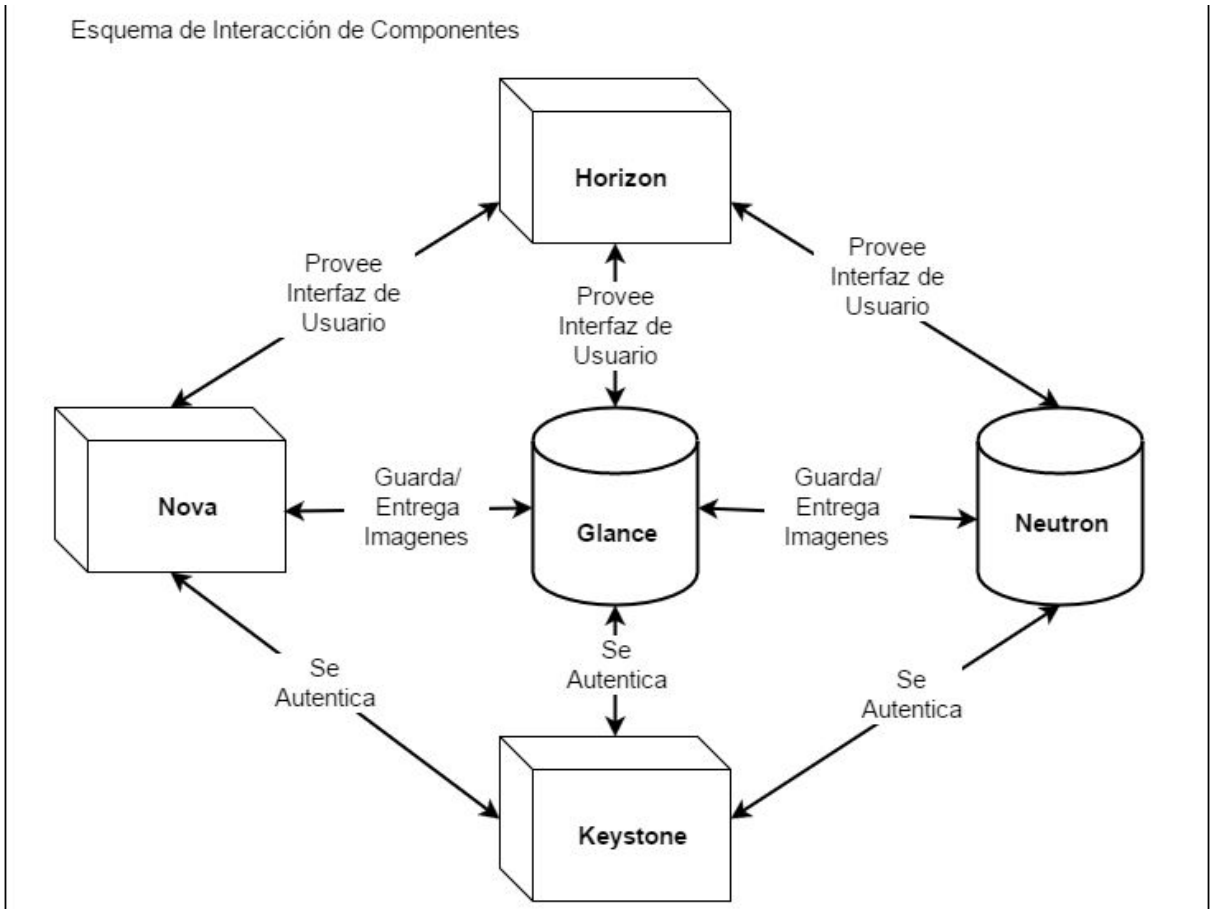


Figura 2. Esquema de interacción entre componentes.

Como se aprecia en la Figura 2, todos los componentes interactúan entre sí con el fin de lanzar una nueva instancia. Es válido mencionar que así como OpenStack provee de módulos opcionales como Horizon para una interfaz gráfica, también tiene otros módulos adicionales con el fin de aumentar la experiencia del usuario final y obtener métricas para el administrador del clúster(Zitzman, 2014).

2.3Esquema de red:

Dentro del esquema de red, es importante mencionar que existen varias configuraciones posibles para ésta red, en este caso se optó por un esquema donde cada nodo dispone de dos NICs. Esto se hace con el fin de dividir la red administrativa y la red a la que tienen acceso las máquinas virtuales. En producción esto se hace con el fin de otorgar IP Públicas a las máquinas virtuales para que los administradores de las máquinas y usuarios tengan acceso a éstas. Para nuestro caso particular esto se lo hace con el fin de no permitir a estudiantes o docentes acceso a la red interna y mantener separación de recursos en caso de intrusión o en términos de disponibilidad del sistema.

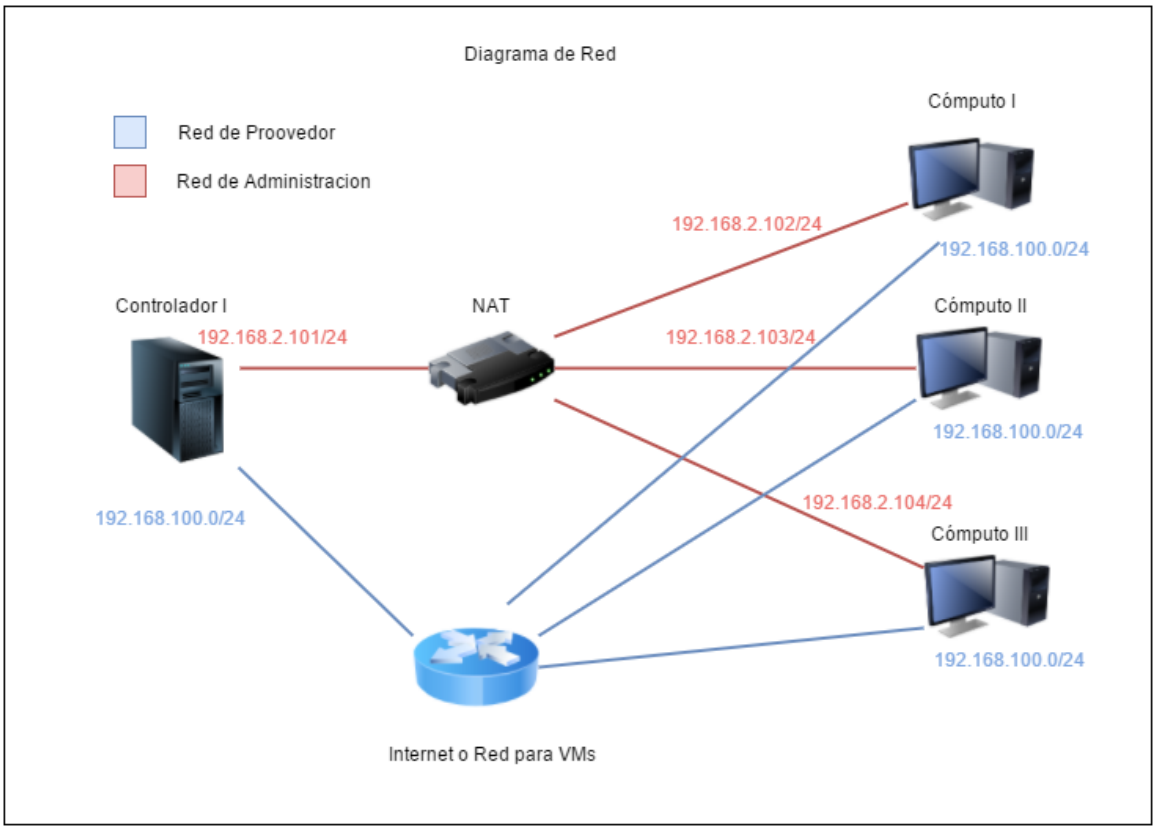


Figura 3.-Infraestructura de Red del Sistema.

La conexión dentro del Cluster mantiene a todos los nodos interconectados a el primer switch en la red de Administración, para las máquinas virtuales se les asigna una IP

en el pool de la red de VMs así como se puede ver en la Figura 3. A pesar de que este sea el esquema estipulado la conexión con un solo NIC se puede lograr creando un NIC virtual y haciendo que este trabaje en modo bridge (Marlin, 2014). A pesar de esto se necesita 2 VLANs en el enrutador para que el segundo NIC encuentre su segmento de red.

3. INSTALACIÓN DE NODOS DE CÓMPUTO E INSTANCIAS

Para facilitar la instalación de los nodos de cómputo, se creó un script a ser agregado a la imagen base de ubuntu server 16.04. Este script se encarga de instalar y configurar los componentes necesarios para que el nodo donde sea instalado funcione como un componente adicional después de la instalación del sistema operativo.

3.1Descripción:

La instalación de un nodo de cómputo es simple. Instalamos chronyc, un cliente de sincronización de horario para poder establecer a el nodo controlador como el único manejador de horario. Después de eso se instalan los componentes Nova y Neutron, para más tarde proceder a la configuración de los distintos archivos que hacen que el sistema funcione acorde a la especificación de Neutron y Nova del controlador. Una vez que el sistema está configurado se procede a la verificación por parte del nodo controlador que debe ser realizada manualmente por el administrador.

3.1.1 Pasos de configuración realizados por el script de instalación.

Primero se instala el componente Nova, después se pasa después a agregar los nodos existentes al archivo de configuración. Es importante también agregar el servidor controlador al archivo de enrutamiento. Después de esto pasamos a editar la lista de servidores de sincronización de hora en el archivo de configuración de chrony. En este

archivo debemos especificar el servidor que dará las actualizaciones de tiempo por lo cual comentamos todos los pools existentes y agregamos al nodo controlador.

Reiniciamos el servicio y pasamos a la edición del archivo de configuración de Nova, donde tan solo debe copiar el contenido de cualquier otro nodo de cómputo existente, tan solo modificando la IP en el archivo. Procedemos a verificar si el nodo tiene capacidades de virtualización de hardware. Esto nos devuelve un entero que en caso de ser mayor a uno significa que soporta virtualizacion por hardware por lo tanto no hay que hacer nada más, en caso de que el entero sea menor a uno debemos definirlo en el archivo de configuración de Nova. Esto concluye la configuración de Nova para lo cual pasamos a configurar Neutron.

Para la instalación del componente Neutron se procede instalar desde la terminal. Ya que solo se necesita el componente mínimo para lograr una conexión con Neutron del controlador la instalación es mínima y se debe editar el archivo de configuración, una vez más copiando los contenidos de una instalación válida en un nodo de cómputo funcional. Finalmente editamos el archivo de configuración de red con el fin de establecer la red en la cual las máquinas virtuales interactúan. Para este caso se utiliza un adaptador adicional en cada nodo el cual tiene acceso a una red completamente separada de la red local por la que interactúan los nodos. En caso de no disponer de 2 NICs por computadora se puede crear un NIC virtual y hacerlo trabajar en modo bridge con el NIC físico. A pesar de eso esto significa crear una VLAN dentro del router o manejador de red local. Finalmente se procede a reiniciar el nodo y la instalación quedará completada.

3.1.2 Descripción de rutas y código para instalación de nuevo nodo.

Instalamos el servicio de chrony y Nova.

- `sudo apt-get install nova-compute && apt-get install chrony`

Editamos el archivo de enrutamiento.

- `/etc/hosts`

Agregamos la ruta al nodo controlador.

- *`192.168.2.101 Controlador Xeon`*

Editamos el archivo de configuración de chrony.

- `/etc/chrony/chrony.conf`

Agregamos el parámetro para la sincronía al nodo controlador.

- `iburst server ControladorXeon`

Editamos el archivo de configuración de Nova.

- `/etc/nova/nova.conf`

Agregamos la IP asignada por el administrador.

- `ip_addres = x.x.x.x`

Definimos mediante el comando si el hardware soporta virtualizacion.

- `egrep -c ‘ (vmx|svm)’ /proc/cpuinfo`

En caso de que no lo soporte debemos especificarlo mediante el siguiente comando en el archivo de configuración de Nova.

- `[libvirt] \ virt_type = qemu`

Pasos para la configuración de Neutron

Instalamos el cliente básico de Neutron.

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

Editamos el archivo de configuración copiandolo de un nodo que ya sea funcional.

- `/etc/neutron/neutron.conf`

3.1.3 Script para conexión remota a través de SSH *Forwarding*.

Dentro de la sección de conexión una vez establecidos los nodos se debe crear una conexión remota con un servidor externo a menos que se disponga de un pool de IPs para poder utilizar. Dado que las máquinas virtuales generan una conexión nueva dentro de la red Provider, esto permite dos opciones. La primera opción consiste en crear un túnel remoto a un servidor público o VPS con el fin de que el puerto de conexión ya sea 22 (SSH) o 5900(VNC) sea público para que el usuario pueda acceder a este desde donde esté.

La funcionalidad de esta opción consiste en un script que elige un puerto aleatorio disponible en el servidor externo público, después con la llave privada del servidor crea un túnel con el puerto 22 en caso de un sistema basado en UNIX y lo lleva a un puerto aleatorio. Se le informa al usuario del puerto elegido mediante un mensaje en el log de inicialización del servidor. Si se desea mapear más puertos se puede hacer mientras este puerto esté disponible. Para el caso se utiliza un VPN que será válido por un año pero fácilmente puede ser reconfigurable de acuerdo a las necesidades del administrador.

La segunda opción es implementar un sistema de red local a través de un enrutador y switches con el fin de que los usuarios se puedan conectar directamente a la red de máquinas virtuales. Esto se puede lograr creando una conexión de túnel utilizando OpenVPN hacia el enrutador o una máquina que funcione como enrutadora de conexiones de VPN.

En nuestro caso se optó por implementar la primera opción que consiste en utilizar el script de puertos aleatorios. Se decidió hacerlo así porque el programa piloto requiere de

bajos recursos, por lo tanto agregar nodos con un mínimo de dos NICs podría causar un impacto negativo en la cantidad de nodos o en recursos financieros a ser invertidos.

4. FUNCIONAMIENTO DE MÁQUINAS VIRTUALES Y ACCESO

Para el acceso y administración al componente Horizon se debe acceder mediante un túnel existente y una configuración existente a través de OpenVPN. Una vez que el acceso mediante el archivo .ovpn es realizado correctamente se debe ingresar a la interfaz gráfica de Horizon. Para hacerlo entramos desde la URL **`http://192.168.2.101/horizon/`** . Aquí los datos de acceso serán generados por los administradores del cluster de acuerdo a políticas de seguridad y las contraseñas creadas se indicarán en un documento oficial. Con estos datos se accede al panel de administración.

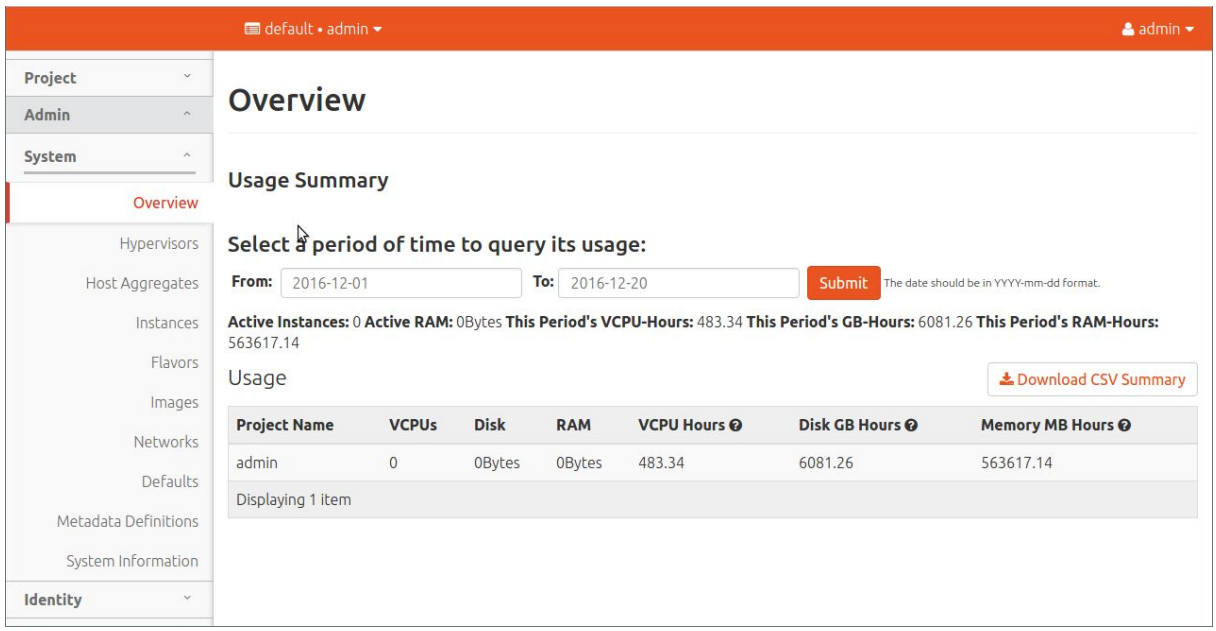


Figura 4. Página principal del componente de administración.

Como se observa en la figura 4,la interfaz gráfica levantada por Horizon permite verificar datos de funcionamiento del clúster así como estadísticas básicas sobre tiempo de uso de

procesador, tiempo de uso de memoria y de disco. Es importante mencionar que el panel de usuario tiene una menor cantidad de opciones y el manejo es mucho más limitado. Así también, desde la interfaz grafico de horizon se pueden realizar varias operaciones como: creación de flavors, subida de imágenes en distintos formatos, creación de nuevos usuarios y manejo de instancias. Todo esto también está disponible dentro de la terminal de bash en el nodo controlador.

4.1 Administración de controlador y opciones adicionales:

Primeramente, el componente Horizon es capaz de interactuar con el API de todos los componentes, es así que puede crear instancias desde el panel de administración de una manera simple. Al lanzar el creador de instancia se solicitan pocos detalles como nombre de instancia, flavor a ser utilizado y imagen base. Detalles adicionales tales como llave para administración remota y scripts de inicio también están disponibles.

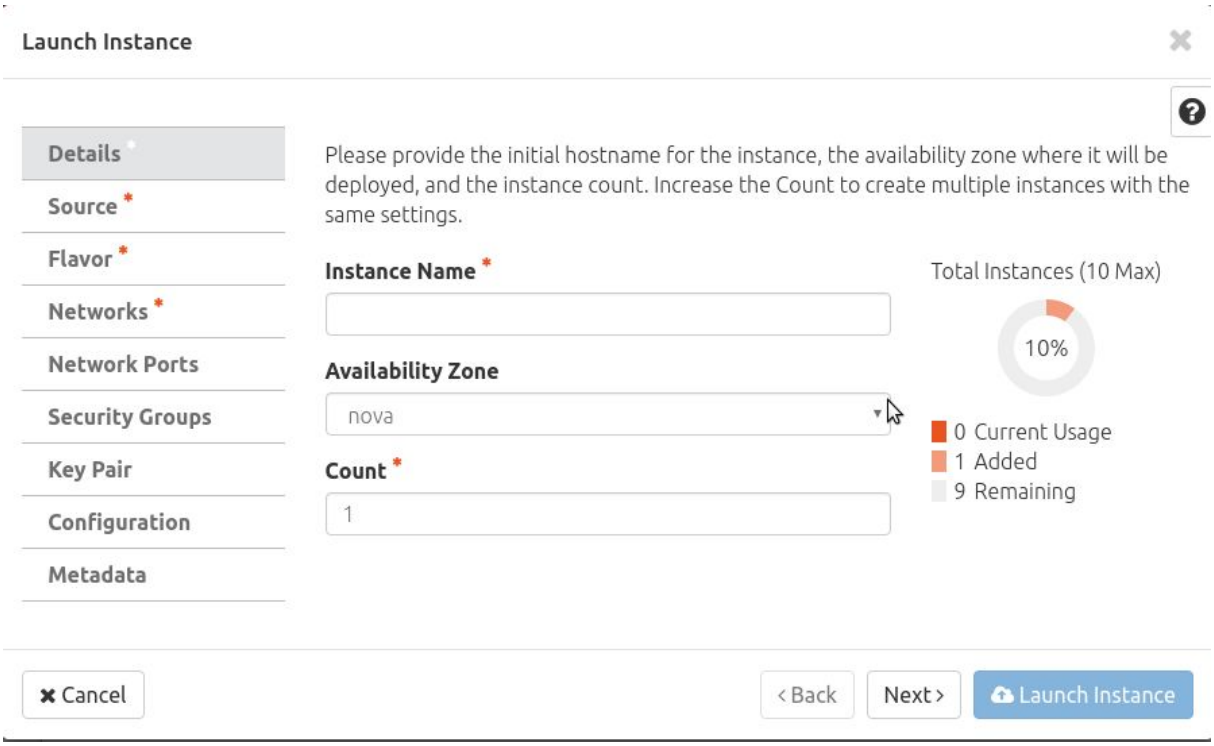


Figura 5. Interfaz de creación de nueva instancia.

Tal como se puede apreciar en la Figura 5 los estados de creación de una instancia pasan entre: spawning, initialization, running, stopped. Estos estados determinan si la máquina virtual está lista para ser ejecutada o permanece en proceso de inicio. Adicionalmente existen estados de error para los cuales se puede consultar el archivo de logs */var/logs/nova.log* . Esto se realiza en el nodo controlador y ayuda a determinar la causa del error.

4.2 Usos comunes para Administradores y Usuarios:

Dentro de los casos más generales para administración de cluster se encuentra la combinación de nodos para generar nodos con mayor potencia y permitir a administradores realizar tareas complicadas de manejo de servidores. Muchas veces un administrador de red debe trabajar con una base de datos la cual debe ser compartida entre varias aplicaciones y tener una alta disponibilidad. En este caso el administrador puede levantar máquinas virtuales con componentes como ironic que permitan funcionalidad en conjunto. Esto garantiza que altas cargas con varios clientes pueden ser soportadas.

Además, los usuarios pueden utilizar sistemas para mantener servidores de alta disponibilidad y cargas relativamente bajas. Por ejemplo un usuario que necesita tener una página web activa con el objetivo de obtener métricas, en este caso el cluster garantiza alta disponibilidad. También puede ser útil al momento de realizar pruebas que necesitan de largos periodos de actividad. Por ejemplo realizar un monitoreo de accesos en logstash o entrenamiento en técnicas de defensa ante ataques de DDoS.

4.2.1 Ejemplos de funcionamiento de usuario.

Como punto de ejemplo se puede crear una maquina virtual para que un estudiante pueda comprender un entorno de desarrollo básico de páginas web y base de datos. En este caso básico el usuario deberá proceder a elegir la imagen creada por el administrador que contiene preinstalada la base de datos y el servidor de apache en versión 2. Una vez la instancia es creada el estudiante deberá hacer port forwarding del puerto 80 hacia un puerto aleatorio remoto en el servidor público para el caso llamado VPS.

De manera similar si necesita crear una instancia de ambiente de desarrollo básico, debe elegir la imagen generada por el administrador que contiene las herramientas básicas de desarrollo. Una vez levantada la máquina virtual se debe proceder a crear una conexión mediante VNC haciendo port forwarding del puerto 5900 a un puerto aleatorio en el VPS. El proceso es simple ya que está automatizado con un script en el cual solo se necesita cambiar el puerto el cual se desea que se haga el forwarding.

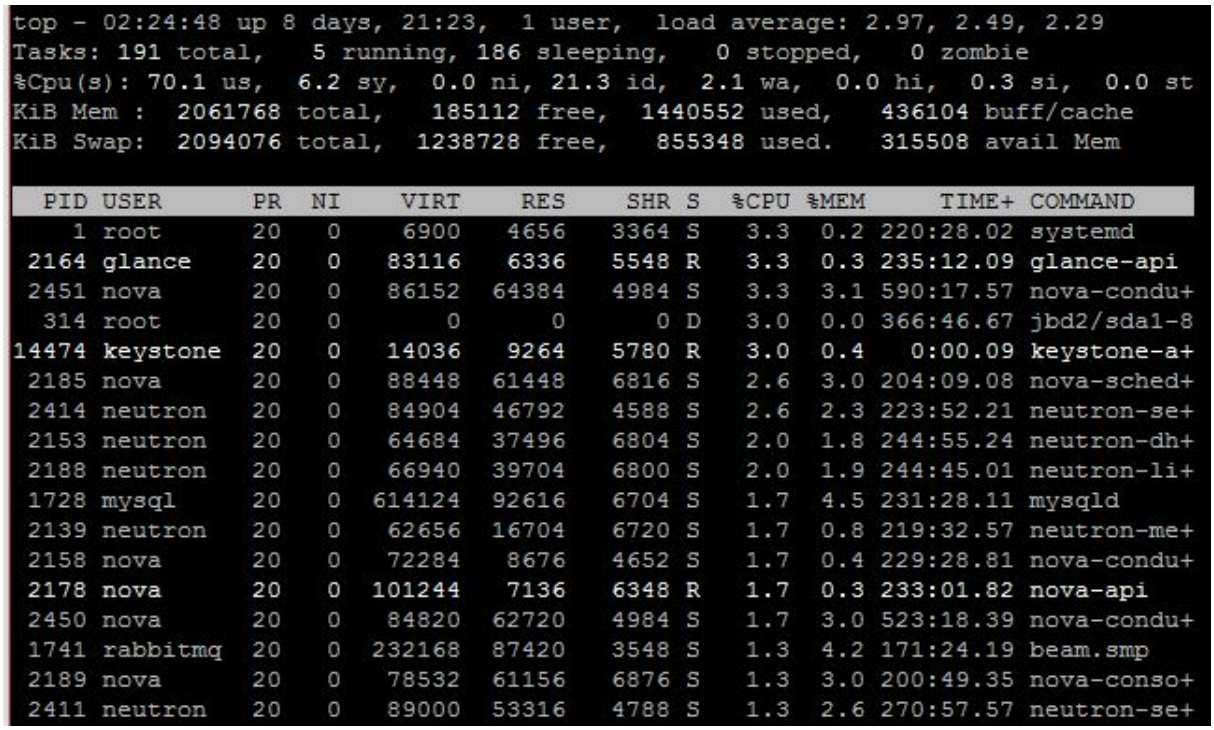
4.2.2 Casos de modificación de administrador

Para el administrador el proceso de instalación de paquetes en un ambiente básico es simple. Se debe modificar el archivo de inicialización de la imagen de nube, para el caso general de un sistema Ubuntu. En este caso se debe modificar el archivo `/etc/cloud/cloud.cfg .`Este contiene detalles como paquetes para ser instalados post la inicialización del sistema y su configuración, para más detalles sobre su modificación se puede leer el MAN page de Cloud-init (Cloudinit, 2016). La modificación se puede realizar utilizando el paquete guestfish para Linux.

5. RESULTADOS Y RECOMENDACIONES A FUTURO

5.1 Estado de funcionamiento del sistema:

Se mide el rendimiento de los nodos del sistema con o sin carga para encontrar si el sistema está en capacidad de soportar el sistema. Para medir la carga del procesador se utiliza la herramienta TOP que nos entrega datos de las cargas para el nodo controlador durante los últimos 1, 5 y 15 minutos para tener una idea de la carga total del sistema.



```
top - 02:24:48 up 8 days, 21:23,  1 user,  load average: 2.97, 2.49, 2.29
Tasks: 191 total,   5 running, 186 sleeping,   0 stopped,   0 zombie
%Cpu(s): 70.1 us,  6.2 sy,  0.0 ni, 21.3 id,  2.1 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem :  2061768 total,   185112 free,  1440552 used,   436104 buff/cache
KiB Swap:  2094076 total,  1238728 free,   855348 used.   315508 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
    1 root      20   0    6900   4656   3364 S   3.3  0.2 220:28.02 systemd
 2164 glance    20   0   83116   6336   5548 R   3.3  0.3 235:12.09 glance-api
 2451 nova      20   0   86152  64384   4984 S   3.3  3.1 590:17.57 nova-condu+
  314 root      20   0       0      0      0 D   3.0  0.0 366:46.67 jbd2/sda1-8
14474 keystone  20   0   14036   9264   5780 R   3.0  0.4   0:00.09 keystone-a+
 2185 nova      20   0   88448  61448   6816 S   2.6  3.0 204:09.08 nova-sched+
 2414 neutron   20   0   84904  46792   4588 S   2.6  2.3 223:52.21 neutron-se+
 2153 neutron   20   0   64684  37496   6804 S   2.0  1.8 244:55.24 neutron-dh+
 2188 neutron   20   0   66940  39704   6800 S   2.0  1.9 244:45.01 neutron-li+
 1728 mysql     20   0  614124  92616   6704 S   1.7  4.5 231:28.11 mysqld
 2139 neutron   20   0   62656  16704   6720 S   1.7  0.8 219:32.57 neutron-me+
 2158 nova      20   0   72284   8676   4652 S   1.7  0.4 229:28.81 nova-condu+
 2178 nova      20   0  101244   7136   6348 R   1.7  0.3 233:01.82 nova-api
 2450 nova      20   0   84820  62720   4984 S   1.7  3.0 523:18.39 nova-condu+
 1741 rabbitmq  20   0  232168  87420   3548 S   1.3  4.2 171:24.19 beam.smp
 2189 nova      20   0   78532  61156   6876 S   1.3  3.0 200:49.35 nova-conso+
 2411 neutron   20   0   89000  53316   4788 S   1.3  2.6 270:57.57 neutron-se+
```

Figura 6.- Carga de Nodo Controlador.

Como se puede observar en la figura anterior con todos los componentes necesarios corriendo del lado del nodo controlador la carga promedio se establece en 2.5. Lo cual quiere decir que el procesador al tener 4 núcleos se mantiene en una carga relativamente

baja. Esto nos demuestra que el nodo controlador a pesar de tener varios servicios corriendo continua con una carga que puede aguantar más procesos o componentes.

5.2 Evaluación de nodos de cómputo:

Así como el nodo controlador tiene una carga al manejar los nodos de cómputo, cada nodo de cómputo debe soportar una carga mínima para poder entregar información a los componentes que integra como Neutron y Glance. Es importante mencionar que la carga es relativa a la cantidad de núcleos en el procesador. Para el caso del nodo de la Figura X, este tiene cuatro núcleos.

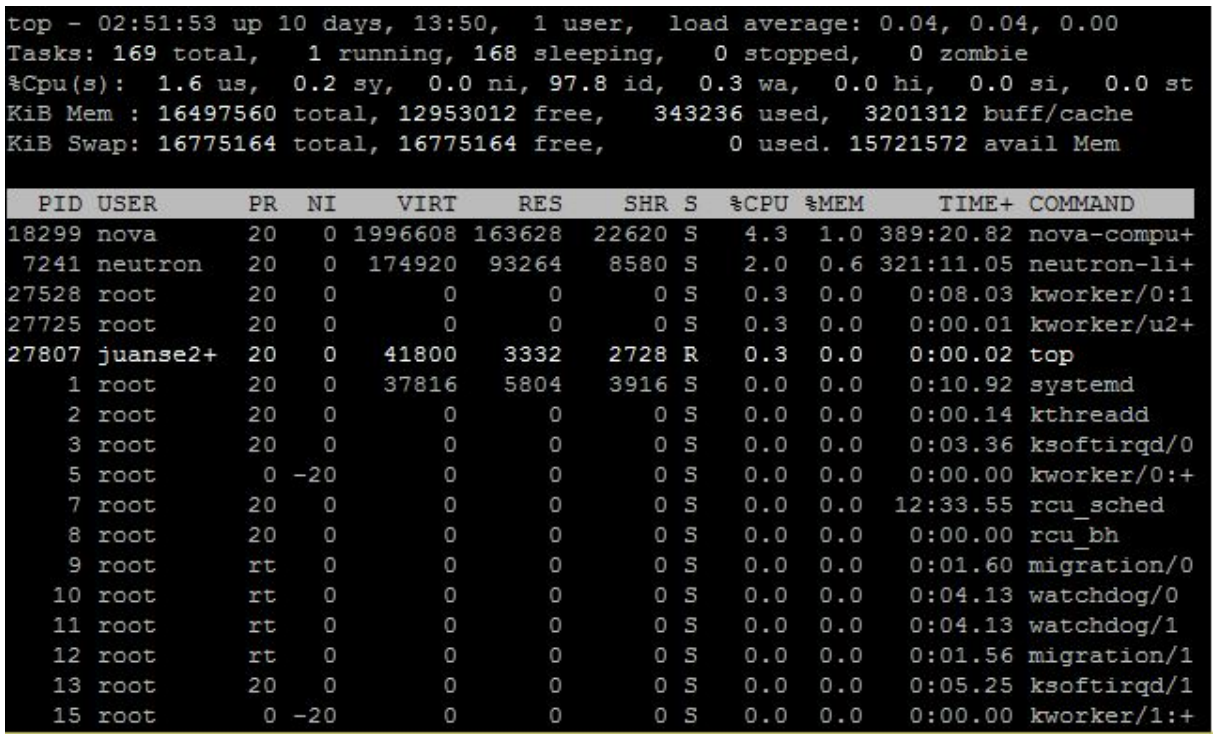


Figura 7.- Carga de nodo de Cómputo

Como se aprecia en la imagen los nodos de carga sufren de un estrés menor, a pesar de que debe aumentar la carga por cada máquina virtual, se entiende que es relativa a la capacidad del procesador y no excede el 3.96 para que no exista congelamiento o *queueing*.

5.3 Benchmark del sistema:

Para constatar la relación entre poder de procesamiento en la máquina nativa y performance en la máquina virtual se realizó benchmarking en las plataformas. Durante el proceso se utilizó la aplicación Bonnie++ que mide datos acerca del rendimiento de sistema en términos de sistema operativo. Esto incluye memoria, carga de procesador entre otros. Esta prueba se realizó en el nodo de cómputo con procesador AMD y en el nodo de cómputo celeron.

El test consiste en medir el performance, para ejecutamos el test en cada nodo de manera nativa y después creamos una nueva instancia en Ubuntu 16.04 correspondiente a la arquitectura del procesador. El objetivo es comparar el rendimiento de la máquina de manera nativa y de la instancia en términos de poder de procesamiento. A partir de esto se pudieron inferir los siguientes resultados.

Celeron

Version 1.97		Sequential Output						Sequential Input				Random Seeks			Sequential Create						Random Create					
	Size	Per Char		Block		Rewrite		Per Char		Block				Num Files	Create		Read		Delete		Create		Read		Delete	
		K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	/sec	% CPU		/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU
ComputoCeleron	3G	290	93	45621	22	17956	9	791	69	52238	13	166.1	13	16	14332	72	+++++	+++	24007	63	15372	73	+++++	+++	23404	65
	Latency	41605us		1974ms		1229ms		67481us		126ms		1296ms		Latency	3567us		1929us		2131us		4024us		1131us		5841us	

Amd

Version 1.97		Sequential Output						Sequential Input				Random Seeks			Sequential Create						Random Create					
	Size	Per Char		Block		Rewrite		Per Char		Block				Num Files	Create		Read		Delete		Create		Read		Delete	
		K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	/sec	% CPU		/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU
ComputoAMD	32216M	800	97	115615	21	57759	12	3386	94	151771	16	225.2	13	16	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++
	Latency	16220us		1220ms		5677ms		23170us		145ms		449ms		Latency	202us		1132us		649us		173us		57us		374us	

Como se puede Apreciar en la Figura 8 el rendimiento de las instancias es inferior al rendimiento de la máquina nativa. Esto se puede dar ya que la máquina nativa tiene acceso a todos los componentes del sistema mientras que a la instancia tan solo se le asignó una porción de todo el sistema. Aun así en el sistema Pentium Celeron se puede apreciar que el

rendimiento es similar ya que se le otorgaron características similares a la instancia y a la máquina real.

Así podemos concluir que la afectación del rendimiento es mínima y el impacto es relativo a la cantidad de recursos otorgados al sistema. Es importante mencionar también que el sistema operativo y versión del kernel pueden afectar en el resultado. A pesar de que los tests se realizaron intentando aislar los entornos para tener métricas más exactas esto no se logró al 100%.

6. CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones:

- Para establecer los nodos básicos es importante entender que tipos de imágenes van a correr en el clúster y en base a esto elegir las arquitecturas adecuadas. Si la carga de un nodo es superior a la de otro debido a que la aplicación se ejecuta en la arquitectura incorrecta pueden haber serios problemas de carga.
- Es importante comprender que muchos de los equipos utilizados para el programa tienen tiempos de vida mucho menores que uno equipo nuevo. Al utilizar equipos de casa como servidores es muy probable que estos lleguen a dañarse con facilidad y provocan errores por lo tanto es importante tener un servidor fiable para storage o levantar imágenes con la posibilidad de que la información se pierda.
- Establecer políticas de uso para cada usuario de manera que los estudiantes tengan una cantidad de recursos limitados y no utilizan el clúster más de lo necesario, ya que

si todos los usuarios levantan más nodos de los que deberán puede haber pérdidas o fallas en el sistema.

- La carga de componentes en el nodo controlador es alta, por lo tanto al hacer deployment completo o en entornos de producción es importante crear un esquema donde se separan los componentes en al menos 2 nodos controladores. La carga para un solo nodo es muy alta y los nodos de reciclaje no tienen una tendencia a resistir cargas altas.
- Las redes que sean creadas deben tener como objetivo pasar a través del internet y evitar completamente la red local para evitar problemas de seguridad y de configuración. Muchas veces las redes locales prohíben accesos como por ejemplo a servidores NTP lo cual puede limitar la sincronización de hora entre nodos.
- Los nodos de cómputo deben en su gran mayoría soportar virtualización por hardware ya que esto genera una gran capacidad de procesamiento y permite maquinas virtuales de 64 bits. Esto por otro lado no se puede lograr usando QEMU y virtualización por medio de software.
- Es importante crear una variedad de flavors para que los clientes desperdicien la menor cantidad de recursos y fomentar el uso de políticas que limiten los recursos y hagan que cuando una instancia no esté en uso se apague automaticamente o suspenda hasta que el usuario determine si es justificado o no.
- El clúster permite alta disponibilidad en aplicaciones de tipo servidor tales como Apache2, MySQL Server y PHP. Es importante demostrar que estas aplicaciones pueden ser ejecutadas con éxito y su uso puede extenderse.

6.2 Recomendaciones:

- Para el buen funcionamiento del sistema es importante realizar un mantenimiento rutinario cada 60 Días donde se revise el estado de los componentes.
- Los nodos adicionales que sean agregados al sistema deben tener como minimo 1GB de ram y un procesador de la tecnología Pentium 4 o superior con un a tarjeta de red Megabit Ethernet. Nodos con tarjetas de velocidad menor causaron latencias más altas de lo normal en toda la red.
- Al reiniciar el clúster verificar en cada uno de los nodos que la hora de sincronía sea la correcta. En caso de que Neutron no pueda acceder a una instancia puede ser por la hora de sistema ya que muchos equipos no cuentan con pilas de BIOS con carga.
- La carga que soporta el nodo controlador es proporcional a la cantidad de nodos de cómputo, es importante dividir componentes o aumentar la memoria ram y agregar otro procesador al servidor al querer trabajar con más de 50 nodos de Cómputo.
- Cuando un nodo deja de funcionar el mecanismo de levantamiento automático forza a que la carga se distribuya en otros nodos. Cuando este proceso se este realizando es muy importante dejar al nodo controlador desatendido y deshabilitar temporalmente la interfaz web (Horizon).
- Es una buena práctica una vez que el sistema sea funcional, manejar contenedores de sistemas operativos. Linux containers es una tecnología que permite la distribución de carga para aplicaciones similares. Agregar esta característica al cluster puede reducir la carga y aumentar el ancho de banda en aplicaciones como Apache o MySQL.
- Para establecer servicios básicos tales como APACHE o entornos de desarrollo con interfaz gráfica, es importante que el administrador cree imágenes pre-editadas con el

contenido anteriormente mencionado para reducir el trabajo de los usuarios y establecer políticas correctas de uso.

- El servidor de DNS de el nodo controlador siempre debe permanecer en 8.8.8.8 ya que si falla el query DNS el servicio de NTP no podra obtener la hora del sistema por lo tanto habria un problema de sincronizacion y certificados. Al momento de existir un problema de sincronizacion de hora Neutron no podrá crear la conexión entre nodos.
- Cuando se crea una red con una sola NIC es obligatorio crear dos VLANs, al segmentar la red las máquinas virtuales no tomarán la configuración por defecto del servidor DHCP de Neutron.
- Es una buena práctica implementar Linux Containers por defecto, par el caso particular se necesita un nodo de cómputo adicional que solo corra los containers por lo cual no fue posible hacer su implementación en el piloto.
- Implementar módulos adicionales, Orchestrate y Telemetry pueden ayudar a la sección de métricas en el sistema. Debido a la falta de hardware no se pudo instalar más componentes.

7. REFERENCIAS BIBLIOGRÁFICAS

P. Jain, A. Datt, A. Goel and S. C. Gupta, "Cloud service orchestration based architecture of OpenStack Nova and Swift," *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, 2016, pp. 2453-2459.

doi: 10.1109/ICACCI.2016.7732425

A. Bousselmi, J. F. Peltier and A. Chari, "Towards a massively distributed IaaS operating system: Composition and evaluation of OpenStack," *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, Berlin, Germany, 2016, pp. 1-6.

doi: 10.1109/CSCN.2016.7785190

Vyas, Uchit. "Designing Your First Cloud With Openstack". *Springer e-books*. N.p., 2016. Print.

A. Celesti, D. Mulfari, M. Fazio, M. Villari and A. Puliafito, "Improving desktop as a Service in OpenStack," *2016 IEEE Symposium on Computers and Communication (ISCC)*, Messina, 2016, pp. 281-288.

doi: 10.1109/ISCC.2016.7543755

Button, Kimberly. "20 STAGGERING E-WASTE FACTS". *Earth911.com*. N.p., 2016. Web. 20 Dec.
2016.

Columbus, Louis. "Roundup Of Cloud Computing Forecasts And Market Estimates, 2016". *Forbes.com*. N.p., 2016. Web. 20 Dec. 2016.

Kupidura, Bartek. "Understanding Openstack Authentication: Keystone PKI | Mirantis". *Mirantis*. N.p., 2013. Web. 20 Dec. 2016.

Marlin, John. "Configuring Windows Failover Cluster Networks". *Microsoft Blog*. N.p., 2014. Web. 20 Dec. 2016.

"Openstack Docs: System Architecture". *Docs.openstack.org*. N.p., 2016. Web. 20 Dec. 2016.

Simonite, Tom. "The Foundation Of The Computing Industry'S Innovation Is Faltering. What Can Replace It?". *MIT Technology Review*. N.p., 2016. Web. 20 Dec. 2016.

"Vmware Compatibility Guide". *VMware Tech Support*. N.p., 2016. Web. 20 Dec. 2016.

Zitzman, Sharone. "Openstack Wiki In Short – A Quick Guide To Open Cloud". *Cloudify*. N.p., 2014. Web. 20 Dec. 2016.

Christensson, P. (2009, October 27). *Definitions A-Z*. Retrieved 2016, Dec 20, from http://techterms.com.

Brown, C. V., DeHayes, D. W., Hoffer, J. A., Martin, E. W., & Perkins, W. C. (2014). Managing information technology. Great Britain: Pearson.

ANEXO A: DEFINICIONES

- **Base de Datos:** Estructura de datos que almacena información organizada que permite a los usuarios acceder, actualizar y buscar información en base a la relación de los datos almacenados (Christensson, 2009).
- **BASH:** Lenguaje de programación de alto nivel desarrollado para UNIX. Fue diseñado originalmente para el desarrollo de programas para decodificadores y dispositivos de mano. (Christensson, 2012).
- **Cloud:** El término "nube" proviene de diagramas de red tempranos, en los que la imagen de una nube se utiliza para indicar una red grande, como una WAN. La nube finalmente se asoció con toda la Internet, y los dos términos se utilizan ahora sinónimo. La nube también se puede utilizar para describir servicios en línea específicos, que se denominan colectivamente "cloud computing".(Christensson, 2016)

- **Clúster**: Un clúster también puede referirse a un grupo de máquinas que trabajan juntas que realizan una función similar. Los nodos trabajan juntos para completar una sola tarea. Este proceso se llama "computación paralela" ya que los nodos realizan operaciones en tándem. (Christensson, 2016)
- **Hardware:** Conjunto de componentes físicos que ejecutan operaciones del sistema de ordenador (Brown et al. ,2014).
- **NIC:** Traduce a "Tarjeta de interfaz de red". Pronunciada "nick", esta es la tarjeta que hace físicamente la conexión entre la computadora y el cable de red. Estas tarjetas normalmente utilizan una conexión Ethernet y están disponibles en 10, 100 y 1000 configuraciones de Base-T. (Christensson, 2016)
- **Nodo:** Cualquier sistema o dispositivo conectado a una red también se llama nodo. Por ejemplo, si una red conecta un servidor de archivos, cinco equipos y dos impresoras, hay ocho nodos en la red. Cada dispositivo de la red tiene una dirección de red, como una dirección MAC, que identifica de forma exclusiva cada dispositivo. Esto ayuda a mantener un registro de dónde se están transfiriendo los datos hacia y desde la red.(Christensson, 2012)
- **Núcleo:** Un núcleo de procesador (o simplemente "núcleo") es un procesador individual dentro de una CPU. Muchas computadoras hoy en día tienen procesadores multi-núcleo, lo que significa que la CPU contiene más de un núcleo. (Christensson, 2016)
- **Red:** Cuando tiene dos o más equipos conectados entre sí, tiene una red. El propósito de una red es permitir el intercambio de archivos e información entre múltiples sistemas. Internet podría ser descrito como una red global de redes. Las redes de ordenadores se pueden conectar a través de cables, como cables Ethernet o líneas

telefónicas, o de forma inalámbrica, utilizando tarjetas de red inalámbrica que envían y reciben datos a través del aire. (Christensson, 2016)

- **Servidor:** Un servidor es una computadora que proporciona datos a otras computadoras. Puede servir datos a sistemas en una red de área local (LAN) o una red de área extensa (WAN) a través de Internet. (Christensson, 2016)
- **Sistema Operativo:** Un sistema operativo, o "SO", es un software que se comunica con el hardware y permite que otros programas se ejecuten. Se compone de software del sistema, o los archivos fundamentales de su computadora necesita para arrancar y funcionar. Cada computadora de escritorio, tableta y teléfono inteligente incluye un sistema operativo que proporciona funcionalidad básica para el dispositivo. (Christensson, 2016)
- **Software:** Conjunto de programas que controlan las operaciones del sistema operativo de ordenador (Brown et al. ,2014).
- **Token:** En la conexión en red, un token es una serie de bits que circulan en una red de token-ring. Cuando uno de los sistemas de la red tiene el "token", puede enviar información a los otros equipos. Dado que sólo hay un token para cada red de token-ring, sólo una computadora puede enviar datos a la vez.(Christensson, 2016)
- **Unix:** También conocido como UNIX, aunque las letras no representan nada. La plataforma que lidera la industria con el sistema operativo más común para los servidores Web.(Christensson, 2016)
- **Virtualización:** Se refiere a ejecutar varios sistemas operativos en una sola máquina. Mientras que la mayoría de las computadoras solo tienen un sistema operativo instalado, el software de virtualización permite que una computadora ejecute varios sistemas operativos al mismo tiempo. (Christensson, 2016)