UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
VERITATE SOLA NOBIS IMPONETUR VIRILIS TOGA. 1948

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

Propuesta de diseño para la implementación de una Infraestructura como Servicio mediante plataformas de código abierto

Autor: Laura Broche González

Tutores: MSc. Ing. Carlos M. Bustillo Rodríguez

Dr.C.Ing. Héctor Cruz Enriquez

Santa Clara

2016

"Año 58 de la Revolución"

UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
VERITATE SOLA NOBIS IMPONETUR VIRILIS TOGA. 1948

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

Propuesta de diseño para la implementación de una Infraestructura como Servicio mediante plataformas de código abierto

Autor: Laura Broche González

lbgonzalez@uclv.cu

Tutores: MSc. Ing. Carlos M. Bustillo Rodríguez

Administrador de Red UCLV
cbustillo@uclv.edu.cu

Dr.C.Ing. Héctor Cruz Enriquez

Director de Informatización, UCLV
hcruz@uclv.edu.cu

Santa Clara

2016

"Año 58 de la Revolución"

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Autor

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

La ciencia es hoy la tecnología del mañana.

Edward Teller

DEDICATORIA

Dedico este tesis especialmente a mi abuelo Bonifacio Broche González, el que estuvo siempre orgulloso de mis progresos y solo le faltó verme exponer esta tesis, donde quieras que estés sé que está orgulloso de esto, y a mis abuelos Danilo, Miriam y Nereida.

A mis padres Yigany y Jorge Luis pues ellos siempre han sido mis guías.

A mi hermana Elizabeth y mi prima Melissa, que les sirva de motivación para cuando tengan que hacer las suyas.

A mis tías Dania y Lidia que siempre están complaciéndome.

A mis amigas por iluminar mi vida con su presencia.

AGRADECIMIENTOS

En el transcurso del trabajo de esta tesis he tenido el placer y la suerte de encontrarme muchas personas que, gracias a su apoyo y guía, han hecho posible que este proyecto se haya completado felizmente.

Quiero reconocer el inmenso trabajo llevado a cabo por Carlos Miguel Bustillo Rodríguez, tutor de esta tesis, así como la inspiración y atención prestada, responsable de la calidad de este documento. Agradecer a Héctor Cruz Enriquez, también tutor y director del grupo de informatización en la UCLV en el cual se ha llevado a cabo todo este trabajo. Hago una mención especial a Álvaro Simón García, por sus consejos a la hora de instalar y desplegar OpenNebula y sus explicaciones pacientes de los aspectos más intricados de la tecnología que se maneja en este trabajo. También a Yalina Rodríguez de Armas por aclarar las dudas respecto a LXC.

Otros dos grandes culpables de que esta investigación se haya llevado a cabo son mis padres, Yigany y Jorge Luis. Sin su dedicación no hubiese sido posible terminar mi carrera y hacer este trabajo. Gracias de corazón. Como también se lo debo a mis abuelos Danilo, Miriam, Nereida y Broche por su apoyo en cada paso que di, y mis hermanas Elizabeth y Yanelis. Sin dejar al resto de la familia que no los puedo mencionar a todos porque es muy extensa. Muchas gracias a mi novio Giovanni por estar a mi lado en estos momentos. A mis incondicionales amiga Marianela, Jessica, Eylis … por aguantarme tantos años y en especial a Patricia que estuvimos juntas siempre durante toda la carrera. También a todos mis amigos del aula Jorge Alberto (y su familia), Adrián, Manso, Rasiel, Ernesto, Julio, Artion, Yuly y demás.

Muchas gracias a todo el que estuvo implicado en este largo proceso.

TAREA TÉCNICA

1. Búsqueda de información sobre la implementación y funcionamiento de tecnologías *Cloud Computing*.
2. Análisis de la infraestructura que posee la institución para aplicar la nueva implementación.
3. Estudio de las principales ventajas del uso de las tecnologías de IaaS.
4. Selección de una plataforma para la gestión eficiente de la nube.
5. Implementación de una Infraestructura como Servicio en la UCLV.
6. Evaluación de la efectividad de la propuesta de la implementación a través de la herramienta seleccionada.

RESUMEN

Las tecnologías: virtualización de servidores y *Cloud Computing* ofrecen nuevas formas de utilizar los medios informáticos al permitir acceso de forma lógica, en lugar de física, a datos, capacidad de cómputo, almacenamiento y otros recursos. Existen distintos modelos de servicio de *Cloud Computing*, entre ellos, Infraestructura como Servicio donde se ofrece a los clientes varios recursos computacionales, entre los más destacados: abastecimiento de procesamiento, espacio de almacenamiento y equipos de red. Entre las plataformas de gestión de nube de código abierto más usada se encuentran OpenNebula, Eucalyptus, OpenStack y CloudSatck. Se determinó OpenNebula como la herramienta más abierta, escalable, flexible y de simple instalación y despliegue. Con la utilización de esta plataforma se logró la gestión y administración eficiente de los sistemas de virtualización y alto desempeño. Se implementó esta tecnología de prueba en la UCLV, para ello se consultaron las guías que ofrece la comunidad para su instalación y despliegue, en las mismas se detectaron algunas deficiencias a las que se le dieron solución a lo largo del proyecto. Asimismo se dejaron plasmadas algunas configuraciones útiles para la gestión de las máquinas virtuales.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Los avances en las tecnologías de virtualización y cómputo distribuido dan origen al paradigma de *Cloud Computing*, que se presenta como una alternativa a los tradicionales sistemas de Clústeres y *Grids* para ambientes de cómputos de alto rendimiento (*High Performance Computing,* HPC).

La virtualización es un componente fundamental para implementar el paradigma *Cloud Computing* dado que permite realizar balance de carga, ahorro de energía, recuperación tras una falla y un manejo versátil para el mantenimiento del sistema. Esta tecnología provee una abstracción de los recursos de *hardware* mediante la ejecución simultánea de instancias de múltiples sistemas operativos (*SO*) en máquinas virtuales (*VMs*) sobre un único *hardware* físico.

Cloud Computing, proporciona grandes conjuntos de recursos físicos y lógicos (como puede ser infraestructura, plataformas de desarrollo, almacenamiento y/o aplicaciones), fácilmente accesibles y utilizables por medio de una interfaz de administración *web*, con un modelo de arquitectura "virtualizada".

Estos recursos son proporcionados como servicios ("*as a service*") y pueden ser dinámicamente reconfigurados para adaptarse a una carga de trabajo variable (escalabilidad), de esta manera se logra una mejor utilización y se evita el sobre o sub dimensionamiento (elasticidad). Existen tres modelos de despliegue en un sistema: nube pública, nube privada y nube híbrida. A su vez también hay tres modelos de servicio:

1. Infraestructura como Servicio (*Infrastructure as a Service*, IaaS) ofrece como servicio a los clientes varios recursos computacionales, entre los más destacados: abastecimiento de procesamiento, espacio de almacenamiento y equipos de red.

2. Plataforma como Servicio (*Platform as a Service,* PaaS) brinda a los usuarios la facilidad de desplegar sus aplicaciones sobre la infraestructura de *Cloud Computing,* basadas en lenguajes y herramientas de programación que el proveedor soporte.
3. *Software* como Servicio (*Software as a Service,* SaaS) permite al usuario utilizar cualquier tipo de aplicaciones manejadas por el proveedor, sin la necesidad de que el usuario adquiera, instale o maneje cualquier tipo de actualización del servicio.

Los consumidores, desarrolladores y proveedores de una Infraestructura como Servicio necesitan tomar la decisión sobre qué plataforma es más adecuada para gestionar e implementar su infraestructura. En las últimas décadas, la tecnología de código abierto ayuda a los clientes que no desean utilizar la infraestructura comercial para la nube. Existen diferente plataforma de código abierto, entre ellas OpenNebula, Eucalyptus, OpenStack y CloudStack son las que mayormente han sido usada para la gestión de la nube, así como las alternativas que proveen nubes comercialmente.

La UCLV cuenta con dos sistemas de virtualización: a) VMWare ESX: sistema propietario y b) Proxmox VE: sistema basado en *software* libre. Ambas plataformas de virtualización forman clústeres por separados, lo cual implica que los especialistas necesiten dominar ambas tecnologías que emplean comandos, interfaces de administración e hipervisores completamente diferentes. De ahí a que convertir una máquina virtual de un sistema a otro lleve tareas adicionales.

Por este inconveniente se propone la implementación de una IaaS, ya que el usuario gestiona mediante una aplicación *web* todos los recursos necesarios y despliega por sí mismo las máquinas virtuales requeridas. Esta situación da paso a la siguiente pregunta:

¿Cómo contribuir a la gestión eficiente de los sistemas de virtualización y alto desempeño con el empleo de la tecnología IaaS?

Por tanto es necesario tener en cuenta:

- ¿Cuáles son las tecnologías *Cloud Computing* que existen en la actualidad?
- ¿Qué herramienta para la gestión integrada de los sistemas de virtualización y de alto desempeño es más eficiente?

- ¿Qué elementos se deben tener en cuenta al implementar la tecnología IaaS seleccionada?

Por ello el objetivo general de este trabajo es: "Proponer una implementación de la tecnología IaaS que garantice la gestión eficiente de los sistemas de virtualización y alto desempeño."

Del cual se derivan los siguientes objetivos específicos:

- Describir las tecnologías *Cloud Computing* existentes.
- Seleccionar una herramienta de código abierto para la gestión eficiente de la nube.
- Diseñar una propuesta de implementación de la tecnología IaaS que garantice la gestión eficiente de los sistemas de virtualización y alto desempeño con el empleo de la herramienta seleccionada.

Organización del informe

El informe está estructurado de la siguiente forma: introducción, capitulario, conclusiones, recomendaciones, referencias bibliográficas, glosario de términos y anexos. A continuación se resume brevemente el contenido de los capítulos.

Capítulo 1: Tecnología *Cloud Computing*.

Se dedica a la caracterización de la tecnología *Cloud Computing*, se evalúa cada uno de sus modelos de implementación y servicio. También se describen otras tecnologías como la virtualización.

Capítulo 2: Herramientas de código abierto para la gestión eficiente de nubes.

Se hace una comparación de las herramientas de código abierto para la gestión de nubes más utilizadas como: OpenNebula, OpenStack, CloudStack y Eucalyptus, y se selecciona la más eficiente de acuerdo a la necesidades de la institución.

Capítulo 3: **Implementación de la plataforma de gestión OpenNebula**.

Se mencionan los principales pasos a tener en cuenta en la instalación y despliegue de la plataforma OpenNebula, así como las principales deficiencias encontradas y las soluciones a las mismas.

CAPÍTULO 1. TECNOLOGÍA *CLOUD COMPUTING*

Los usuarios siempre esperan un servicio de vanguardia que satisfaga sus necesidades, es por eso que se hacen más comunes aquellos servicios que, con su uso se aprovecha al máximo los recursos disponible. Con esta idea surgen las tecnologías: virtualización de servidores y computación en la nube (del inglés, *Cloud Computing)* que ofrecen nuevas formas de utilizar los medios informáticos al permitir acceso de forma lógica, en lugar de física, a datos, capacidad de cómputo, almacenamiento y otros recursos.

Esto trae consigo un gran impacto económico pues de esta manera se reducen los costos de *hardware* y mantenimiento, lo que conlleva además a una notable disminución del consumo de energía, necesidad de refrigeración, y contaminación. Es importante resaltar que con el uso de una Infraestructura como Servicio (*Infrastructure as a service*, IaaS), se logra la gestión y administración eficiente de los sistemas de virtualización y alto desempeño.

1.1 Introducción a la tecnología *Cloud Computing*

En los últimos años se han visto evolucionar tecnologías vitales para el mundo empresarial, las cuales permiten que *Cloud Computing* irrumpa como un paradigma o modelo a costos razonables para acceder y proveer recursos computacionales sobre *Internet*.

El Instituto Nacional de Normas y Tecnología (*National Institute of Standards and Technology*, NIST) presenta una de las definiciones de *Cloud* más clara y comprensible. La define como un modelo que habilita acceso a red ubicuo, conveniente, bajo demanda para compartir un conjunto de recursos configurable, que pueden ser rápidamente provistos y liberados con mínimo esfuerzo o interacción del proveedor de servicios. Distingue las características de *Cloud*, el modelo de entrega y los métodos de desarrollo. Resalta así, los

cinco (5) aspectos claves de *Cloud Computing*: auto servicio bajo demanda, acceso a la red universal, un conjunto de recursos independiente de la ubicación, rápida elasticidad y servicio a la medida [1].

Cloud Computing no es un desarrollo revolucionario reciente, sino es el resultado de la evolución de varias tecnologías. Conceptos precursores son: *utility computing*, computación bajo demanda y *grid computing* [2].

Es un modelo de aprovisionamiento de recursos de Tecnologías de Información (IT) que potencia la prestación de servicios y los negocios, esto facilita la operativa del usuario final y del proveedor del servicio. La característica básica de este modelo es que los recursos y servicios informáticos, tales como infraestructura, plataforma y aplicaciones, son ofrecidos y consumidos a través de Internet sin que los usuarios tengan que tener ningún conocimiento de lo que sucede detrás [3].

1.1.1 Características de *Cloud Computing*

Se pueden enumerar una serie de características relacionadas con este paradigma de computación en la nube, ejemplo de estas son:

Auto-servicio

Se simplifica la gestión de los administradores de la infraestructura. Generalmente por medio de una interfaz *web* o mediante línea de comandos (*Command Line Interface*, CLI), los usuarios son capaces de aprovisionarse de los recursos que necesitan de una forma sencilla. El usurario final no necesita conocer cómo se diseña la infraestructura, cuáles son las tecnologías subyacentes, ni aprender una extensa documentación para hacerla funcionar.

Escalabilidad y Elasticidad

Los recursos que se ofrecen en la nube son altamente escalables, dado que una aplicación o un usuario pueden agregar o reducir dinámicamente sus recursos en respuesta a la variación en la carga de trabajo.

Multipropósito

El sistema puede estar diseñado para que varios usuarios puedan compartir la infraestructura y le den un uso específico, sin comprometer la privacidad y seguridad de los datos de cada usuario.

Orientada a servicio

Este paradigma computacional se ideó para desplegar servicios ya sean bajo algún costo o de manera gratuita en dependencia del propósito.

Disponible bajo demanda

Los recursos se entregan bajo demanda en un corto plazo o reservados con antelación. La gestión del *hardware* se abstrae del consumidor, estos no incurren en gastos de infraestructura ni en los costes asociados a la gestión de la misma. Esto permite eliminar los costes iniciales de adquisición de la infraestructura y pasar directamente a consumir bajo demanda, lo que acorta el tiempo de acceso al mercado. El consumidor únicamente incurre en gastos de operacionales, como gastos de explotación o funcionamiento.

Eficiencia

Mediante una estricta monitorización de los recursos de la infraestructura se permite planificar y predecir su comportamiento. Se puede tener control sobre los recursos que se utilizan y el estado de los servicios de que disponen los usuarios. Conocer el uso que se le da a la infraestructura permite avanzarse a los acontecimientos y actuar en consecuencia [4].

Autorreparable

La probabilidad de un fallo de *software* es más alta que la de *hardware*, no obstante se asume, que estos ocurren y son inevitables, la recuperación del sistema *software* se realiza de una manera mucho más rápida (asegura la continuidad del negocio), que lo que supondría la recuperación de un sistema tradicional, pues esto trae consigo compra de nuevo recurso *hardware*, instalación, configuración y pruebas [5].

Acuerdo a Nivel de Servicio (*Service Level Agreement*, SLA)

Los servicios se prestan bajo un contrato entre las partes, la negociación de los términos es siempre una difícil tarea y debe ir acompañada de indicadores que permitan evaluar la

prestación del servicio. La prestación de SLAs de forma dinámica son capaces de cuantificar el uso que se hace del servicio [4].

Sostenibilidad

El paradigma *Cloud* como la virtualización se presenta como herramientas decisivas para reducir las emisiones de dióxido de carbono (CO_2) y el consumo energético en los Centros de Procesamiento de Datos (CPD) [6].

La solución que aporta el modelo *Cloud Computing* conjuntamente con la virtualización es mejorar la gestión de la infraestructura mediante optimización y planificación del uso.

1.1.2 Taxonomía

Existen varios puntos de vista en cuanto a la organización de *Cloud Computing***,** en la Figura I.1 Anexo I se muestra el más aceptado, donde los consumidores de servicios (*service consumers*) utilizan los servicios prestados por los proveedores (*service providers*) a través de la Nube; los proveedores gestionan la infraestructura y los desarrolladores de servicios (*service developers*), son los encargados de crear los servicios a través de los estándares abiertos, los cuales son necesarios para la interacción entre los roles.

El consumidor de los servicios, es el usuario final o empresa que en realidad utiliza el servicio, ya sea *software*, plataforma o infraestructura como servicio, ver Figura 1.I Anexo I.

En dependencia del tipo de servicio y su función, el consumidor trabaja con diferentes interfaces de usuario o de programación. Este no necesita saber acerca de *Cloud Computing* ya que las interfaces son parecidas a las de escritorio. Otras interfaces de usuario proporcionan funciones administrativas, tales como iniciar y detener las VM o gestionar el almacenamiento de las nubes.

Los consumidores escriben el código de una aplicación y usan diferentes interfaces en correspondencia de la aplicación que se realice. Trabajan con los SLA y los contratos. Generalmente, éstos se negocian entre el consumidor y el proveedor. Las expectativas del consumidor y la reputación del proveedor son una parte clave de esas negociaciones [7].

El desarrollador del servicio crea, edita y supervisa los servicios que ofrecen las nubes; por lo general, las aplicaciones que se entregan directamente a los usuarios finales a través del modelo SaaS (*Software as a Service*) son "líneas de negocio", las aplicaciones escritas en IaaS (*Infraestructure as a Service*), y los niveles de PaaS (*Platform as a Service*) [8].

El proveedor del servicio es el encargado de entregar al consumidor los servicios. Los detalles de estas tareas se especifican más adelante, ya que existen diferentes tipos de servicios y otros aspectos que deben tomarse en cuenta para poder ofrecerlos, ver Figura I.1 Anexo I.

En *Software* como Servicio el proveedor instala, gestiona y mantiene el *software*. El proveedor no necesariamente es dueño de la infraestructura donde se ejecuta el *software*, pero independientemente de eso, el consumidor no tiene acceso a esa infraestructura, únicamente a las aplicaciones.

Para Plataforma como Servicio, el proveedor gestiona la infraestructura para la plataforma de las nubes, por lo general una infraestructura, para un tipo concreto de aplicación. El consumidor no puede acceder a la infraestructura por debajo de la plataforma.

Para Infraestructura como Servicio, el proveedor mantiene el almacenamiento, la base de datos, la herramienta de gestión o el entorno para alojar las máquinas virtuales. El consumidor utiliza dicho servicio como si fuera una unidad de disco duro, bases de datos, o una máquina, pero no puede acceder a la infraestructura que aloja este servicio [7].

Estos tres tipos de servicios, serán retomados más adelante para tratarlos con mayor detalle.

1.2 Modelos de implementación en *Cloud Computing*

Se deben considerar varios aspectos a la hora de tomar la decisión de implementar un modelo de *Cloud Computing*. Existen tres modelos básicos a considerar, los cuales pueden ser de origen propietario, o basados en *software* libre.

1.2.1 Modelo público

El proveedor de los servicios de *Cloud Computing* es dueño de la infraestructura física y pone a disposición del cliente los servicios de la nube a través de Internet; ésta es su

característica principal, y es la que permite que el usuario pueda acceder a dichos servicios en cualquier momento y lugar, ver Figura I.2 Anexo I [7].

El usuario tiene el control de determinados recursos y puede conectarse a través de un navegador *web* o un cliente SSH, en dependencia del tipo de servicio que utilice. De acuerdo al uso de los servicios que ofrece la nube, el usuario tiene que pagar únicamente por lo que consuma. Esto representa un gran ahorro en la inversión inicial de un proyecto, puesto que no se tiene que comprar toda la infraestructura para desplegarlo.

Una nube pública provee servicios a múltiples clientes, y típicamente se implementa en un centro controlado, instalaciones seguras, con vigilancia de 24 horas y un sistema de alimentación ininterrumpida. A medida que las necesidades del usuario se incrementan, eventualmente, éste podría requerir mayores prestaciones del proveedor, lo cual no representa ningún riesgo en el rendimiento ya que el modelo permite contar con escalabilidad de una manera sencilla.

Los sistemas en la nube controlan y optimizan el uso de los recursos de manera automática, por tanto, el uso de éstos puede seguirse, controlarse y notificarse, lo que aporta transparencia tanto para el proveedor como para el consumidor del servicio [5].

Al ser un sistema establecido en una red pública, la seguridad es un aspecto importante; actualmente, se utilizan métodos para transmitir la información cifrada pero hay que tener en cuenta que los datos no se encuentran almacenados localmente. Esta característica es lo que promueve crear otro modelo de nube privada. A pesar de ello, los proveedores cuentan con centros de datos que se dedican, específicamente, a custodiar y salvaguardar los datos ya que se implementan todas las medidas de seguridad necesarias, tanto de *hardware* como de *software*, de forma que no haya jamás una pérdida de información ni de integridad de los datos. La única precaución que hay que tener, es encontrar el proveedor que ofrezca las garantías y prestaciones adecuadas a los servicios que se requieren.

1.2.2 Modelo privado

Es una emulación de la nube pública, pero en una red privada. Ofrece los mismos servicios que una nube pública con la ventaja de que el usuario cuenta con sus propios recursos, lo

que le permite tener el control total de seguridad y calidad de servicio (*Quality of Service, QoS*), sobre ellos.

Entre los requisitos básicos de una nube privada se pueden mencionar: medición y control, gestión, seguridad, despliegue o implementación, interoperabilidad y un formato de máquina virtual común. Una nube privada no requiere estándares de la industria, ni Interfaz de Programación de Aplicaciones (*Application Programming Interface,* API) comunes entre otras nubes para la herramienta de *software* que se ha utilizado en su implementación.

Las nubes privadas pueden ser implementadas directamente por el propietario o por un proveedor. En este caso el proveedor, únicamente, se encarga de instalar todos los elementos necesarios para que el cliente disponga de la infraestructura de *Cloud Computing*, en su propio centro de datos. La otra opción consiste en que el usuario implemente su propia nube, para esto se puede usar sistemas basados en *software* libre.

Este modelo se puede implementar en un centro de dato corporativo. Pueden obtener el soporte de la misma compañía, de un proveedor de *Cloud Computing*, o de un tercero como una firma de subcontratación. Además permite incrementar el nivel de seguridad de los datos que se consideren como sensibles, pues la infraestructura está bajo el control de la misma organización que la utiliza, ver Figura I.2 Anexo I [7].

La escalabilidad en este modelo se presenta de una manera sencilla pues en el caso de que el sistema que se implemente haga uso máximo sus recursos, se puede contar con la interconexión a una nube pública, lo cual se conoce como nube híbrida. De esta manera se pueden solucionar picos de demanda, sin la necesidad de invertir en más recursos de *hardware* que sólo se utilizan en determinados momentos [5].

1.2.3 Modelo híbrido

La nube híbrida combina los modelos público y privado. Tiene la ventaja de contar con los beneficios de ambos modelos, lo cual permite aumentar la capacidad de una nube privada con los recursos de una nube pública para poder mantener niveles de servicio adecuados frente a rápidas fluctuaciones de carga de trabajo. La Figura I.2 del Anexo I muestra la conexión de una empresa con una nube híbrida, se tiene también dentro de sus dominios una nube privada [7].

Este modelo de *Cloud Computing* puede ser particularmente efectivo cuando ambos modelos de nube, se ubican en la misma localidad. Este introduce la complejidad de determinar cómo distribuir aplicaciones entre las nubes pública y privada. Entre las cuestiones que se deben tener en cuenta está la relación entre los datos y recursos de procesamiento.

Si la cantidad de datos que se maneja es pequeña, una nube híbrida puede ser más conveniente, ya que esos datos pueden ser almacenados y procesados ya sea en la nube privada o en pública. En el caso de manejar grandes volúmenes de datos y que se requiera una pequeña capacidad de procesamiento, se aconseja utilizar una Nube privada, pues no se justificaría contratar un gran ancho de banda para transferir los datos a la nube pública para que esta los procese [5].

1.3 Arquitectura en capas y modelo de servicio de *Cloud Computing*

Cloud Computing tiene la capacidad de ofrecer cualquier tipo de servicio basándose en la estructura tradicional de capas que va desde el *hardware* hasta las aplicaciones. Los servicios que ofrecen las nubes son útiles para cualquier tipo de consumidor o entidades de negocios.

Los ofrecimientos de *Cloud Computing* son relativamente amplios, ya que los clientes se conectan a los diversos servicios a través de una red (mediante un navegador *web* o cliente SSH), sin importar el tipo de dispositivos ni la localización de los usuarios, lo cual implica un ahorro de recursos al utilizar dispositivos antiguos que tal vez ya no tenían buen uso pues, para conectarse con la nube no se necesita de un sistema operativo completo.

El Internet sirve como una buena solución para situaciones en las que es más importante obtener un resultado, que el propio *hardware* o *software* que se necesita para conseguirlo. El uso de navegadores *web* ha dado lugar a una migración constante del modelo tradicional de centros de datos a un modelo basado en la nube [7].

Desde el punto de vista del proveedor (ver Figura I.3 Anexo I), la arquitectura de *Cloud Computing* comprende tres capas. A continuación se explica detalladamente cada una de ellas y los diferentes servicios ofrecidos por la capa superior.

1.3.1 *Hardware*

Esta capa está en la parte inferior de la arquitectura en capas de *Cloud Computing,* contiene a todos los dispositivos físicos que hacen funcionar a la nube y envuelve a un sin número de servidores apilados, por lo general dentro de centros de datos.

Algunos de los servidores están destinados a almacenar datos y archivos, y otros a procesar información. Todos ellos son fácilmente reemplazables y si se requiere ampliar la capacidad de la nube, tan sólo se añaden más o se puede construir un nuevo centro de datos, si los recursos económicos son suficientes [9].

1.3.2 Virtualización

La capa de virtualización se encuentra entre la capa de hardware y la capa de servicios.

La virtualización es un término amplio que se refiere a la abstracción de los recursos de una computadora. Este término se menciona desde antes de 1960, y es aplicado en diferentes aspectos de la informática, desde sistemas operativos completos hasta componentes individuales.

La virtualización de la plataforma crea una interfaz externa que esconde una implementación subyacente mediante la combinación de recursos en ubicaciones físicas diferentes, o mediante la simplificación del sistema de control [10]. Existen distintos tipos de virtualización, que se tratan con más detalles en la sección 1.4.

La virtualización es un componente fundamental para implementar el paradigma *Cloud Computing*. Esta tecnología provee una abstracción de los recursos de *hardware* y permite ejecutar simultáneamente múltiples instancias de sistemas operativos en máquinas virtuales (VM) sobre un único *hardware* físico [11] [12].

Cuando un usuario solicita una colección de VM a una nube, estas deben estar completamente aisladas de las otras en términos de acceso a la CPU, memoria, red y almacenamiento persistente.

Todas estas formas de acceso aislado deben estar asociadas a un usuario autenticado con propósitos de seguridad. La virtualización del sistema operativo y los hipervisores

proporcionan un aislamiento no autenticado de CPU y memoria, pero no de la red privada entre VM ni almacenamiento persistente por cada usuario,

Cloud Computing permite tener el control del acceso de cada usuario tanto a la red como a la máquina virtual, y en correspondencia del servicio que utilice en una nube tendrá el control de determinados recursos [13].

1.3.3 Modelos de servicios en *Cloud Computing*

La capa de servicios se encuentra en la parte superior de la arquitectura de capas de *Cloud Computing.*

La computación de hoy en día se desarrolla de tal forma que todo lo que se encuentra en un centro de datos se pueda ofrecer como servicio. Actualmente los proveedores de *Cloud Computing* ofrecen: Infraestructura como Servicio (*Infrastructure as a Service, IaaS*), Plataforma como Servicio (*Platform as a Service, PaaS*) o *Software* como Servicio (*Software as a Service, SaaS*), en la Figura 1.3.1 se muestran algunos ejemplos.

Los servicios que ofrecen por las nubes pueden ser manejados de diferentes formas de acuerdo al modelo de infraestructura que se utilice. En una nube de modelo privado la empresa tiene el control de acceso, y sobre toda la infraestructura que la conforma, lo que le permite disponer de servicios personalizados que se pueden adecuar totalmente a sus necesidades [14].

Figura 1.3.1 Modelos de servicio de *Cloud Computing*.

Por otro lado, el modelo público de *Cloud Computing*, es de propiedad y está controlado por un proveedor externo que proporciona acceso al mismo bajo suscripción. Por lo tanto, los servicios están definidos en un catálogo que no da margen a que sean personalizados [15].

Los tres niveles de servicio pueden ser ofrecidos si se implementa *Cloud Computing* de modelo público, privado o híbrido, y su utilización puede ir desde aplicaciones *web*, hasta Computación de Alto Rendimiento (*HPC*), ver Figura I.4 del Anexo I.

Infraestructura como Servicio (IaaS)

En Infraestructura como Servicio, la capacidad que se suministra a los clientes es el abastecimiento de procesamiento, espacio de almacenamiento, equipos de red y otros recursos computacionales importantes para que estos puedan desplegar y ejecutar *software* de forma arbitraria, lo cual puede incluir sistemas operativos y aplicaciones. Se ofrecen diferentes equipos como servidores, sistemas de almacenamiento, dispositivos de enrutamiento y otros que permiten manejar cargas de trabajo, que van desde los pequeños componentes hasta aplicaciones de computación de alto rendimiento, ver Figura I.5 Anexo I [15].

La infraestructura se ofrece, normalmente, mediante una plataforma de virtualización. Los clientes no gestionan ni controlan la infraestructura de la nube, pero tienen control sobre los sistemas operativos, almacenamiento, aplicaciones desplegadas y la posibilidad de tener un control limitado de componentes de red seleccionados [13].

En la Figura 1.4 se muestran algunos recursos ofrecidos como servicio a los clientes que utilizan IaaS (procesamiento de datos, ciclos de CPU, memoria y almacenamiento [15, 16].

IaaS permite a los proveedores de servicios el alquiler de recursos de *hardware*. Los clientes no invierten en infraestructura alguna ni incurren en gastos de operación. Estos pagan sólo por el uso del servicio; es decir, el costo se basa en el tiempo y número de recursos que utilice.

Una vez que se tenga una IaaS, se maneja un escalado dinámico e inmediato, de acuerdo a la aplicación y a la necesidad de recursos a utilizar. No importa la localización del usuario con respecto a la localización del proveedor del servicio. Se comparte la capacidad entre

múltiples usuarios para tener en cuenta los picos de carga y poder escalar rápidamente y sin dificultades. Se usan tecnologías de virtualización, que permiten a los usuarios ejecutar el número de VM que necesiten.

Existen diversas soluciones de *software* para generar IaaS, tanto código abierto como de ámbito privado: *Vmware, Citrix, 3Tera, Abiquo, Enomaly, Eucaplyptus, Proxmox, OpenNebula*, OpenStack, CloudStack, entre otros [17].

La ventaja más inmediata de elegir este tipo de soluciones, es desplazar una serie de problemas a los proveedores relacionados con la gestión de las VMs. También se debe situar el ahorro de costes, al pagar sólo por lo que se consume y aprovechar las economías de escala que tienen gigantes como *Amazon*. Además se tiene que la IaaS permite una escalabilidad, de forma que se puede contratar más recursos según se necesiten.

Plataforma como Servicio (PaaS)

En Plataforma como Servicio, el cliente puede desarrollar, probar e implementar sus aplicaciones en los centros de datos del proveedor, a través de diferentes lenguajes y herramientas de programación que el proveedor del servicio soporte. En general, este tipo de servicio permite construir sistemas de alto nivel, ya que proporciona todos los recursos necesarios para crear aplicaciones y servicios desde Internet, sin tener que descargar o instalar el *software*. Los clientes no gestionan ni controlan la infraestructura de la nube, pero tienen el control sobre las aplicaciones desplegadas y la posibilidad de controlar las configuraciones de entorno de alojamiento de las aplicaciones, ver Figura I.6 Anexo I [15].

Una desventaja de PaaS es la falta de interoperabilidad y portabilidad entre proveedores, es decir, si se crea una aplicación con un proveedor de nube y se decide cambiar de proveedor, muchas veces el precio que se debe pagar es relativamente alto o al final es imposible hacerlo [16]. Además, si PaaS deja de funcionar, los datos y aplicaciones de los clientes también van a desaparecer para evitar este tipo de inconvenientes, es necesario controlar dos factores: la información de la empresa y el servicio.

El servicio de plataforma ofrece cierto soporte de ayuda en la creación de interfaces de usuario basadas en Lenguaje de Marcado de Hipertexto (*HyperText Markup Language*, HTML) o *JavaScript.* También soporta el desarrollo de interfaces *web* con SOAP y REST

que permiten la construcción de múltiples Servicios *Web*, llamados aplicación *web* híbrida o *mashups.*

Adicionalmente, este tipo de servicio se diseña para brindar facilidades de acceso a múltiples clientes simultáneos, manejo de concurrencia, balance de carga y seguridad. Ejemplos de PaaS son: *Velneo, Abiquo.com, SimpleDB SQS Google App Engine*, entre otros.

***Software* como Servicio (SaaS)**

Es un modelo de distribución de *software*, el cual se encuentra alojado como una sola instancia, que se ejecuta en la infraestructura de la nube y es distribuido a múltiples clientes a través de una red (pública o privada). El *software* se implanta dentro de la nube y está disponible para el usuario a través de un navegador *web*, sin necesidad que el usuario lo adquiera, instale o realice algún tipo de mantenimiento, ya que el proveedor maneja todas las actualizaciones y el funcionamiento del servicio. Queda fuera del alcance del cliente el sugerir al proveedor realizar cambios al servicio que presta, ver Figura I.7 Anexo I.

SaaS provee una aplicación o una parte del *software* desde el proveedor de servicios. La capacidad proporcionada a los clientes consiste en utilizar las aplicaciones del proveedor que se ejecutan en una infraestructura de nube. Adicionalmente, el usuario no se preocupa dónde está instalado el *software*, qué tipo de sistema operativo utiliza o el lenguaje en el que éste está escrito.

En la Figura I.4 se esquematiza, de forma general, la manera en que los usuarios pueden acceder a SaaS, de manera que el proveedor ofrece sus servicios a través de una red pública como el Internet.

Este modelo permite que se desarrollen tecnologías como los Servicios *Web* y Arquitectura Orientada a Servicio (*Service Oriented Architecture,* SOA); adicionalmente, existen varios tipos de *software* que no necesitan mucha interacción con otros sistemas, como la videoconferencia, manejo de Servicios de IT y manejo de contenido *web*. Algunos ejemplos de SaaS son: *Google Apps, Documany, TeamBox, KubbosGupigupi, Salesforce, Basecamp, Gmail, Salesforce.com, MediaWiki, Moodle, WordPress*, etcétera.

1.3.4 Ventajas y desventajas de la tecnología *Cloud Computing*

Cloud Computing permite acceder a todas las aplicaciones y documentos desde cualquier lugar del mundo. Esta representa un gran cambio en la forma en que se almacena la información y se ejecutan las aplicaciones. El usuario final promedio se puede beneficiar de la computación en la nube, pero algunos usuarios deben evitar estas aplicaciones basadas en *web*, al menos por ahora [14]. Las ventajas que esta tecnología ofrece son:

- Gastos de informática menores: No se necesita un ordenador de alta potencia y precio para procesar aplicaciones basadas en *web*. Dado que estas se ejecutan en la nube y no en el ordenador de escritorio, este no necesita la potencia de procesamiento o de espacio en disco duro que exige el *software*.
- Mejora del rendimiento: Con un menor número de programas que almacene la memoria del ordenador, se ve un mejor rendimiento del mismo. Como se utilizan aplicaciones de la nube, las computadoras serán más rápidas porque tienen menos programas y procesos cargados en memoria.
- Costos reducidos de *software:* En lugar de comprar costosas aplicaciones de *software*, se puede conseguir casi todo lo que se necesita de forma gratuita o a costos muy bajos.
- Las actualizaciones de *software* son instantáneas: Cuando la aplicación está basada en la *web*, las actualizaciones desde el punto de vista del usuario, se hacen de forma automática.
- Capacidad de almacenamiento casi ilimitada: Se pueden tener capacidades en el orden de los terabytes disponibles en la nube. Lo que se necesite para almacenar, se puede redimensionar a pedido y dinámicamente.
- Aumento de la fiabilidad de los datos: A diferencia de la computación de escritorio, en la que un fallo del disco duro puede destruir a todos los valiosos datos, si el disco del ordenador simplemente deja de funcionar eso no afecta a los datos. Si el ordenador personal se bloquea, todos los datos siguen en la nube, todavía accesibles. *Cloud Computing* utiliza las últimas tecnologías de copias de seguridad y redundancia de discos duros, además el cliente no se debe preocupar, la nube las realiza automáticamente.

- Acceso universal a los documentos: Los datos que están en la nube pueden accederse desde cualquier ordenador con conexión a Internet.
- Facilitar la colaboración en grupo: Compartir documentos conduce directamente a la colaboración en grupo. Múltiples usuarios pueden colaborar fácilmente en documentos y proyectos.
- Independencia del dispositivo: No se necesita comprar un dispositivo específico, un sistema operativo especial o un programa para un dispositivo en particular [18].

A pesar de todas estas ventajas la computación en la nube también presenta algunas desventajas como:

- Se requiere una conexión permanente a Internet: En el período que la red falle no se puede trabajar en la nube. También se dificulta en las zonas donde las conexiones son de mala calidad o poco fiables.
- No funciona bien con conexiones a baja velocidad: Una conexión a Internet a baja velocidad, por ejemplo servicios telefónicos (*modems*), hace que la computación en nube sea en muchos casos imposible. Las aplicaciones basadas en *web* requieren una gran cantidad de ancho de banda para descargarse, al igual que documentos de gran tamaño.
- Puede ser lento: Incluso con una conexión rápida, las aplicaciones basadas en *web* seguramente son más lentas que aplicaciones similares instaladas en su ordenador de escritorio. Esto se basa en muchas variables de las que depende el procesamiento en la nube, por ejemplo cada actualización tiene que ser enviada de ida y vuelta desde su ordenador hacia los servidores en la nube. Si los servidores de la nube en ese momento hacen una copia de seguridad, o si Internet está demasiado saturado nunca se tiene una respuesta instantánea como suele pasar con aplicaciones de escritorio.
- Teóricamente siempre existe la posibilidad de que los datos almacenados se puedan perder: La mayoría de las empresas que brindan el servicio toman precauciones para que esto no ocurra. Por ejemplo instalan, líneas de datos redundantes conectadas a cortafuegos físicos, sistemas de alimentación eléctrica ininterrumpidos, almacenamiento tolerantes a fallos, servicios de copias de seguridad automáticos, almacenamiento de las copias de seguridad en ambientes protegidos físicamente

(contra incendios o robos), pero no obstante, al ser medios físicos, nunca darán una seguridad del 100%.

1.4 Virtualización

La virtualización se diseñó por primera vez en 1960 para optimizar el uso de servidores. Fue IBM (*International Business Machines Corporation*) la que se encargó de transformar un servidor en distintas máquinas virtuales. Esto permitió alcanzar la multitarea, es decir ejecutar múltiples aplicaciones y procesos al mismo tiempo [19].

1.4.1 Virtualización de plataforma

La virtualización de plataformas involucra la simulación de una máquina real, la cual está alojada en un sistema anfitrión (*host*) y se ejecuta a través de un *software* llamado hipervisor, el cual controla que todas las máquinas virtuales sean atendidas correctamente. El hipervisor crea una capa de abstracción entre el *hardware* de la máquina física (*host*) y el sistema operativo de la máquina virtual, de tal forma que maneja los recursos de las máquinas físicas subyacentes para que el usuario pueda crear varias VMs, cada una de ellas presenta, una interfaz del *hardware* compatible con el SO elegido [20].

Existen dos tipos de hipervisores: Sin sistema anfitrión (*Non Hosted, ver* Figura II.1 Anexo II*),* el hipervisor trabaja directamente sobre el *hardware* físico de la máquina real, y el denominado con sistema anfitrión (*Hosted*, Figura II.2 del Anexo II), el cual corre sobre un sistema operativo anfitrión [21].

Dentro de este tipo de virtualización se encuentran tres subdivisiones:

Virtualización de aplicaciones

Las aplicaciones virtuales se ejecutan en un pequeño entorno virtual que actúa como una capa entre la aplicación y el sistema operativo, así se eliminan los conflictos entre aplicaciones y, además, entre aplicaciones y el SO.

Un caso de esta virtualización se ve en la máquina virtual usada por Java, que hoy en día se encuentra presente en muchos tipos de dispositivos. Otras VM de este tipo son: Ruby, Parrot (usada por Perl), *Smalltalk*, entre otras.

Virtualización de escritorio

El escritorio virtualizado es almacenado remotamente en un servidor central en lugar de almacenarlo en el disco duro local. Los usuarios trabajan en su escritorio remoto desde su PC (*Personal Computer*), portátil o un dispositivo móvil, *smartphone* o cliente ligero (*thin client*).

Virtualización de equipos y servidores

Permite ejecutar varias máquinas virtuales en un solo equipo anfitrión, ofrece la disponibilidad de diferentes servidores dentro de una misma máquina, se abstraen a los servidores virtuales del *hardware* real y permite asignarles los recursos necesarios, o incluso poder moverlos de un servidor físico a otro, en tiempo real [22].

La virtualización de servidores se divide en:

- **Virtualización completa**

Es una técnica que permite ejecutar sistemas operativos huésped mediante la simulación de todo el *hardware* del equipo anfitrión, sin tener que modificarlos, y utiliza como medio un hipervisor que permite compartir el *hardware* real.

Su principal ventaja es que los SO se ejecutan sin ninguna modificación sobre la plataforma, pero estos deben ser soportados por la arquitectura virtualizada [16].

La virtualización completa permite que el *hardware* simulado de cada máquina virtual sea independiente del equipo anfitrión.

Los sistemas operativos huéspedes (*guest*) corren en las máquinas virtuales y se ejecutan sobre el monitor del equipo virtual (hipervisor *o Virtual Machine Manager, VMM*) el cual se comunica directamente con el *hardware*, el tipo de hipervisor que se especifica es del tipo *Non Hosted,* ver Figura II.3 (a) del Anexo II [22].

Ejemplos de plataformas que ofrecen virtualización completa: *KVM, VMware Workstation*.

- **Paravirtualización**

La paravirtualización surge como una forma de mejorar la eficiencia de las máquinas virtuales y acercarlo al rendimiento del *hardware* nativo. Por eso, los sistemas virtualizados

huésped, deben estar basados en SO que se modifiquen para ejecutarse sobre un hipervisor. De esta forma no es necesario que este monitorice todas las instrucciones, sino que los sistemas operativos huésped y anfitrión colaboren en la tarea [16]. El modelo de paravirtualización, ver Figura II.3 (b) Anexo II, incluye APIs que asisten al monitor del equipo virtual [21].

En la virtualización completa, todo el sistema se emula (BIOS, disco, etc.), pero en la paravirtualización, su módulo de gestión opera con un sistema operativo que se adapta para trabajar en una máquina virtual. La paravirtualización suele funcionar mejor que el modelo de virtualización completa, simplemente porque en una implementación totalmente virtualizada (virtualización completa), todos los elementos deben ser emulados, a diferencia de la paravirtualización en la que solo el *hardware* subyacente es emulado. Ejemplos de paravirtualización: Xen, VMware ESX.

- **Virtualización híbrida**

La virtualización híbrida o también conocida como virtualización nativa. Constituye una combinación de la virtualización completa o de la paravirtualización en conjunto con técnicas de aceleración de I/O (Entrada/Salida) y memoria. Permite sacar ventaja de las últimas tecnologías de la arquitectura x86, Intel VT y AMD-V diseñadas para la virtualización asistida por hardware [21].

Su funcionamiento es similar al de la virtualización completa, es así que los sistemas operativos se pueden instalar sin una modificación previa [16].

1.4.2 Ventajas de los sistemas virtualizados

La virtualización ofrece grandes ventajas como ahorro energético pues se dejan de utilizar varios equipos que consumen energía, los mismos son substituidos por un solo servidor que aloja los servicios. Se ahorra espacio físico de almacenamiento ya que son menos cómputos y menos espacio independientemente del tipo de *hardware*. Se simplifica el manejo del entorno, desde un solo punto se puede ver el estado de todos los servidores físicos y virtuales, lo que implica facilidad de gestión. Otras de las ventajas son la facilidad de transportación, los servidores virtualizados se llevan de una máquina a otra y la rápida

recuperación tras las fallas, en un escenario físico se pueden tardar días o semanas en recuperar un entorno, con la virtualización esto puede ser un proceso de minutos.

Por otra parte se tienen virtualizados varios sistemas operativos en paralelo como: Windows, Linux, Novell, Solaris, entre otros, los cuales pueden funcionar simultáneamente. Con la creación de escritorios virtuales se puede tener una conexión local o remota a los usuarios. La virtualización permite tener los entornos de desarrollo, integración y producción que se separan perfectamente, por lo que son más efectivos [19]. La virtualización también presenta algunas desventajas, ver Tabla II.1 del Anexo II.

Las ventajas que ofrece la virtualización son una alternativa viable ante las limitaciones del esquema cliente servidor, ya que esta permite consolidar sistemas, cargas de trabajo y entornos operativos para obtener una mejor capacidad de respuesta y mejorar la flexibilidad de las TICs, mediante la creación de varios sistemas virtuales dentro de un único sistema físico con el particionamiento de *hardware*. Esta partición subdivide un servidor físico en fracciones donde cada uno puede correr con un sistema operativo conocido como máquina virtual las cuales son administradas por un gestor de VM lo que en su conjunto constituye un sistema de virtualización [19].

1.4.3 Hipervisores

El hipervisor es un *software* que permite que, múltiples sistemas operativos compartan el mismo *hardware* [20, 23].

XEN

Xen es una VM de código abierto desarrollada por la Universidad de Cambridge. Es un hipervisor que se ejecuta directamente en el *hardware*, permite que múltiples SO se ejecuten al mismo tiempo y en el mismo *hardware*, esto lo hace sin emulación o traducción de instrucciones.

Los sistemas operativos se modifican explícitamente para correr Xen (aunque mantiene la compatibilidad con aplicaciones de usuario). Esto permite a Xen alcanzar virtualización de alto rendimiento sin un soporte especial de *hardware*.

Intel realiza diversas contribuciones a Xen que le permite añadir soporte para sus extensiones de arquitectura VT-X *Vanderpool*. Esta tecnología permite que SO sin modificar actúen como nodo dentro de las máquinas virtuales de Xen, siempre y cuando el servidor físico soporte las extensiones VT de Intel o AMD.

Xen se optimiza para los servidores, ejecuta varios SO como Linux, Windows o Solaris, cada uno con su propio núcleo, de manera segura y limpia. Este *software* de virtualización se incorpora en algunas versiones de *Linux Enterprise*, *Red Hat*, *Mandriva*, *Debian*, *Fedora*. Además cuenta con algunas características como proporcionar aislamiento seguro, control de recursos, garantías de *QoS* y migración de máquinas virtuales en caliente [24].

KVM

KVM *(Kernel-based Virtual Machine)* es un *software* de código abierto, que representa una solución de virtualización completa para arquitecturas de *hardware* x86 que contengan extensión de virtualización (Intel-VT ó AMD V). KVM se implementa como un módulo en el *kernel* de Linux, se integra al mismo en la versión 2.6.20. Actualmente es el hipervisor de virtualización oficial del *kernel* de Linux.

Con KVM se puede ejecutar múltiples VMs que ejecutan imágenes de Linux o de Windows sin modificar. Cada máquina virtual tiene *hardware* virtualizado, una tarjeta de red, disco, adaptador de gráficos, etcétera [25].

KVM es capaz de aprovecharse de la comunidad en torno a Linux, cualquier mejora sobre el *kernel*, es una mejora que beneficia a KVM. Este hereda todos los controladores y el amplio soporte *hardware* de Linux, lo que le permite poder ejecutarse en cualquier plataforma x86 donde Linux lo haga.

Algunas de las características principales de KVM son:

Seguridad: Bajo KVM, cada VM se implementa como un proceso. El mismo se aprovecha del modelo de seguridad estándar de Linux: *SELinux/AppArmor*. Estos modelos proporcionan el aislamiento y el control de los recursos necesarios.

Gestión de memoria: KVM hereda las características de gestión de memoria de Linux. La memoria utilizada por una VM se gestiona de la misma forma que la de otro proceso, se guarda en disco (*swapped*) y se utiliza en páginas grandes (*large pages*). Permite también el

uso a las VMs de grandes cantidades de memoria. KVM soporta las últimas características de virtualización de memoria que proporcionan los fabricantes Intel y AMD. Estas tecnologías persiguen reducir el uso de la CPU y aumentar el rendimiento de los hipervisores.

Migraciones en caliente: KVM permite migraciones en caliente (*live migrations*), esta característica permite mover una VM en ejecución entre servidores físicos (hipervisores) sin interrupción del servicio. Estas migraciones son transparentes al usuario, la VM permanece encendida, las conexiones de red activas y las aplicaciones en ejecución mientras que la máquina se realoja en un nuevo servidor físico.

Rendimiento y escalabilidad: KVM posee los mismos rasgos de rendimiento y escalabilidad que caracteriza a Linux. Soporta VM de hasta 16 CPUs virtuales y 256 GB de RAM [26].

VMWare

VMware es un sistema de virtualización por *software*, es decir es un programa que simula un sistema físico (un ordenador, un *hardware*) con unas características de *hardware* determinadas. Cuando se ejecuta el programa (simulador), proporciona un ambiente de ejecución similar a todos los efectos a un ordenador físico (excepto en el puro acceso físico al *hardware* simulado), con CPU (puede ser más de una), BIOS, tarjeta gráfica, memoria RAM, tarjeta de red, sistema de sonido, conexión USB, disco duro (pueden ser más de uno), etcétera [27] [28].

Un virtualizador por *software* permite simular varios sistemas operativos dentro de un mismo *hardware* de manera simultánea, lo que permite un mayor aprovechamiento de recursos. No obstante, y al ser una capa intermedia entre el sistema físico y el SO que funciona en el *hardware* emulado, la velocidad de ejecución de este último es menor, pero en la mayoría de los casos es suficiente para usarse en entornos de producción [29].

VMware ofrece las mejores soluciones y mayor confianza para transformar entornos de TI en infraestructuras flexibles y automatizadas de nube, esto ayuda a que el cliente obtenga ahorros de costes y beneficios de productividad [30].

LXC

LXC (Contenedores de Linux, *Linux Containers*) es algo parecido pero no es una máquina virtual, es más bien un entorno virtual que provee su propio entorno de procesos y redes. A diferencia de las VMs, que necesitan un anfitrión de *hardware* con mucha memoria RAM, los LXC no consumen apenas recursos del anfitrión [31].

LXC aprovecha una funcionalidad del núcleo Linux llamada ***cgroups*** (grupos de control del kernel), que está presente en el *kernel* desde la versión 2.6.29 (y por lo tanto en todas las versiones modernas 3.x). LXC es por lo tanto virtualización a nivel del SO [32].

El mismo permite tener una distribución de Linux dentro de otra, que utiliza el *kernel* del anfitrión y la tarjeta de red, pero trabaja de manera encapsulada a todos los efectos, con su propia IP de red, su propia interfaz de red (encapsulada) y sus propios procesos. Esto permite tener dentro de un servidor Linux, decenas o centenares de contenedores, cada uno con su configuración, distribución, versión, etcétera, y perfectamente encapsulados. Se pueden administrar en remoto fácilmente, por ejemplo, mediante SSH [33].

Su interés a nivel empresarial es evidente, por la reducción de costes que supone frente a las máquinas virtuales y la facilidad de implementación así como de administración. Además de su gran utilidad para hacer pruebas de seguridad en redes de todo tipo. LXC tiene como beneficios el aislamiento de aplicaciones y sistemas operativos a través de contenedores. También gestiona la asignación de los recursos en tiempo real y el control de las interfaces de red y la aplicación de los recursos dentro de los contenedores.

A pesar de las ventajas que tiene posee como limitaciones que todos los contenedores LXC se ejecutan en el interior del núcleo del sistema anfitrión y no con un núcleo diferente. Este hipervisor sólo permite al sistema operativo Linux. La seguridad depende en el sistema anfitrión, LXC no es seguro, si se necesita un sistema seguro, se recomienda utilizar KVM [34].

1.5 Conclusiones

La Universidad Central Marta Abreu de las Villas (UCLV) cuenta con dos sistemas de virtualización: a) VMWare ESX: sistema propietario y b) Proxmox VE: sistema basado en *software* libre. Ambas plataformas de virtualización forman clústeres por separados, lo cual

implica que los especialistas necesiten dominar ambas tecnologías que emplean comandos, interfaces de administración e hipervisores completamente diferentes. De ahí a que convertir una máquina virtual de un sistema a otro lleve tareas adicionales.

Por este inconveniente se propone la implementación de una IaaS, ya que el usuario gestiona mediante una aplicación *web* todos los recursos necesarios y despliega por sí mismo las máquinas virtuales requeridas. Se desea simular un ambiente similar al existente en la UCLV. Los hipervisores escogidos para hacer las pruebas necesarias son: a) KVM, virtualización completa y b) LXC, virtualización de contenedores. Los servidores tienen como sistema operativo Ubuntu 14.04 LTS/Debian Jessie.

Los servicios se brindan a través de una nube privada, ya que la infraestructura que se gestiona internamente en la entidad. En este caso el proveedor de servicio será la UCLV y los clientes, los usuarios que pertenecen a la red interna de la institución.

Para crear la nube IaaS se necesita una herramienta de *software* que sea capaz de gestionar y administrar eficientemente todos los recursos, hoy en día existen varias, por esto hay que hacer un análisis de la que se va a utilizar. En el capítulo 2 se hace una comparativa para seleccionar cuál de estas herramientas es más flexible, eficiente y de fácil instalación y despliegue para el personal especializado.

CAPÍTULO 2. HERRAMIENTAS DE CÓDIGO ABIERTO PARA LA GESTIÓN EFICIENTE DE NUBES

La implementación de una Infraestructura como Servicio, permite el desarrollo de plataformas y *software* computacional, su capacidad proporciona al usuario funciones de red, informática y recursos de almacenamiento en las que se pueden instalar y ejecutar *software* arbitrario en los servidores del proveedor, o a su vez *software* como sistemas operativos y aplicaciones pueden estar incluidos dentro del acuerdo de servicio IaaS. También le facilita al usuario tener un control total sobre los sistemas operativos y aplicaciones desplegadas en IaaS [17].

Los consumidores, desarrolladores y proveedores de la nubes necesitan tomar la decisión sobre qué plataforma es más adecuada para implementar su infraestructura. En las últimas décadas, la tecnología de código abierto ayuda a los clientes que no desean utilizar la infraestructura comercial para la nube. Existen diferentes plataformas de código abierto, entre ellas OpenNebula, Eucalyptus, OpenStack y CloudStack son las que mayormente han sido empleadas para la gestión de la nube, así como las alternativas que proveen nubes comercialmente. A continuación, se comparan y analizan las herramientas de *software* anteriormente mencionadas las cuales proporcionan una plataforma de gestión de nubes IaaS [35].

2.1 Principales características de OpenNebula, Eucalyptus, OpenStack y CloudStack

Las plataformas de gestión de nubes comerciales realizan cargos por la cantidad de horas usadas, el tiempo de CPU, por requerimientos de almacenamiento y ancho de banda de red. Si la organización tiene gran número de usuarios, es mejor comprar su propio *hardware* y

crear una nube privada. Aquí es donde las plataformas de código abierto OpenNebula, Eucalyptus, OpenStack, y CloudStack entran en ~~la~~ escena, que permiten la creación de nubes sin necesidad de licencias [35].

La Figura 2.1.1 muestra la arquitectura genérica de los sistemas de computación en la nube, estos tienen seis componentes básicos.

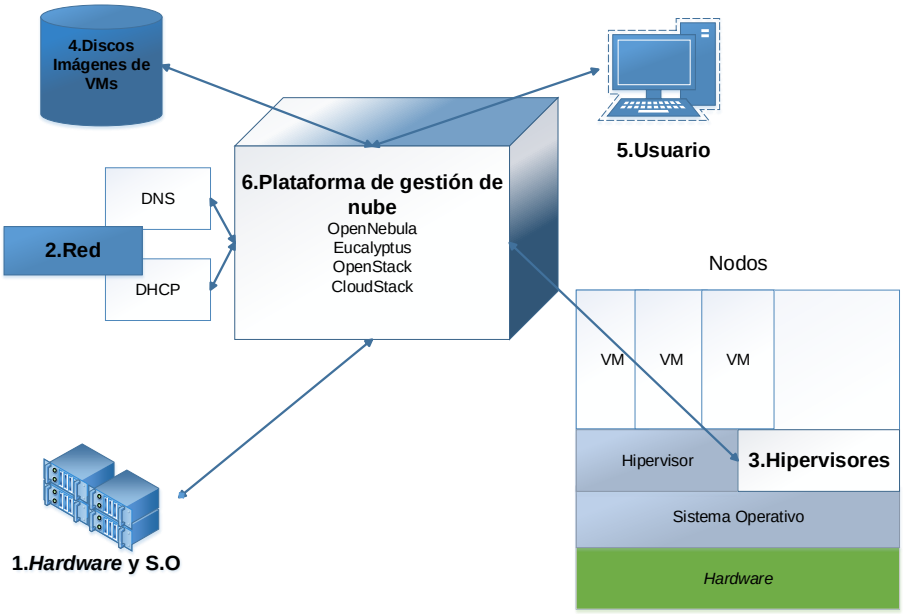

Figura 2.1.1 Arquitectura de las plataformas de gestión de nubes.

Como se puede observar en la Figura 2.1.1, los componentes básicos de una plataforma de gestión de nubes son:

1. *Hardware* y sistemas operativos.
2. Red: incluye DNS, DHCP y la organización de las máquinas físicas de la subred en dependencia de la plataforma de gestión de la nube.
3. Los hipervisores: Se encuentran entre el SO y las máquinas virtuales. Los diferentes tipos de hipervisores son Xen, KVM, VMWare, etcétera.
4. Imágenes de disco de VM: Pueden ser imágenes de plantillas de discos, que se utilizan para crear varias máquinas virtuales en la plataforma de la nube, o imágenes en tiempo de ejecución, en realidad utilizados por las VMs en tiempo de ejecución.

5. Interfaz de usuario *Frontend*: Hay básicamente dos tipos de interfaces, proveedor o administrador de la nube y el consumidor.

6. Plataformas de gestión de la nube: Las más comunes son OpenNebula, Eucalyptus, OpenStack, CloudStack. Esta ofrece la interfaz en el *Frontend*, utiliza las imágenes de disco a partir de las señales de repositorio de VMM para configurar la VM y luego mediante el DHCP asigna y configura direcciones IP a cada VM.

2.1.1 Eucalyptus

Eucalyptus es una arquitectura *software* de código abierto basada en Linux que implementa nubes privadas e híbridas. Ofrece IaaS de tal forma que los usuarios pueden provisionar sus propios recursos en función de sus necesidades. Se diseñó para ser fácilmente instalada y desplegarla de la forma menos instructiva posible [36].

Eucalyptus proporciona una capa de red virtual de tal forma que se aísla el tráfico de red de diferentes usuarios y permite que uno o más clústeres parezcan pertenecer a la misma Red de Área Local (LAN).

También proporciona soporte para distintos hipervisores (Xen y KVM). Tiene la peculiaridad de crear un entorno similar a *Amazon Web Site*, por lo que el usuario familiarizado con la utilización del CLI de *Amazon* podrá emplear Eucalyptus de forma inmediata. Además, tiene la capacidad de interactuar con *Amazon EC2* y los servicios *S3* (*Simple Storage Service*) de nube pública y ofrece la posibilidad de crear un nube híbrida.

Eucalyptus es fácil de instalar, y la documentación proporcionada por el desarrollador es completa. Sin embargo, se puede señalar algunos puntos como, por ejemplo, que la interfaz *web* que se suministra tiene un número muy limitado de funcionalidades implementadas y ello supone que cualquier usuario no experimentado debería emplear el interfaz CLI para iniciar sus VMs, o que es difícil crear una máquina virtual nueva con un *kernel* no suministrado por Eucalyptus. Además, hay que señalar que Eucalyptus no dispone de sistema de monitorización propio del estado de los nodos, por lo que sería necesario recurrir a herramientas de terceras partes como, por ejemplo, Nagios [37, 38].

Hay que señalar que Eucalyptus tiene una funcionalidad que no se encuentra en otros gestores de nubes y es la posibilidad de poner en reposo aquellos nodos de computación

que no tengan VMs en funcionamiento, lo que contribuye a racionalizar el consumo energético.

2.1.2 CloudStack

CloudStack es una arquitectura *software* de código abierto que efectúa el despliegue, la configuración y la gestión de entornos de computación elástica. Permite construir cualquier tipo de nube (privada, pública e híbrida) y soporta los hipervisores Xen Server y KVM [38].

CloudStack es desarrollado por *Cloud.com* y proporciona tres versiones diferentes:

1. *CloudStack Community Edition*: código abierto, soportado por la comunidad.
2. *CloudStack Enterprise Edition*: emplea código abierto y código propietario. Fue diseñado para empresa y se distribuye de forma comercial.
3. *CloudStack Service Provider Edition*: Emplea código abierto y código propietario. Fue diseñado para los proveedores de servicios y se distribuye de forma comercial.

CloudStack es fácil de instalar. Sin embargo, se ha detectado que la documentación oficial disponible es deficiente, ya que hay numerosas opciones de configuración del entorno que no se encuentran documentadas. Esto dificulta de forma considerable la realización de pruebas y la puesta a punto del entorno, ya que hay distintas opciones parametrizables cuyos valores pueden ser incompatibles entre sí.

Este gestor se destaca especialmente por su interfaz *web* que ofrece una gestión completa de la nube tanto para el administrador del sistema como para un usuario no privilegiado. Además, aporta gran cantidad de información, como la monitorización, las estadísticas de utilización de los recursos, la información del registro y las alertas. CloudStack dispone de opciones que hasta el momento no se habían observado en otros gestores de nube. Por ejemplo, la posibilidad de definir máquinas virtuales de alta disponibilidad que el sistema mantendrá en funcionamiento sin intervención del usuario o del administrador del sistema, la posibilidad de efectuar la instalación por medio de la interfaz *web* de una nueva máquina virtual mediante una imagen ISO de instalación, la posibilidad de efectuar balance de carga entre máquinas virtuales; la posibilidad de acceder a la máquina virtual en modo gráfico por medio del interfaz *web* [39].

Uno de los puntos más discutibles se refiere a la gestión de la infraestructura *hardware*. CloudStack, en función del modo de red seleccionado, puede llegar a ser muy rígido a la hora de efectuar cambios en la infraestructura, lo que impide, por ejemplo, modificar de forma sencilla el rango de direcciones IP a utilizar por las máquinas virtuales [40].

CloudStack es un gestor muy recomendable en aquellos entornos en los que es imprescindible efectuar la gestión de las máquinas por medio de interfaz *web*.

2.1.3 OpenStack

OpenStack es una plataforma de código abierto, simple y escalable, avalada por *Rackspace* y la Administración Nacional de Aeronáutica y del Espacio **(**NASA), que aporta la plataforma Nebula, bajo licencia Apache 2.0.

Rackspace aportó el código que potencia el servicio de entrega de contenidos y almacenamiento de los archivos de la nube (*Cloud Files*) y los Servidores de producción en la Nube (*Cloud Servers*). La NASA dio la tecnología que soporta a Nebula, su propio servicio de computación en la nube, con características de alto rendimiento, trabajo en redes y eficiente gestión del almacenamiento de datos, para lograr la gestión de grandes conjuntos de datos científicos [41].

Es también una comunidad de proveedores de servicios de nube y fabricantes de tecnología (como *Red Hat, AT&T, Canonical, Cisco, Dell, GoDaddy, HP, IBM, Intel, Rackspace Hosting, Nexenta, AMD, Suse, VmWare, Oracle, Yahoo*, entre otras.) que se dedica al desarrollo de infraestructuras de *software* libre para arquitecturas de nubes públicas, privadas e híbridas.

OpenStack cuenta con otros servicios como: *OpenStack Compute*: que permite gestionar el despliegue y ejecución de aplicaciones a través de múltiples servidores. *OpenStack Object Storage*: gestiona el almacenamiento de datos en varios servidores que trabajen de manera conjunta en clústeres, para conseguir un almacenamiento masivo de objetos estáticos, de manera superflua y fiable.

La tecnología Nova, se basa en el protocolo de mensajería AMQP y es el sistema utilizado en la NASA para proveer sistemas de virtualización bajo demanda [41].

Además, por su manejo fácil, una persona puede implementar en su hogar su propia nube. OpenStack tiene una gran capacidad de extensibilidad mediante APIs que son fáciles de implementar y adaptar (muy al estilo de Amazon), públicas y del tipo libre de vendedor, por lo que muchos proveedores de servicio han volteado a ver a OpenStack como una alternativa clave para sus propias iniciativas de infraestructura en la nube. OpenStack con su tecnología modular en base a los requerimientos de "Nube" que se necesiten entregar permite integrar distintos proyectos a la arquitectura que se crea de forma progresiva y estable.

El módulo de cómputo de OpenStack soporta varios hipervisores como: KVM, LXC, QEMU, VMware, Xen, PowerVM, Hyper-V, entre otros [42].

2.1.4 OpenNebula

OpenNebula es un *software* código abierto que permite construir cualquier tipo de nube: privada, pública e híbrida. Se diseña para ser integrado con cualquier tipo de red y almacenamiento, para así adaptarse a cualquier centro de datos existente. Gestiona el almacenamiento, las redes y las tecnologías de virtualización. Proporciona la posibilidad de desplegar servicios en infraestructuras distribuidas, donde combina recursos de centros de datos así como de nubes remotas, de acuerdo con las políticas de despliegue [43].

Este proyecto fue inicialmente desarrollado por la Universidad Complutense de Madrid en 2008. Más adelante el número de participantes crece y más organizaciones se unen al desarrollo del mismo. Algunos destacados contribuidores son IBM, Suse y AT&T [37].

OpenNebula ofrece la solución más sencilla, pero rica en características y flexible para la gestión integral de los centros de datos virtualizados que permite crear nubes privadas, públicas e híbridas IaaS. Su interoperabilidad hace que la nube evolucione mediante el aprovechamiento de los activos de TI existentes, protege sus inversiones, y evita la dependencia de un proveedor.

Este gestor es una solución lista para la empresa que incluye todas las características necesarias para proporcionar una oferta de nube (privada) en las instalaciones, y para ofrecer servicios en la nube pública. Esta herramienta es compatible con Xen, KVM y

VMware y *Amazon EC2*. Es posible integrarlo con LXC, aunque no viene incorporado de forma predeterminada.

Cloud Computing empresarial es el siguiente paso en la evolución de la virtualización del centro de datos. OpenNebula combina tecnologías de virtualización existentes con características avanzadas para múltiples tendencias, provisión automática y elasticidad, que siguen en desarrollo ascendente impulsado por las necesidades reales de los administradores de sistemas y *DevOps*.

El proyecto *OpenNebula.org* persigue los siguientes objetivos con el fin de conducir la innovación en la gestión del centro de datos de nube de clase empresarial [44]:

- Desarrollar una solución más sencilla, potente, altamente escalable y adaptable para la creación y gestión de centros de datos virtualizados y nubes de una empresa.
- Proporcionar constructores de nubes e integradores con un sistema modular que se puede implementar una variedad de arquitecturas de nube y puede interactuar con cualquier servicio de centro de datos.
- Asegurar la estabilidad y la calidad de la distribución de *software*.
- Colaborar con los usuarios más exigentes de herramientas de gestión de la nube y centros de datos.
- Apoyar el ecosistema de componentes de código abierto que se ha creado en torno al proyecto.
- Apoyar a la comunidad de usuarios y desarrolladores que contribuyen a OpenNebula.
- Colaborar con los principales proyectos de investigación en innovación de computación en la nube.

2.2 Comparación entre los gestores

2.2.1 Según las perspectivas del usuario y del desarrollador

Desde la perspectiva de desarrollador para conocer qué tan abierto es el código **s**e sugiere usar las siguientes medidas:

- Modelo de Desarrollo**:** Es el código que se desarrolla a través de Internet a la vista del público.
- Compromiso del desarrollador: Es el desarrollo abierto a contribuciones externas.
- Modelo de Gobierno: Forma en la que se toman las decisiones en el plan de trabajo.

Existen distintos modelos de gobiernos dentro de los que se encuentran reglas de meritocracia (*Meritocratic*) donde los participantes adquieren influencia sobre la toma de decisiones de sus contribuciones. Por otra parte está el dictador benévolo (*benevolent dictator*) donde mantienen un estricto control sobre el proceso de toma de decisiones, un dictador benevolente es responsable de determina las decisiones finales cuando la comunidad está en desacuerdo.

Las cuatro plataformas de gestión de nubes son totalmente *software* de código abierto, aceptan contribuciones en virtud de acuerdos de licencia similares, y se desarrollan públicamente a través de Internet. Sin embargo, hay una diferencia en sus modelos de gobierno. Mientras OpenStack sigue un método básico con un Consejo de Administración que proporciona supervisión estratégica y CloudStack sigue las reglas de la meritocracia Apache; Eucalyptus y OpenNebula son gestionados por una sola organización que se centra en el interés del proyecto y estratégicamente lleva a asegurar que cumple con las necesidades de los usuarios y la comunidad. Gobierno dictador benévolo es el modelo que siguen por otros proyectos de éxito como Android o Linux Kernel, es la forma más eficaz para centrarse en la calidad de la ingeniería, para responder a los usuarios, y asegurar el apoyo a largo plazo.

Por otra parte está la perspectiva de los usuarios. En este caso, hay que tener en cuenta tanto la perspectiva del usuario consumidor como del creador de la nube.

Desde la perspectiva del consumidor de “nube abierta” es todo acerca de las APIs y formatos de datos. Da libertad de API comunes para ejecutar en cualquier lugar, es esta la libertad que se soporta o no por el código abierto. Esto proporciona la capacidad para que el usuario compare ofertas de nube, seleccione la oferta que mejor se adapte a sus necesidades, y cambie de proveedor si no está satisfecho con el servicio o si encuentra una oferta más competitiva.

Tabla 2.1 Comparación según la perspectiva del desarrollador [35].

	OpenStack	**CloudStack**	**Eucalyptus**	**OpenNebula**
Modelo del desarrollador	Pública al desarrollador	Pública al desarrollador	Pública al desarrollador	Pública al desarrollador
Compromiso del desarrollador	Contrato de licencia de colaborador	Contrato de licencia de colaborador	Contrato de licencia de colaborador	Contrato de licencia de colaborador
Modelo de gobierno	Fundación	Técnica meritocracia	Dictador benévolo	Dictador benévolo

Desde la perspectiva del constructor de nube abierta significa que el *software* de código abierto de la comunidad es de nivel empresarial y apoyado comercialmente sin tener que instalar una distribución mejorada del proveedor (que sería mucho más cerca de un modelo de núcleo abierto). Aquí es donde los compradores y usuarios de tecnología pueden evaluar por sí mismos la apertura.

Se sugiere utilizar las siguientes medidas:

- Ecosistema API: Se refiere a que si el *software* soporta un estándar efectivo con un amplio ecosistema.
- Preparación para producción: El *software* de código abierto está listo para uso de la empresa y comercialmente soportado.

Tabla 2.2 Comparación según la perspectiva del usuario [35].

	OpenNebula	**CloudStack**	**Eucalyptus**	**OpenStack**
Ecosistema API	API Amazon	API Amazon	API Amazon	API OpenStack
Preparación para producción	Lista para la empresa y con apoyo directo de los desarrolladores	Lista para la empresa y con apoyo directo de los desarrolladores	Lista para la empresa y con apoyo directo de los desarrolladores	No, solo está disponible a través de los proveedores específicos de pila

No hay mucho más que decir acerca de los ecosistemas de API de nube, no se profundizará en cuál de las API de nube es un estándar efectivo, ni qué ecosistema es más grande y con más rápido crecimiento**.**

- Ecosistema API: Se refiere a que si el *software* soporta un estándar efectivo con un amplio ecosistema.
- Preparación para producción: El *software* de código abierto está listo para uso de la empresa y comercialmente soportado.

Desde esta perspectiva, es evidente que el Eucalyptus y OpenNebula son más abiertos. Ambos proyectos ofrecen una solución de nube de código abierto listo para la empresa. Cualquier organización puede usar la distribución de código abierto para construir una nube de producción, y recibir el apoyo de mejor esfuerzo a través de una lista de correo de la comunidad. Además, cualquier organización puede adquirir soporte comercial directamente de los desarrolladores.

El aspecto importante está en que estos proyectos no entregan las ediciones de empresa de su *software*, y lo apoyan en una comunidad comercial. En otras palabras, las versiones de la comunidad de Eucalyptus y OpenNebula no son ediciones limitadas de versiones empresariales. CloudStack podría ser también incluido en este grupo, dado que *Citrix CloudPlatform* es básicamente una distribución empresarial que no proporciona funciones ampliadas.

Se trata de mostrar cómo los cuatro proyectos de código abierto tienen distintas culturas y conductores, esto se reflejan en las dimensiones de la apertura. Por ejemplo, las cuatros implementan diferentes modelos de gobierno porque se abordan diferentes necesidades. Mientras Eucalyptus y OpenNebula satisfacen las necesidades de los usuarios, CloudStack sirve mejor a las necesidades de los desarrolladores, y OpenStack sirve a las necesidades de los vendedores, por lo que tienen una base de tecnología y una marca de *marketing* para construir sus propios conjuntos de nubes.

2.2.2 Según el modelo de la nube

Existen dos modelos nubes diferentes:

1. La virtualización del centro de datos
2. Provisión de Infraestructura

En la Tabla 2.3 se muestran algunas características que diferencias a los dos modelos de nubes anteriormente mencionados.

Tabla 2.3 Virtualización de centros de datos y provisión de infraestructura.

	Virtualización de centros de datos	**Provisión de infraestructura**
Aplicaciones	Aplicaciones de múltiples niveles definidos de manera tradicional, "Empresa"	"Rediseñado" aplicaciones para colocar dentro del paradigma de la nube
Interfaces	API rico en funciones y portal de administración	API de nubes simples y portal de autoservicio
Capacidades de gestión	Gestión completa del ciclo de vida de los recursos virtuales y físicos	Gestión simplificada del ciclo de vida de los recursos virtuales y abstracción de la infraestructura subyacente
Implementación de nube	Mayormente privadas	Mayormente públicas
Diseño interno	Diseño de abajo hacia arriba dictado por la gestión de la complejidad del centro de datos	Diseño descendente dictada por la aplicación eficiente de las interfaces de la nube
Capacidades de empresa	Alta disponibilidad, tolerancia a fallos, replicación, programación, entre otras, proporcionadas por la plataforma de gestión de la nube	La mayoría de ellos creados en la aplicación, como en "diseño para el fracaso"
Integración del centro de datos	Fácil de adaptación para cualquier entorno de la infraestructura existente para aprovechar las inversiones en TI	Construido sobre nueva infraestructura homogénea y comerciable

La Figura 2.2.1 sugiere claramente que OpenNebula y Eucalyptus como están en distintos cuadrantes del gráfico responden a necesidades diferentes e implementan filosofías opuestas. OpenNebula se inclina más para la virtualización de centros de datos pues posee un API rico en funciones y un portal de administración, además ofrece alta disponibilidad y tolerancia a fallos.

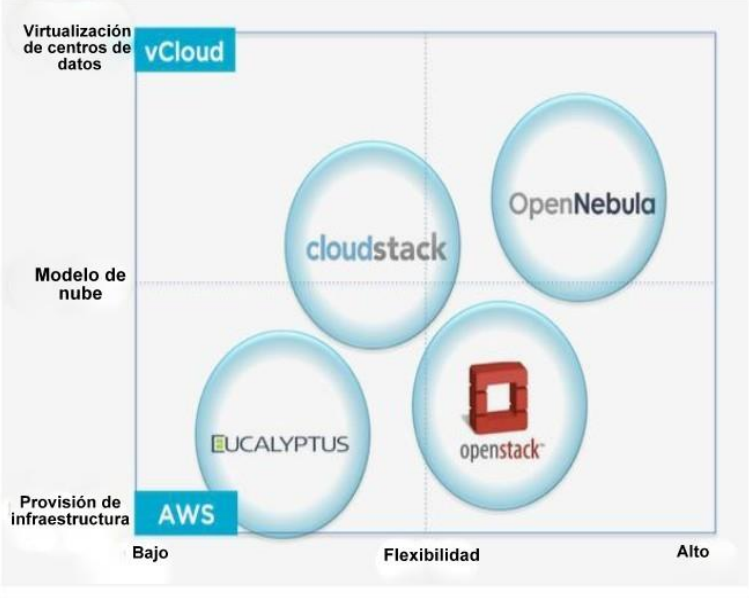

Figura 2.2.1 Virtualización de centros de datos y Provisión de infraestructura.

En cuanto OpenStack, OpenNebula representan soluciones flexibles que se pueden adaptar a las necesidades, excepto que equivocadamente se piensan que ambas permiten el mismo tipo de nube pues OpenStack es más bien para la provisión de infraestructura.

Eucalyptus y OpenStack se responsabilizan por la misma necesidad y así también compiten por el mismo tipo de nube.

Al analizar las diferentes plataformas de gestión de nube de código abierto, se observa que hay diferencia al diseñar la implementación con cada una de ellas. OpenNebula y CloudStack se inclinan más hacia la virtualización del centro de datos y Eucalyptus y OpenStack por la provisión de infraestructura. Entre los cuatro OpenNebula es más flexible en comparación con todas las demás plataformas.

En la Tabla 2.4 se reflejan algunas características de los distintos gestores de nubes. En esta se observa que los cuatro son de código abierto, escalables, permiten la implementación de una IaaS, como otros rasgos que tienen en comunes entre ellas. Por otro parte se aprecia que soportan diferentes interfaces e hipervisores, presentan diferencias en las redes, el almacenamiento, autentificación, entre otras.

Tabla 2.4 Característica de las plataformas de gestión de nubes.

	OpenNebula	**Eucalyptus**	**OpenStack**	**CloudStack**
Tipo de Servicio	IaaS	IaaS	IaaS	IaaS
Escalabilidad	Escalable	Escalable	Escalable	Escalable
Interfaz	*OCCI, EC2 Query, EBS, UNIX-like CLI, interfaz simple y avanzada Sunstone, Ruby, Java, XML-RPC API*	EC2, S3, EBS, *Rest Interface*	EC2, S3, *RestFul API*	Fácil uso con interfaz la *Web*, CLI, *Restful API*, y compatible con EC2 y S3
Hipervisor	XEN, KVM, VMware y LXC (a través de un complemento de tercero)	VMware (ESX/ESXi), KVM, XEN	KVM, XEN, LXC, UML, VMware	VMware, KVM, XenServer, XEN, LXC
Red	1. Red de servicio 2. Red de clientes (*dummy*, FW, 802.1Q, *Open vSwitch* y VMware)	IP Elástica, grupos de seguridad, servidor DHCP	1. Redes planas 2. Redes VLAN	Redes VLAN, VMware, Firewall, direcciones IP del sistema reservados
Implementación del DevOps	Chef, Puppet	Chef, Puppet	Chef, Puppet, Crowbar	Puppet
Autentificación	Usuario/Contraseña, SSH, X509, LDAP	LDAP, CHAP	X509, LDAP	LDAP y autentificación de usuario, SSH
Licencia	Código abierto Apache	Código abierto + Comercial	Código abierto Apache	Código abierto Apache
Almacenamiento (transferencia de imagen)	Sistema, Imagen y almacenamiento de archivos de datos SAN/NAS Server, vmfs, LVM, Ceph	Walrus (http/s)	Swift (http/s), Unix File System (ssh)	VHD, OVA, QCOW2
Frecuencia promedio de edición	Más de 6 meses	Más de 4 meses	Menos de 4 meses	Más de 4 meses
Implementación del *software*	Instalado en interfaz	Compuesto por módulos. Los nodos necesitan instalar el *software* Eucalyptus	Compuesto por módulos. Los nodos necesitan instalar el *software* OpenStack	Compuesto por módulos. Los nodos necesitan instalar el *software* CloudStack

El sistema OpenNebula al ser completamente abierto permite una interoperabilidad con los componentes de las infraestructuras actuales. Esta plataforma no depende exclusivamente de ningún tipo de hipervisor, igualmente no requiere de determinados recursos, pues puede adaptarse a infraestructuras ya existentes del volumen y capacidades que se deseen.

Como estructura IaaS, se compone de una serie de herramientas (CLI, Planificadores, etc.), un Núcleo (*Core*) donde se gestionan las peticiones vía *XML-RPC*, igualmente en el mismo tiene lugar la gestión de las máquinas virtuales y los *pools* de bases de datos SQL.

Finalmente en una capa inferior se encuentran los controladores que son la unión directa entre la capa *hardware* y el núcleo de OpenNebula. En esta capa se encuentran los controladores de Almacenamiento (*NFS, ISCSI, SSH*), de máquinas virtuales (dependientes de los hipervisores) y de Información (proporcionan información de las máquinas y depende también de los hipervisores). Por otra parte la integración y desarrollo del API de OpenNebula permite programar en *Java*, *Ruby* o en *XML-RPC* y se puede extender o integrar el entorno de la nube según se desee.

Dentro de los hipervisores soportados dentro de la estructura de OpenNebula se pueden identificar soluciones libres como *Xen*, *KVM*, *QEMU*/*KVM*; también es soportado LXC pero no de forma nativa. Igualmente soporta *libvirt* como capa de abstracción entre *KVM*/*XEN*. Dentro de los hipervisores no libres soporta VMWARE en todos sus productos *(VMWare ESXI, ESX, Server*).

OpenNebula proporciona a los diferentes usuarios de la plataforma herramientas de gestión para: recursos flexible y despliegue de máquinas pre-configuradas, usuarios (uso, facturación, etcétera), perfiles de seguridad, redes (subsistema propio de redes con independencia entre las máquinas virtuales completa, VLANs), almacenamiento (en red, local, en la nube), alta disponibilidad y clústeres, Zonas, Virtual Centers (VDCs), Nubes Híbridas (*Amazon EC2*), entre otras.

Estas funcionalidades convierten a OpenNebula en un completo gestor de VDC (*Virtual Data Center*) que permite gestionar desde la capa más básica de red y almacenamiento hasta los procesos de gestión de usuarios, tiempos de uso y explotación de los recursos además de su escalado bajo demanda.

2.2.3 ¿Por qué OpenNebula?

Es importante establecer claramente lo abierto, simple, escalable, y flexible que es OpenNebula. A continuación se muestran algunas razones por las que se escoge esta herramienta de gestión de nube [45]:

- Potente e innovador: es la más avanzada e innovadora de las plataformas de gestión de nubes, con funcionalidad en la clase empresarial para la gestión de los centros de datos virtualizados y para construir nubes privadas e híbridas.
- Adaptable, extensible e integrable: ofrece abiertas, adaptables y extensibles arquitectura, interfaces y componentes para construir un servicio en la nube y hace que las operaciones en la nube se ajusten a las políticas existentes.
- Interoperable: proporciona interoperabilidad y portabilidad a los consumidores con la elección de la nube a través de los estándares e interfaces que más se usan.
- Totalmente de código abierto: no tiene propiedad ni desarrollo de una edición limitada de una versión empresa, es verdaderamente código abierto, distribuido bajo licencia Apache.
- Simple: A pesar de su sofisticación técnica y funcionalidad avanzada, es fácil de descargar, instalar y actualizar.
- Estable y probado: Se prueba con rigor a través de un proceso de garantía de calidad interna y por una gran comunidad con una escalabilidad, fiabilidad y rendimiento que se prueba en muchas implementaciones escalable y robusta de producción.
- Maduro: Desarrollo que se impulsa por las necesidades del usuario y madura a través de muchos ciclos de lanzamiento.
- Producto de clase empresarial: OpenNebula comprende todas las funcionalidades clave para *Cloud Computing* empresarial, almacenamiento y redes en una sola instalación, asegura su estabilidad a largo plazo y el rendimiento a través de un único proceso de actualización de parches integrados.
- Soporte en un solo sitio: Gran variedad comercial y apoyo de los desarrolladores de la comunidad OpenNebula.

Independientemente de esto también se selecciona esta herramienta por las facilidades que le brinda el proyecto de cooperación entre universidades cubanas RedTIC con apoyo del VLIR-OUS, ya que se pueden hacer intercambios con especialistas de alto nivel de la Universidad de Ghent en Bélgica, donde hay personal especializado en el tema con la disposición de ayudar a la instalación y despliegue de esta plataforma. También se impartió un curso durante este año enfocado a HPC y unos de los temas tratados fue OpenNebula, el cual facilitó de antemano tener conocimiento de cómo funciona la misma.

2.3 Descripción de la arquitectura de OpenNebula

Se asume que la infraestructura física de OpenNebula adopta una arquitectura de clúster clásica con un *Frontend*, y un conjunto de nodos, donde se ejecutarán las máquinas virtuales. Hay por lo menos una red física para comunicar a todos los anfitriones con el *Frontend,* Figura 2.3.1 [46].

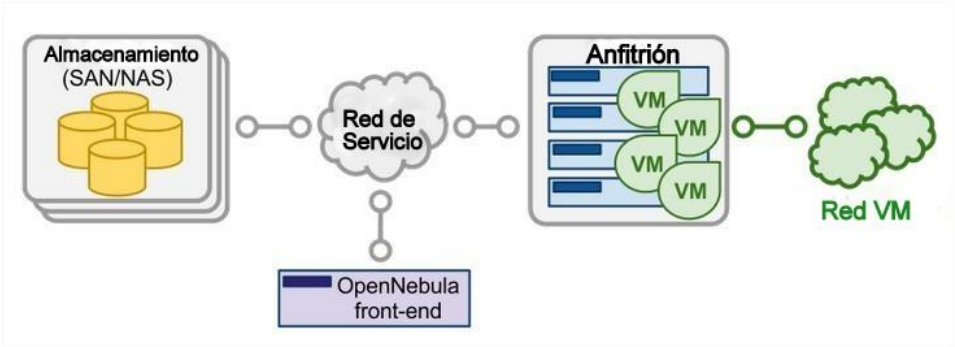

Figura 2.3.1 Arquitectura de OpenNebula [47].

Los componentes básicos de un sistema de OpenNebula son:

- ***Frontend***: ejecuta los servicios OpenNebula.
- **Anfitriones** (se pueden ver también como *host* o nodo)**:** Hipervisor habilitado, que proporcionan los recursos necesarios para las máquinas virtuales.
- **Almacenes de datos (*Datastore*)**: Contienen las imágenes base de las máquinas virtuales.

- **Redes:** Redes físicas que se utilizan para apoyar los servicios básicos tales como la interconexión de los servidores de almacenamiento y operaciones de control de OpenNebula, y redes virtuales VLAN para las VMs.

2.3.1 Dimensionamiento de la nube

La dimensión de una infraestructura de nube puede deducirse directamente de la carga de trabajo prevista en términos de VMs que se deben sostener. Esta carga de trabajo también es difícil de estimar, la comunidad de OpenNebula para construir una nube eficiente recomienda tener en cuenta algunos aspectos a la hora de dimensionarla como:

- **CPU:** la relación de compromiso prevista es que por cada núcleo de CPU que se desee usar en cada VM, debe existir uno físico. Estas CPU físicas pueden propagarse entre los diferentes físicos servidores.
- **Memoria:** La planificación de la memoria es sencilla, ya que no hay compromiso excesivo de la memoria en OpenNebula. Siempre es práctico contar para una sobrecarga de un 10% del hipervisor (esto es un límite superior absoluto, en dependencia del hipervisor esto se puede ajustar). Por lo tanto, con el fin de sostener una carga de trabajo de 45 VMs con 2 GB de RAM cada una, se necesita 90 GB de memoria física. El número de servidores físicos es importante, ya que cada uno va a incurrir en una sobrecarga de 10% debido a los hipervisores.
- **Almacenamiento:** Es importante entender cómo OpenNebula utiliza el almacenamiento, principalmente la diferencia entre el almacén de datos del sistema y de imagen. El almacén de datos de imagen es donde OpenNebula almacena todas las imágenes registradas que se pueden utilizar para crear VM, por lo que la regla de oro es dedicar suficiente espacio para todas las imágenes que se deseen guardar. El almacén de datos del sistema es donde las VMs que actualmente corren almacenen sus discos, y es más difícil de estimar correctamente ya que los discos volátiles entran en juego con contrapartida en el almacén de datos de imagen (los discos volátiles se crean en el hipervisor). Un enfoque válido es limitar el almacenamiento disponible para los usuarios mediante la definición de cuotas en el número máximo de VMs y también el almacenamiento máximo volátil que un usuario puede exigir, y asegurar suficiente

espacio de almacén de datos de imagen en el sistema y cumplir con el límite de cuotas que se fija. En cualquier caso, actualmente, OpenNebula permite a los administradores de la nube añadir más almacenes de datos de imágenes o de sistema, si es necesario.

2.3.2 *Frontend*

El servidor que mantiene la instalación de OpenNebula se llama *Frontend*. Esta máquina necesita conexión de red con cada anfitrión y, posiblemente, el acceso a los almacenes de datos (ya sea por conexión directa o red). La instalación de la base de OpenNebula requiere menos de 50 MB. Los servicios de OpenNebula incluyen:

- Demonio de administración (*oned*) y el planificador (*mm_sched*)
- Servidor de interfaz web (*sunstore-server*)

Hay varias plataformas que se certifican para actuar como *Frontend* para cada versión de OpenNebula, por ejemplo: RedHat Enterprise Linux, Ubuntu Server, Debian, CentOS, Ceph, KVM, VMware, entre otras. La base de datos predeterminada que se utiliza es **SQLite.** Si se planea un entorno de producción o despliegue a gran escala, se debe considerar el uso de **MySQL.**

El número máximo de servidores (nodos de virtualización) que se pueden gestionar por una sola instancia OpenNebula (zona) depende en gran medida del rendimiento y la escalabilidad de la infraestructura de la plataforma subyacente, principalmente el subsistema de almacenamiento. No se recomienda más de 500 servidores dentro de cada zona, pero hay usuarios con 1.000 servidores en cada zona [47].

2.3.3 Monitoreo

El subsistema de supervisión recopila información relativa a los nodos y las VMs, tales como el estado del anfitrión, los indicadores básicos de rendimiento, así como el estado de VMs y el consumo de la capacidad. Esta información se recoge mediante la ejecución de un conjunto de exploraciones estáticas ofrecidas por OpenNebula. La salida de estas exploraciones se envía de dos maneras diferentes:

- **Modelo UDP-*push*:** Cada anfitrión periódicamente envía los datos de control por medio del protocolo de datagrama de usuario (UDP) al *Frontend*, este lo recopila y

lo procesa en un módulo dedicado a estas cuestiones. Este modelo es altamente escalable y su límite (en términos de número de VMs por segundo) se limita al rendimiento del servidor que ejecuta *oned* y el servidor de base de datos.

- **Modelo *Pull*:** OpenNebula periódicamente consulta activamente cada anfitrión y ejecuta las encuestas a través de *SSH* Este modo se limita por el número de conexiones activas que se pueden hacer al mismo tiempo, los anfitriones *s*e consultan de forma secuencial.

2.3.4 Virtualización del anfitrión

Los anfitriones son las máquinas físicas donde se ejecutarán las VMs. Hay varias plataformas, las cuales se certifican para actuar como nodos para cada versión de OpenNebula. El Subsistema de Virtualización es el componente a cargo de la comunicación con el hipervisor que se instala en los nodos y toma las acciones necesarias por cada paso del ciclo de vida de la VM. OpenNebula soporta de forma nativa tres hipervisores: Xen, KVM y VMware. Además permite incorporarle LXC mediante un complemento desarrollado por terceros [48].

2.3.5 Almacenamiento

OpenNebula utiliza almacenes de datos (*datastore*) para manejar las imágenes de disco de máquina virtual. Las versiones anteriores se refieren a este concepto como repositorio de imágenes. Típicamente, un almacén de datos se respalda por los servidores SAN / NAS. En general, cada almacén de datos tiene que ser accesible a través del *Frontend* donde se utiliza cualquier tecnología: NAS, SAN o almacenamiento de conexión directa.

Cuando se implementa una VM, las imágenes se transfieren desde el almacén de datos a los nodos. En dependencia de la tecnología de almacenamiento que se maneja puede significar una transferencia real, un enlace simbólico o la creación de un volumen LVM.

OpenNebula se envía con tres clases de almacenes de datos diferentes:

- Almacenes de datos del sistema: para mantener las imágenes de VMs en ejecución, en función de la tecnología de almacenamiento que se utiliza, estas imágenes

temporales pueden ser copias completas de la imagen original, *qcow deltas* o enlaces simples del sistema de archivos.

- Almacenes de datos de imagen: se almacenan en el repositorio de imágenes de disco. Las imágenes de disco se mueven, o clonan desde el almacén de datos del sistema cuando las VMs se despliegan o apagan; o cuando los discos están unidos o instanciados.
- Almacén de datos de archivo: es un almacén de datos especial que se utiliza para guardar archivos planos y no imágenes de disco. Los archivos planos se pueden utilizar como *kernels*, *ramdisk*s o archivos de contexto.

Los almacenes de datos de imágenes pueden ser de diferentes tipos, a través de la tecnología de almacenamiento subyacente:

- Sistema de Archivo: para almacenar imágenes de disco en forma de archivos. Los archivos se almacenan en un directorio que se monta desde un servidor SAN / NAS.
- Sistema de archivos de Máquinas Virtuales (*Virtual Machine File System,* VMFS), un almacén de datos que se especializa en formato VMFS y se utiliza con hipervisores de VMware. No se puede montar en el *Frontend* de OpenNebula desde VMFS porque no es compatible.
- Administrador de volúmenes lógicos (LVM), almacén de datos que proporciona OpenNebula con la posibilidad de utilizar volúmenes LVM en lugar de archivos planos para mantener las imágenes virtuales. Esto reduce la sobrecarga de tener un sistema de archivos en su lugar y por lo tanto aumenta el rendimiento.
- *Ceph*, para almacenar imágenes de disco donde se utilizan dispositivos de bloque *Ceph*.

2.3.6 Red y autentificación

OpenNebula proporciona un subsistema de red fácilmente adaptable y personalizable con el fin de integrar mejor a los requisitos específicos de la red de centros de datos existentes. Se necesitan al menos dos redes físicas diferentes:

- Una **red de servicio** es necesaria para que los demonios de OpenNebula del *Frontend* puedan acceder a los anfitriones con el fin de gestionar y controlar los

hipervisores, y mover los archivos de imagen. Es recomendable instalar una red dedicada para este fin.

- Una **red cliente (*instance network*)** para ofrecer conectividad entre las VMs a través de los anfitriones. Para hacer un uso efectivo de las implementaciones de VMs probablemente se tienen que tomar una o más redes físicas accesibles para esto.

El administrador OpenNebula puede asociar a uno de los controladores siguientes a cada anfitrión:

- dummy: controlador predeterminado que no realiza ninguna operación de la red. También se tienen en cuenta las reglas de cortafuegos.
- *FW*: se aplican las reglas del cortafuego, pero el aislamiento de redes es ignorado.
- *802.1Q*: restringe el acceso a la red a través de etiquetado VLAN, que también requiere que los equipos de interconexión soporten VLAN.
- *ebtables*: restringe el acceso a la red a través de reglas *ebtables.* No se requiere una configuración especial de *hardware.*
- *ovswitch*: restringe el acceso a la red con el conmutador virtual *Open vSwitch* (OVS).
- *VMware*: utiliza la infraestructura de red de VMware para proporcionar una red aislada y compatible con 802.1Q para máquinas virtuales que emplean el hipervisor VMware.

Autenticación: Se puede elegir entre los siguientes modelos de autenticación de acceso OpenNebula: usuario / contraseña, ssh, X509 y LDAP.

2.4 Conclusiones

El auge de los hipervisores libres como (KVM o Xen) hacen que la integración con los sistemas GNU/Linux esté muy optimizada y los resultados sean de alta calidad, esto es un mercado en donde GNU/Linux está bien posicionado. Como emblemas de plataformas de gestión de la nube existen en la actualidad los siguientes productos: OpenStack, CloudStack, OpenNebula y Eucalyptus.

Estos *software* ofrecen la flexibilidad de la comunidad, por ejemplo: acceso a la comunidad, recursos compartidos, actualizaciones rápidas, libertad en la modificación y adaptación de los productos por necesidades de los clientes actuales, así como una completa interoperabilidad entre plataformas y una estandarización al acceso de los recursos.

En la UCLV se necesita la más flexible, que ofrezca facilidades de instalación y manejo, y que sea lo más abierta posible. Todas estas características las cumple OpenNebula, por lo que las pruebas en la entidad se van a llevar a cabo con la misma.

CAPÍTULO 3. IMPLEMENTACIÓN DE LA PLATAFORMA DE GESTIÓN OPENNEBULA

En el presente capítulo se mencionan los principales pasos a tener en cuenta en la instalación y despliegue de la plataforma OpenNebula. Así como las deficiencias encontradas en las guías que brinda *OpenNebula.org*, para resolver estos problemas se proponen sugerencias. Como resultado se realizaron guías para la instalación y despliegue de la plataforma de gestión de código abierto OpenNebula, una con la configuración del hipervisor LXC y otra con KVM ambas para el sistema operativo Debian, en esta última se describe cuáles son los pasos a seguir si se utiliza los controladores de red *Open vSwitch* o *dummy*.

3.1 Infraestructura empleada

Para la implementación de una IaaS con la plataforma de gestión de código abierto OpenNebula en la UCLV, se destinaron 5 servidores profesionales marca Dell con los siguientes requerimientos mostrados en la tabla 3.1.

Solamente los procesadores *Intel(R) Xeon(R) CPU E5310* soportan virtualización por *hardware* por tanto fueron seleccionados para instalar el hipervisor KVM, lo cual es requerido. Cada servidor posee dos interfaces de red lo que permite desplegar dos redes separadas como es recomendado en la documentación oficial: una para la gestión y otra para las máquinas virtuales. En la tabla 3.2 se muestra la distribución de las redes empleadas y en la tabla 3.3 se describe cómo están los servidores según la función a emplear.

Tabla 3.1 Servidores de OpenNebula.

Modelo	**CPU**	**RAM**	**HDD**	**NIC**
Dell 1955 (*Blade*)	Intel(R) Xeon(TM) CPU @ 3.00GHz (dos procesadores físicos)	1 GB	73 GB	*Intel Gigabit* (dos NICs)
Dell 1955 (*Blade*)	Intel(R) Xeon(TM) CPU @ 3.20GHz (dos procesadores físicos)	1 GB	73 GB	*Intel Gigabit* (dos NICs)
Dell 1955 (*Blade*)	Intel(R) Xeon(R) CPU E5310 @ 1.60GHz (dos procesadores físicos)	4 GB	73 GB	*Intel Gigabit* (dos NICs)

Tabla 3.2 Distribución de la red para el despliegue de OpenNebula.

	Red	**Puerta de Enlace**	**DNS**
Red de administración	10.12.112.0/24	10.12.112.254	10.12.1.50 10.12.1.51
Red de clientes	10.12.114.0/24	10.12.114.254	10.12.1.50 10.12.1.51

Tabla 3.3 Distribución de los servidores de OpenNebula según su función.

Nombre	***IP***	***SO***	***Hipervisor***
Frontend	10.12.112.140	Ubuntu 14.04 LTS	
Nebula1	10.12.112.141	Debian Jessie	LXC
Nebula2	10.12.112.142	Debian Jessie	LXC
Nebula3	10.12.112.145	Debian Jessie	KVM + *dummy*
Nebula4	10.12.112.146	Debian Jessie	KVM + *Open vSwitch*

3.2 Instalación de OpenNebula

Durante la instalación de los mismos se consultaron las guías que ofrece *OpenNebula.org*. En el momento que se redacta el presente informe la versión estable de OpenNebula era la 4.14, a lo largo del desarrollo de los pasos de instalación y configuración se encontraron algunas deficiencias en las guías oficiales, las cuales son comentadas a lo largo de este

capítulo como recomendación para futuras versiones de la plataforma, así como una restructuración del orden a seguir en los pasos con respecto a las guías oficiales.

A lo largo de la instalación de OpenNebula hay dos funciones separadas: interfaz de administración (*Frontend)* y Nodos**.** El servidor *Frontend* ejecuta los servicios de OpenNebula, mientras que los nodos ejecutan las máquinas virtuales en dependencia del hipervisor empleado.

En el caso particular LXC en Ubuntu 14.04 LTS se presentaron problemas a la hora de crear las plantillas LXC, después de investigar a fondo no se dio con una solución efectiva ya que más bien todo indicaba que es problema de la creación de plantillas en dicha versión de Ubuntu, por lo que se decidió montar todos los nodos en Debian Jessie. En cambio el *Frontend* no presenta problemas en la versión de Ubuntu indicada.

Para la realización de las guías de instalación y configuración de OpenNebula (Anexo III y V) que acompañan este informe se siguió la documentación oficial *OpenNebula.org* [49]. Debido que actualmente OpenNebula no incluye soporte para LXC como hipervisor por defecto, se procede a emplear un complemento de terceros desarrollado por Instituto Superior Politécnico José Antonio Echeverría (Cujae) llamado *Addon LXCoNe*, para su instalación y configuración se siguió la guía "*LXCoNe. Installation and Configuration Guide*" disponible en GitHub [48].

3.3 Instalación del *Frontend*

El proceso de instalación del *Frontend* tanto para los hipervisores KVM como LXC es similar y descrito en los Anexos III y V. Primeramente se configuraran los repositorios de OpenNebula según el sistema operativo empleado. Los repositorios de la UCLV poseen la instalación de OpenNebula 14.04 para Debian Jessie y Ubuntu 14.04 LTS, los cuales fueron seleccionados para desplegar la plataforma de gestión.

A continuación se muestran los paquetes disponibles para los sistemas operativos Debian/Ubuntu:

- **opennebula-common**: proporciona los archivos comunes a los usuarios.
- **ruby-opennebula**: API Ruby.

- **libopennebula-java**: API Java.
- **libopennebula-java-doc**: Documentación de la API de Java.
- **opennebula-node**: Prepara un nodo como un nodo OpenNebula.
- **opennebula-sunstone**: Interfaz *Web* OpenNebula Sunstone.
- **opennebula-tools**: Interfaz de líneas de comandos.
- **opennebula-gate**: Permite la comunicación entre VMs y OpenNebula.
- **opennebula-flow**: Administra los servicios y la elasticidad.
- **opennebula**: Demonio de OpenNebula.

El *Frontend* se instaló en Ubuntu 14.04 LTS y se comprobó que la interfaz *web* Sunstone no inicia automáticamente como un servicio con el arranque del sistema operativo, lo que trae consigo que en caso de reiniciar el servidor o fallas eléctricas, no se puede acceder a la administración vía *web* de OpenNebula. Para solucionar este inconveniente, se agregó la siguiente línea en el archivo *"/etc/rc.local"*:

```
service opennebula-sunstone start
```

Además fue necesario cambiar la interfaz por la que escucha Sunstone, ya que por razones de seguridad sólo escucha en la interfaz local (*loopback*). Para ello es necesario modificar el archivo "*/etc/one/sunstone-server.conf*" y ajustar la opción *host* al IP de la interfaz de administración del *Frontend* o especificar *0.0.0.0* para que escuche en todas las interfaces. Finalmente es necesario reiniciar el servicio para aplicar los cambios.

Para desplegar LXC en OpenNebula se deben descargar los controladores para este tipo de hipervisor, modificar el archivo de configuración del demonio de OpenNebula, agregar la información referente al controlador LXC, ajustar los permisos tanto de los nuevos directorios incorporados a OpenNebula, así como asignarle permisos de *root* al usuario *oneadmin* para que ejecute los comandos *lxc* sin contraseña.

3.2.1 Configuración de Sistema de Archivo de Red (*Network File System*, NFS)

Las imágenes son almacenadas en el directorio "*/var/lib/one/datastore/<DATASTORE_ID>"*. Igualmente, cada VM activa tiene su propio directorio que se guarda con el *VM_ID* de la misma, en el *Datastore* del Sistema correspondiente. Estos directorios contienen los discos de la VM y archivos adicionales como *snapshots* y *checkpoint*.

Por ejemplo, un sistema con el *Datastore* de Imagen (1) con tres imágenes de VMs (las VM 0 y 2 corriendo y la 7 apagada), ejecutado en un *Datastore* del Sistema (0), presenta la siguiente estructura:

```
/var/lib/one/datastores
|-- 0/
|   |-- 0/
|   |   |-- disk.0
|   |   `-- disk.1
|   |-- 2/
|   |   `-- disk.0
|   `-- 7/
|       |-- checkpoint
|       `-- disk.0
`-- 1
    |-- 05a38ae85311b9dbb4eb15a2010f11ce
    |-- 2bbec245b382fd833be35b0b0683ed09
    `-- d0e0df1fb8cfa88311ea54dfbcfc4b0c
```

El controlador de transferencia compartido, asume que el *datastore* sea montado en todos nodos del clúster, lo cual típicamente se realiza con un sistema de archivos distribuidos como NFS. Cuando se crea una VM, los discos se copian o se enlazan al directorio correspondiente al *Datastore* del Sistema. El modo de transferencia compartido y el de transferencia *qcow2* usualmente reducen el tiempo de desarrollo de las VM y permite hacer migraciones en caliente de las mismas, lo que puede generar cuellos de botellas en la infraestructura y degradar su rendimiento. Para esto se recomienda:

- Usar diferentes servidores NFS para almacenar las imágenes, así balancear el ancho de banda.
- Emplear un *Datastore* del Sistema ssh, donde las imágenes se copian localmente en cada nodo.
- Ajustar las configuraciones de los servidores NFS a los requerimientos.

En caso de emplear varios *datastores* en servidores NFS distintos se deben compartir cada uno por separado hacia los nodos e incluirlos en el *Frontend*, por ejemplo compartir *"/var/lib/one/datastore/<datastore_id>"*. Si no es el caso con solo compartir el directorio *"/var/lib/one"* entre todos los nodos basta, como se hace en el Anexo III y V. Para los hipervisores LXC es necesario además compartir entre todos los nodos el directorio *"/var/log/one"*.

Por otra parte está tener el *Datastore* del Sistema distribuido mediante ssh en los nodos. El controlador de transferencia ssh utiliza a almacén local del nodo para guardar las imágenes de VM encendidas, y todas las operaciones son locales, esto tiene como inconveniente que consume recursos y no permite el uso de migraciones en caliente.

3.3 Instalación de los nodos

En los nodos luego de configurar los repositorios, se instalan los paquetes necesarios en dependencia del tipo de hipervisor.

Para hipervisores LXC requieren los paquetes:

```
opennebula-node nfs-common bridge-utils xmlstarlet libpam-runtime bc at
libvncserver0 libjpeg62 lxc
```

Para KVM con el controlador de red *dummy* solo necesita:

```
opennebula-node nfs-common bridge-utils
```

Es importante señalar que en caso de utilizar el controlador de red OVS, es recomendable no instalar el paquete *bridge-utils*, pues los puentes de Linux y OVS no funcionan al mismo tiempo.

Es recomendable emplear una segunda interfaz red, normalmente *eth1*, conectada a un puente (*bridge*). El nombre del puente debe ser el mismo en todos los nodos, en este caso *br0*. Separar la red de monitorización y la de VMs es más ventajoso por cuestiones de seguridad. Por otra parte, si se emplea OVS la interfaz que se asocia con el puente pierde la conectividad, por lo que en este caso siempre hay que asociar el puente OVS con una segunda interfaz o una subinterfaz al menos.

La configuración de red según lo mencionado anteriormente queda de la siguiente forma:

```
 auto lo
```

```
iface lo inet loopback

auto eth0
iface eth0 inet static
       address 10.12.112.141
       network 10.12.112.0
       netmask 255.255.255.0
       broadcast 10.12.112.255
       gateway 10.12.112.254

       dns-nameservers 10.12.1.50 10.12.1.51
       dns-search uclv.edu.cu uclv.cu

auto br0
iface br0 inet static
       address 10.12.114.1
       network 10.12.114.0
       netmask 255.255.255.0
       broadcast 10.12.114.255
       gateway 10.12.114.254
       bridge_ports eth1
       bridge_fd 9
       bridge_hello 2
       bridge_maxage 12
       bridge_stp off
```

En la configuración del servicio NFS en ocasiones es posible que no se monten correctamente los directorios al reiniciar el servidor por lo que se recomienda agregar las siguientes líneas de comando en el archivo "/*etc/rc.local"*.

Para nodos LXC:

```
mount /var/lib/one/
mount /var/log/one
```

Para nodos KVM:

```
mount /var/lib/one/
```

3.3.1 Asignar los privilegios necesario al usuario *oneadmin*

Cada una de las operaciones que realiza OpenNebula son bajo el usuario *oneadmin* por lo que se añade el usuario *oneadmin* el archivo *sudoers*, y se habilita para que ejecute comandos desde *root* sin contraseña. Para ello se ejecuta el comando *visudo* y se agrega, justamente debajo de línea: "*root ALL=(ALL:ALL) AL*", lo siguiente:

```
oneadmin ALL= NOPASSWD: ALL
```

Aunque OpenNebula despliega el archivo *"/etc/sudoers.d/opennebula"* con los permisos necesarios para el usuario *oneadmin*, se recomienda además realizar lo comentado anteriormente.

En los nodos con hipervisores LXC, es necesario ajustar los permisos del directorio del contenedor para que el usuario *oneadmin* ejecute los *scripts*:

```
# chmod +rx /var/lib/lxc
```

3.4 Uso básico de los nodos con KVM

3.4.1 Agregar nodo

Antes de unir los nodos se comprueba que el archivo *"/etc/hosts*" contenga el nombre del *Frontend* y de los nodos (esto es necesario hacerlo para el *Frontend y* cada nodo que se va a agregar), por ejemplo:

```
10.12.112.140       frontend.uclv.edu.cu      frontend
10.12.112.141       nebula1.uclv.edu.cu       nebula1
10.12.112.142       nebula2.uclv.edu.cu       nebula2
10.12.112.145       nebula3.uclv.edu.cu       nebula3
10.12.112.146       nebula4.uclv.edu.cu       nebula4
```

Los nodos se agregan mediante el comando *onehost create*, al mismo se le especifica el nombre del nodo, por ejemplo nebula1, y las opciones descritas a continuación:

- --*im/-i*: Información del controlador. Opciones válidas*: kvm, xen, vmware, ec2, ganglia, dummy*.
- --*vm/-v*: Controlador de VMs. Opciones válidas: *kvm, xen, vmware, ec2, dummy*.
- --*net/-n*: *C*ontrolador de red. Opciones válidas: *802.1Q, dummy, ebtables, fw, ovswitch, vmware*.

Una vez que se crea el anfitrión se comprueba que esté activo con el comando *onehost list*. Si falla es probable que se tenga algún problema en la configuración de ssh, por lo que se puede verificar el archivo "*/var/log/one/oned.log"*.

3.4.2 Agregar recursos virtuales

En OpenNebula las VMs son definidas a través de una plantilla, la cual puede ser creada desde la interfaz *web* o desde consola como se muestra a continuación:

```
$ onetemplate create --name "CentOS-7.1" --cpu 1 --vcpu 1 --memory 256 \
--arch x86_64 --disk "CentOS-7.1_x86_64" --nic "red114KVM" --vnc --ssh \
--net_context
```

Es importante aclarar el nombre de la red y de la imagen creada debe de coincidir con el nombre que se le pase a la opción *"--nic"* y *"--disk"* respectivamente del comando *onetemplate*. La opción *"--net_context"* permite activar la red de las imágenes contextualizadas del *Marketplace*, esta opción es obligada para poder activar la red en las VMs. Existe otra muy útil que permite solamente generar la plantilla de configuración especificada, para ello agregar la opción *"--dry"*.

Es válido aclarar en la plantilla el tipo de hipervisor empleado en el nodo, así se evitan fallos, por ejemplo, si se hace una VM con KVM y el planificador la envía a correr a un nodo de LXC, esta puede dar fallida. Al añadir un nodo se indica el tipo de hipervisor, esto permite a OpenNebula conocer qué tipo de hipervisores emplea cada nodo, por lo que se puede tener un entorno heterogéneo sin problemas, por ejemplo tener un número de hipervisores LXC y otros sólo KVM (no se pueden mezclar en el mismo nodo). Después al instanciar o arrancar la máquina virtual OpenNebula emplea un planificador (*scheduler*) como en un sistema de colas (usa por defecto un algoritmo *round robin*) para ejecutar la VM en el nodo con menos carga en ese momento.

Cuando se emplea un entorno heterogéneo de hipervisores (KVM y LXC), desde la interfaz *web* en las plantillas de KVM, en la sección *Scheduling -> Placement* especificar:

```
HYPERVISOR=\"kvm\"
```

Para las plantillas con hipervisores LXC:

```
HYPERVISOR=\"lxc\"
```

Desde la CLI es posible crear la plantilla con todos los parámetros requeridos para este caso:

```
$ onetemplate create --name "CentOS-7.1" --cpu 1 --vcpu 1 --memory 256 \
--arch x86_64 --disk "CentOS-7.1_x86_64" --nic "red114KVM" --vnc --ssh \
--net_context --dry > CentOS-7.1.tmpl
```

Luego editar el archivo *"CentOS-7.1.tmp"*, agregar los requerimientos del planificador y el tipo de hipervisor como se muestra:

```
NAME="CentOS-7.1-test"
OS = [
  ARCH = "x86_64" ]
CPU=1.0
VCPU=1
MEMORY=128
DISK=[
  IMAGE="CentOS-7.1_x86_64"
]
NIC=[
  NETWORK="red114KVM"
]
GRAPHICS=[ TYPE="vnc", LISTEN="0.0.0.0" ]
CONTEXT=[
  SSH_PUBLIC_KEY="$USER[SSH_PUBLIC_KEY]",
  NETWORK = "YES"
]
## Scheduler requeriments
SCHED_REQUIREMENTS = "HYPERVISOR=\"kvm\""
## Hypervisor type
HYPERVISOR="kvm"
```

Finalmente crear la plantilla a partir del archivo "*CentOS-7.1.tmpl"*:

```
$ onetemplate create CentOS-7.1.tmpl
```

Más información sobre las opciones disponibles para las plantillas se encuentra disponible en la documentación oficial de OpenNebula [50].

Por motivos de seguridad las imágenes de máquinas virtuales que se descargan del *Marketplace* de OpenNebula no poseen contraseña de *root*, solo es posible acceder con claves ssh. Se sigue esta política para evitar que un usuario pueda acceder a la VM de otro, si este no ha cambiado la clave. Para ello estas imágenes traen instalado el paquete de red *one-context*, que se encarga de habilitar la red y modificar la clave ssh del usuario *root* en el arranque.

Para ello tiene que estar definida la variable *SSH_PUBLIC_KEY* del usuario que arranque la máquina (por ejemplo *oneadmin*) o incluir esta variable dentro de la sección *CONTEXT* de la plantilla de la máquina virtual, por lo que hay que añadir la llave ssh a la plantilla de usuario, primero es necesario copiar la salida del siguiente comando en el portapapeles:

```
$ cat ~/.ssh/id_rsa.pub
```

Luego editar la plantilla del usuario *oneadmin* y agregar la variable *SSH_PUBLIC_KEY* con el contenido del comando anterior:

```
$ EDITOR=vi oneuser update oneadmin
SSH_PUBLIC_KEY=¨ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAA……¨
```

Para entornos de pruebas se puede emplear la imagen *ttylinux-kvm* disponible en el *Marketplace* de OpenNebula, esta puede ser accedida por usuario *root* y contraseña *password*. Debido a que la configuración de la imagen es muy simple no se especifica la máscara de red y la puerta de enlace.

3.5 Nodo KVM con el controlador *Open vSwitch*

Open vSwitch proporciona dos funcionalidades independientes que pueden ser útiles, pues permite el aislamiento de la red con el uso de VLAN y filtrado de red a través de *OpenFlow*. Cada interfaz de red virtual recibe una etiqueta que permite el aislamiento por VLAN. El ID de la VLAN es el mismo para cada interfaz en la red. Este se asigna de forma dinámica o puede ser especificado mediante el parámetro VLAN_ID en la plantilla de red virtual. La función del filtro de red es muy similar a los controladores del servidor de seguridad, con algunas limitaciones. Se requiere instalar OVS en cada uno de los nodos que lo emplee.

Es recomendable, en caso de tener varias VLAN, no añadir un *bridge* para cada una de ellas, con solo crear uno basta pues el resto lo gestiona OVS (ver Anexo IV). Este controlador es aconsejable sobre todo si se quiere añadir usuarios externos a la nube ya que mejora la seguridad.

Como se menciona anteriormente, no funcionan al mismo tiempo el *bridge* de Linux y OVS, lo más seguro es eliminar el paquete "*bridge-utils*" (si se encuentra instalado) si se va a emplear OVS en un nodo. Independientemente se pueden tener mezcla de hipervisores unos con *bridge* de Linux y otros con OVS en OpenNebula siempre que se especifique el nombre del *bridge* a usar con las variables en la configuración de la red virtual (VNET):

```
BRIDGE="nombre de tu bridge linux"
BRIDGE_OVS="nombre de tu bridge OVS"
```

3.6 Modificación de imágenes

Tanto las imágenes para KVM o LXC es necesario ajustarlas a los requerimientos de la institución o de los usuarios que las emplearán.

3.6.1 KVM

Actualmente OpenNebula no ofrece un procedimiento automatizado para realizar una copia de una imagen y plantilla, por lo que el procedimiento más sencillo para personalizar una imagen KVM es:

- Descargar la imagen con el SO base del Marketplace ya que incluye el *script* de contextualización para la clave ssh y la red.
- Agregar la imagen descargada a OpenNebula y convertirla en persistente con el comando:

```
$ oneimage persistent <id>
```

- Arrancar la imagen y hacer los cambios necesarios.
- Después de finalizados los cambios apagar (*shutdown*) la máquina desde OpenNebula para que el *datastore* se actualice.
- Convertir la imagen en no persistente para que se puedan arrancar varias instancias de la misma:

```
$ oneimage nonpersistent <id>
```

De forma opcional se pueden ajustar los permisos de la nueva imagen creada y de la plantilla para hacerlas públicas, y que sea accesible por otros usuarios.

En la nueva versión estos pasos ya no son necesarios ya que tiene una opción de clonar o arrancar una copia directamente.

3.6.2 LXC

Las imágenes LXC son generadas a través del comando *"lxc-create"*, este se encarga de generar un sistema de archivos mínimo en dependencia del sistema operativo especificado. Una vez creada la imagen se debe de asociar a un *bridge* y ajustar su configuración de red, como se muestra a continuación en el archivo "*/var/lib/lxc/<name>/config"*:

```
# ETH0
lxc.network.type = veth
lxc.network.flags = up
```

```
lxc.network.name = eth0
lxc.network.link = br0
lxc.network.ipv4 = 10.12.114.9/24
lxc.network.ipv4.gateway = 10.12.112.254
```

Luego se procede a iniciar el contenedor y acceder al mismo. El acceso puede ser realizado mediante los comandos *"lxc-console"* o *"lxc-attach"*; este último tiene como ventaja que no se necesita especificar la contraseña del usuario *root* de la imagen.

Una vez dentro es necesario configurar los repositorios como se muestra en el Anexo V ya que la imagen generada no posee casi paquetes instalados. Para instalar solamente los paquetes que trae Debian por defecto, basta con ejecutar los siguientes comandos:

```
# apt-get install tasksel
# tasksel install standard
# apt-get install mc openssh-server
```

Finalmente apagar el contenedor y seguir los pasos mostrados en el Anexo V para subir la imagen a OpenNebula, crear la plantilla y desplegar los contenedores.

3.7 Operaciones útiles

3.7.1 Migración en caliente de máquinas virtuales

Una vez que se crea un nodo OpenNebula le asigna un ID a cada uno para su identificación. Lo mismo sucede cuando se crea una VM, este ID se usa para monitorear y controlar las VMs.

Cuando se desea migrar en caliente de un nodo a otro se utiliza el comando *onevm migrate*. Por ejemplo se tiene un nodo (con HID = 1) y una VM (con VID = 0), la línea queda así:

```
$ onevm migrate --live 0 1
```

Desde la interfaz web Sunstone es posible realizar la operación anterior desde las opciones disponibles al seleccionar una máquina virtual o contendor.

3.7.2 Respaldo de máquinas virtuales

Las VMs en OpenNebula pueden ser respaldadas en diferentes maneras:

- Mediante *saveas*. Consiste en realizar copias de los discos de las VMs y son almacenadas en el *Datastore* de imágenes. El volumen debe ser apagado para lograr

consistencia. No se necesita apagar la VM si se utiliza el comando "*onevm disk-saveas --live"*. El proceso puede ser automatizado a través de un simple *script* o a través de una tarea programada. La VM se recupera manualmente desde OpenNebula (mediante la definición de una VM guardada en un disco).

- Mediante el sistema de almacenamiento subyacente. Consiste en crear un *Datastore* del Sistema con sus propias políticas de respaldo. El proceso de respaldo es transparente a OpenNebula. Este puede tener sus desventajas como: 1) es posible que se requiera algún tipo de sincronización desde la VM para evitar que el sistema de archivos se corrompa y 2) el proceso de recuperación no puede ser inicializado desde OpenNebula.
- Mediante Ceph. Se recomienda emplear este sistema de almacenamiento ya que al estar replicado garantiza un nivel de seguridad alto y no es necesario realizar salvas de las VMs en sí.

Se implementó el primer método dado que es más fácil de desarrollar y no se cuenta con un *datastore* de Ceph por el momento. Para esto se desarrolla el siguiente *script*:

```
#/bin/bash

machine=$1
backup=${machine}_Backup
diskid=$(onevm show | sed -n -e '/VM DISKS/,/VM NICS/ p' | awk '/OS/ {print $1}')

   if [ $(/usr/bin/onevm show ${machine} | grep LCM_STATE | cut -d ":" -f 2) == "RUNNING" ]; then

     #Invoke backup tool by opennebula
     /usr/bin/onevm disk-saveas --live ${machine} ${diskid} ${backup} >> /dev/null

     #Wait for the backup to finish
     until [  $(/usr/bin/oneimage show ${backup} | grep STATE | cut -d ":" -f 2) == "rdy" ]; do
       echo "Not Ready" >> /dev/null
     done

     #OUtput the path for snapshot
     echo $(/usr/bin/oneimage show ${backup} | grep SOURCE | cut -d ":" -f 2)
   fi
```

Luego el *script* puede ser ejecutado de forma manual o mediante una tarea automatizada, solo hay que pasarle como parámetro el ID de la VM que se quiere salvar.

3.7.3 Inicio automático de las máquinas virtuales

En el caso de apagar los servidores por fallas eléctricas o cualquier otro motivo, las máquinas virtuales no se inician con el arranque del sistema operativo, una vía para

solucionar este inconveniente cómoda y sin modificar nada en los nodos, es usar un *hooks*. Cuando el hipervisor se reinicia, todas la VMs que se ejecutan en ese nodo pasan a un estado desconocido (*UNKNOWN*), si se realiza un "*resume"*, OpenNebula reinicia todas la VMs en ese estado.

Un *hook* es un *script* de *prolog/epilog* en OpenNebula que se encarga de hacer una acción en concreto cuando una VM está en un estado determinado, simplemente se añade en la sección *"Hook Manager Configuration"* del demonio de OpenNebula (*/etc/one/oned.conf*) el nuevo *hook*:

```
## This hook start VMs after a power failure or unexpected reboot
VM_HOOK = [
   name      = "hook_vm_on_unknown",
   on        = "UNKNOWN",
   command   = "ft/hook_vm_on_unknown.sh",
   arguments = "$ID $PREV_STATE $PREV_LCM_STATE" ]
```

Se reinicia el servicio para cargar la nueva configuración:

```
# /etc/init.d/opennebula restart
```

Luego se crea el *script "hook_vm_on_unknown.sh"* en *"/var/lib/one/remotes/hooks/ft/"* con el siguiente contenido:

```
#!/bin/bash
if [ $2 = "ACTIVE" ] && [ $3 = "RUNNING" ]; then
     onevm resume $1
fi
```

Finalmente se le asigna permisos de ejecución al *script* anterior:

```
# chmod +x /var/lib/one/remotes/hooks/ft/hook_vm_on_unknown.sh
```

3.7.4 Creación de un *Marketplace* local

OpenNebula ofrece un Marketplace el cual contiene la información de las imágenes de VMs, el mismo solo está disponible si el servidor *Frontend* posee conexión a Internet. AppMarket es un complemento que puede ser adicionado al Marketplace que permite crear un servicio interno para almacenar imágenes, mediante el cual los usuarios de la nube pueden acceder a las imágenes que crearon, así como distribuirlas entre varios clústeres e instituciones. Este servicio se agrega como una pestaña en el Sunstone con el nombre AppMarket, lo que reduce el tiempo necesario para desplegar OpenNebula [51].

El servicio AppMarket ofrece las siguientes características adicionales:

- Compartir los accesorios virtuales mediante instancias de OpenNebula.
- Distribuir internamente los accesorios virtuales desarrollados.
- Proveer accesorios desde un catálogo o portal.
- Integración con OpenNebula.
- Los accesorios pueden ser definidos como múltiples archivos y en una plantilla de OpenNebula.
- Los accesorios pueden ser agrupados en el catálogo.

El mismo requiere un servidor de base de datos MongoDB así como *ruby* y *rubygems* instalados. La guía de instalación y configuración de encuentra en la página del desarrollador [52]. La próxima versión de OpenNebula, que actualmente está en desarrollo, incorpora por defecto esta utilidad.

3.8 Resultados

Se implementó una IaaS en el centro de datos de la UCLV en un entorno de prueba, el mismo es gestionado a través de la plataforma de código abierto OpenNebula. Este está compuesta por cuatro nodos como se distribuyó en el epígrafe 3.1. La figura 3.8.1 muestra el estado actual de la nube OpenNebula de prueba montada en la UCLV.

Se configuraron varias redes de virtualización (Figura 3.8.2) con el objetivo de desplegar las VMs según los requerimientos que necesiten, ya que no todas las redes tienen los mismos privilegios y direccionamiento.

Los nodos configurados emplean el controlador de red *dummy*, debido a que es el controlador más sencillo de desplegar y no requiere de configuraciones adicionales. No obstante, se configuró un nodo con *Open vSwitch* dado a las ventajas que posee en cuanto a seguridad.

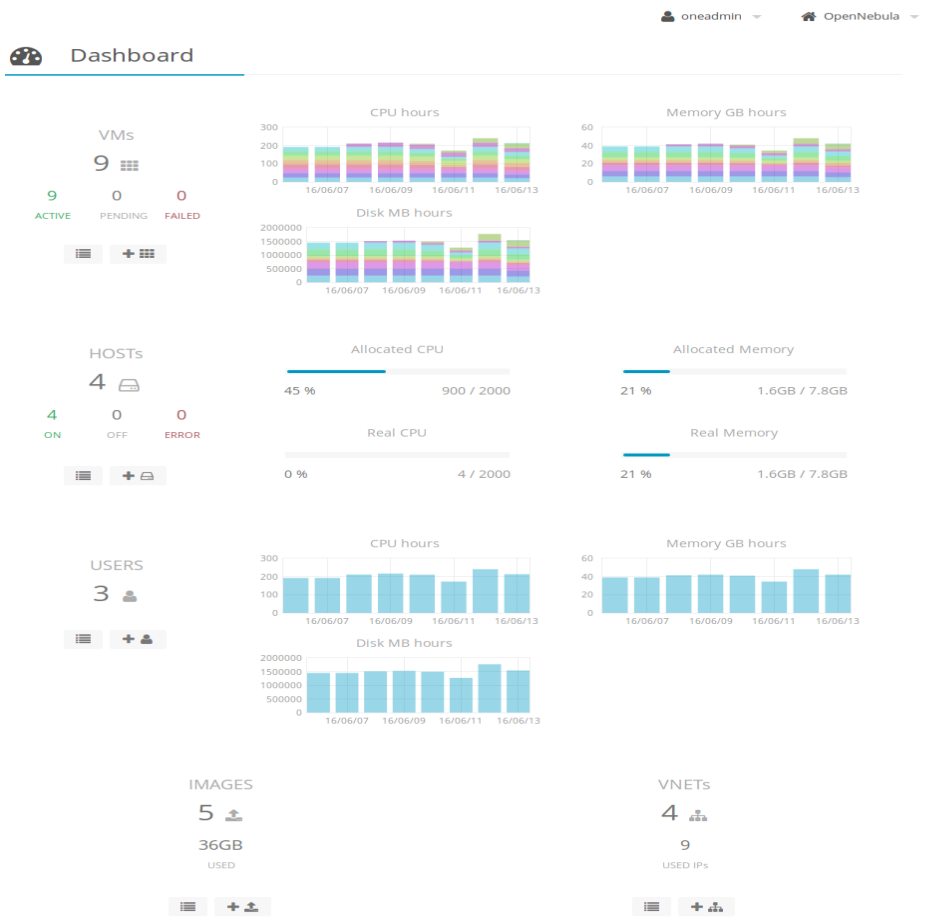

Figura 3.8.1 Interfaz de inicio OpenNebula.

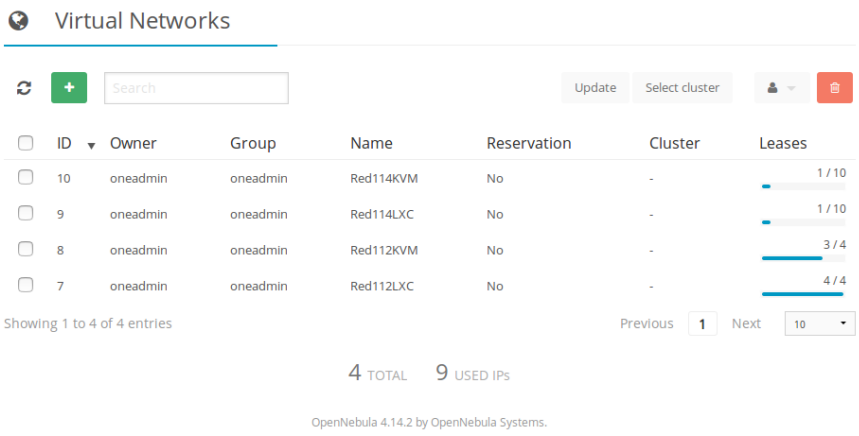

Figura 3.8.2 Configuraciones de red.

Se crearon varias imágenes y plantillas para desplegar las máquinas virtuales y contenedores. Las imágenes se personalizaron según los requerimientos de los usuarios y se le agregaron los repositorios locales de las distribuciones de Linux empleadas. Finalmente

fueron agregadas al *Marketplace* creado en la instalación de OpenNebula, la ventaja de emplear un *Marketplace* local radica fundamentalmente que las imágenes pueden distribuirse con sus plantillas entre varios clústeres o entre instituciones.

Se constató que los contenedores LXC demandan menos recursos que las VM de KVM, como era de esperar, ya que el primero se basa en la paravirtualización y el segundo la virtualización completa y esta requiere que los nodos soporten virtualización por *hardware.* En las siguientes figuras se observa el uso de *hardware* tanto por los hipervisores que en los *Datastore*.

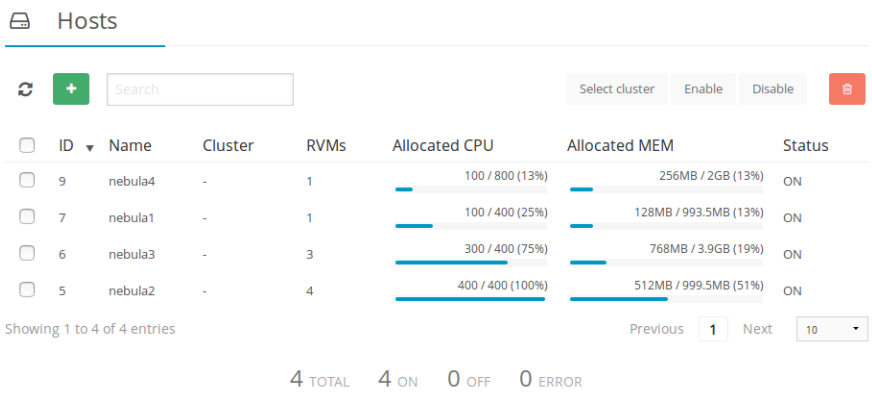

Figura 3.8.3 Uso de *hardware* de los distintos nodos.

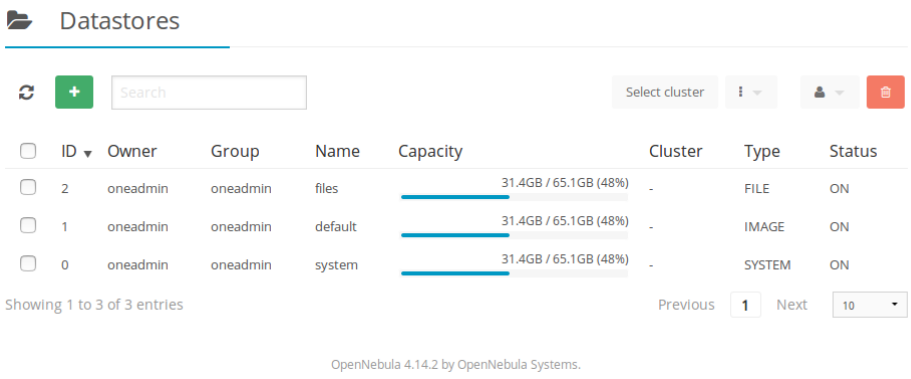

Figura 3.8.4 Uso de *hardware* para los distintos *Datastore*.

Como resultado se realizaron guías para la instalación y despliegue de la plataforma de gestión de código abierto OpenNebula, una con la configuración del hipervisor LXC y otra con KVM ambas para el sistema operativo Debian, en esta última se describe cuáles son los pasos a seguir si se utiliza los controladores de red *Open vSwitch* o *dummy*.

Durante las prueba se detecta como deficiencia en la plataforma que cuando se crea una VM y actualiza su plantilla, luego esta VM no asimila los cambios, pues para que tome los valores nuevos de la plantilla hay que borrar por completo la VM en ejecución, un *poweroff* o *shutdown* no son suficientes. Esto se debe a que la plantilla no se actualiza en el hipervisor (básicamente es el archivo XML que utiliza *libvirtd* para arrancar la máquina). La nueva versión OpenNebula que actualmente está en fase beta es capaz de detectar el cambio y actualizar el *cdrom* y la máquina virtual sin tener que borrarla.

3.9 Conclusiones

El despliegue y configuración de OpenNebula como servicio de IaaS requiere un dominio de la documentación de la plataforma así como de sistemas operativos GNU/Linux y redes de computadoras. En dependencia de los requerimientos de los usuarios se le asignan privilegios, recursos y redes. La plataforma permite gestionar eficientemente los sistemas de virtualización y alto desempeño.

Las deficiencias que se perciben durante el desarrollo de la investigación ya están puestas a mejorarse en la próxima versión de la plataforma.

Algunas opciones que en Proxmox VE se realizan de forma sencilla mediante la interfaz web, en OpenNebula requieren de configuraciones adicionales. Con vistas de asegurar la seguridad de la red e incorporar funciones de red adicionales se recomienda utilizar como controlar de red *Open vSwitch*.

CONCLUSIONES

Esta investigación permite evaluar la implementación de una Infraestructura como Servicio mediante la plataforma de gestión de código abierto OpenNebula en la UCLV.

1. La caracterización de los modelos de implementación y de servicio de la tecnología *Cloud Computing*, permitió implementar una Infraestructura como Servicio por los beneficios que esta ofrece en virtualización de centros de datos y equipos de alto desempeño.
2. Según las plataformas de gestión de nube analizadas, OpenNebula resultó la más eficiente al ser completamente abierta, lo que permite una interoperabilidad con componentes de la infraestructura actual en la UCLV, desde los sistemas operativos hasta hipervisores.
3. Las prácticas desarrolladas en la UCLV demostraron la viabilidad de OpenNebula como solución perfectamente aplicable al entorno existente, pues permite desplegar servicios de IaaS sobre infraestructura privada de forma eficiente, en términos de costos, mantenimiento, consumo eléctrico, escalabilidad, funcionalidad, y recursos humanos.
4. Las guías elaboradas para la instalación y despliegue de la plataforma de gestión de código abierto OpenNebula, para los hipervisores LXC y KVM sirven de material de consulta para el personal de soporte técnico y usuarios de la nube.

RECOMENDACIONES

Se recomienda la implementación de una IaaS en la UCLV, basada en la plataforma de gestión de código abierto OpenNebula.

REFERENCIAS BIBLIOGRÁFICAS

[1] D. C. Wyld, *Moving to the cloud: An introduction to cloud computing in government*: IBM Center for the Business of Government, 2009.

[2] J. Zhu, X. Fang, Z. Guo, M. H. Niu, F. Cao, S. Yue, *et al.*, "IBM cloud computing powering a smarter planet," in *Cloud Computing*, ed: Springer, 2009, pp. 621-625.

[3] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, *et al.*, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, pp. 137-146, 2010.

[4] L. J. Aguilar, "La Computación en Nube (Cloud Computing): El nuevo paradigma tecnológico para empresas y organizaciones en la Sociedad del Conocimiento," *Revista Icade. Revista de las Facultades de Derecho y Ciencias Económicas y Empresariales*, pp. 95-111, 2012.

[5] S. Falla, "Cloud Computing: nueva era de desarrollo'," *Maestros del Web*, 2008.

[6] C. Pettey, "Gartner estimates ICT industry accounts for 2 percent of global CO2 emissions," *Disponible en: https://www. gartner. com/newsroom/id/503867*, vol. 14, p. 2013, 2007.

[7] J. Carolan, S. Gaede, J. Baty, G. Brunette, A. Licht, J. Remmell, *et al.*, "Introduction to cloud computing architecture," *White Paper, 1st edn. Sun Micro Systems Inc*, 2009.

[8] A. Weiss, "Computing in the clouds," *Computing*, vol. 16, 2007.

[9] L. J. Aguilar, "COMPUTACIÓN EN LA NUBE: Notas para una estrategia española en cloud computing," *Revista del Instituto Español de Estudios Estratégicos*, vol. 1, 2013.

[10] J. Lastras Hernansanz, J. Lázaro Requero, and J. D. Mirón García, "Arquitecturas de red para servicios en Cloud computing," 2009.

[11] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Communications of the ACM*, vol. 17, pp. 412-421, 1974.

[12] L. Nussbaum, F. Anhalt, O. Mornard, and J.-P. Gelas, "Linux-based virtualization for HPC clusters," in *Montreal Linux Symposium*, 2009.

[13] Y. Xing and Y. Zhan, "Virtualization and cloud computing," in *Future Wireless Networks and Information Systems*, ed: Springer, 2012, pp. 305-312.

[14] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, 2012, pp. 877-880.

[15] J.-Z. Luo, J.-H. Jin, A.-B. Song, and F. Dong, "Cloud computing: architecture and key technologies," *Journal of China Institute of Communications*, vol. 32, pp. 3-21, 2011.

[16] T. V. Anthony, J. V. Toby, and E. Robert, "Cloud computing: A practical approach," *Anthony Velte, Toby Velte, Robert Elsenpeter.–2010*, 2010.

[17] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Comparison of multiple iaas cloud platform solutions," in *Proceedings of the 7th WSEAS International Conference on Computer Engineering and Applications,(Milan-CEA 13). ISBN*, 2012, pp. 978-1.

[18] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, pp. 1587-1611, 2013.

[19] A. Nakao, "Network virtualization as foundation for enabling new network architectures and applications," *IEICE transactions on communications*, vol. 93, pp. 454-457, 2010.

[20] J. Szefer and R. B. Lee, "Architectural support for hypervisor-secure virtualization," in *ACM SIGPLAN Notices*, 2012, pp. 437-450.

[21] D. S. Carrasco Aguilar and J. J. Bonilla Suárez, "Análisis e implementación de un prototipo de servidor virtualizado sobre una distribución de linux para el uso en PyMES," 2010.

[22] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," in *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, 2010, pp. 222-226.

[23] B. McCorkendale and P. Ferrie, "Using a hypervisor to provide computer security," ed: Google Patents, 2011.

[24] X. project. (febrero del 2016). *Xen Features*. Disponible en: *http://www.xenproject.org/*

[25] I. Habib, "Virtualization with kvm," *Linux Journal*, vol. 2008, p. 8, 2008.

[26] KVM. (febrero del 2016). *Kernel Virtual Machine*. Disponible en: *http://www.linux-kvm.org/page/Main_Page*

[27] B. Ward, *The book of VMware: the complete guide to VMware workstation* vol. 1: No Starch Press San Francisco, 2002.

[28] D. Sarddar and R. Bose, "Architecture of Server Virtualization Technique Based on VMware ESXI server in the Private Cloud for an Organization," *International Journal of Innovation and Scientific Research, ISSN*, pp. 2351-8014, 2014.

[29] VMware. (2015). *VMware Features*. Disponible en: *http://www.vmware.com/*

[30] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, "Vmware distributed resource management: Design, implementation, and lessons learned," *VMware Technical Journal*, vol. 1, pp. 45-64, 2012.

[31] R. Sudo, "LXC (linux containers) linux dentro de linux," ed, 2014.

[32] S. L. E. S. SP4. (2015, febrero del 2016). *Virtualization with Linux Containers (LXC)*. Disponible en: https://www.suse.com/documentation/sles11/singlehtml/lxc_quickstart/lxc_quickstart

[33] R. Rosen, "Linux containers and the future cloud," *Linux J*, vol. 240, 2014.

[34] U. Documentation. (febrero del 2016). *LXC*. Disponible en: https://help.ubuntu.com/lts/serverguide/lxc

[35] S. Roy, S. Nag, I. K. Maitra, and S. K. Bandyopadhyay, "International Journal of Advanced Research in Computer Science and Software Engineering," *International Journal*, vol. 3, 2013.

[36] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, *et al.*, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, 2009, pp. 124-131.

[37] S. Yadav, "Comparative study on open source software for cloud computing platform: Eucalyptus, openstack and opennebula," *International Journal Of Engineering And Science*, vol. 3, pp. 51-54, 2013.

[38] R. Kumar, K. Jain, H. Maharwal, N. Jain, and A. Dadhich, "Apache cloudstack: Open source infrastructure as a service cloud computing platform," *Proceedings of the International Journal of advancement in Engineering technology, Management and Applied Science*, pp. 111-116, 2014.

[39] N. Sabharwal, *Apache cloudstack cloud computing*: Packt Publishing Ltd, 2013.

[40] I. M. Llorente, "Eucalyptus, cloudstack, openstack and opennebula: A tale of two cloud models, 2013," Disponible en*:* http://blog. opennebula. org.

[41] D. Dongre, G. Sharma, M. Kurhekar, U. Deshpande, R. Keskar, and M. Radke, "Scalable cloud deployment on commodity hardware using OpenStack," in *Advanced Computing, Networking and Informatics-Volume 2*, ed: Springer, 2014, pp. 415-424.

[42] Openstack. (2015). *Considering OpenStack? Learn about the Path to Cloud*. Disponible en: https://www.openstack.org/

[43] D. Milojičić, I. M. Llorente, and R. S. Montero, "OpenNebula: A cloud management tool," *IEEE Internet Computing*, pp. 11-14, 2011.

[44] OpenNebula.org. *About the OpenNebula Project*. Disponible en: http://www.opennebula.org/about/project/

[45] OpenNebula.org. (marzo del 2016). *Why OpenNebula?* Disponible en: http://www.opennebula.org/about/why/#toggle-id-1

[46] R. Kumar, L. Adwani, S. Kumawat, and S. K. Jangir, "OpenNebula: Open Source IaaS Cloud Computing Software Platforms," in *National Conference on Computational and Mathematical Sciences (COMPUTATIA-IV), Technically Sponsored By: ISITA and RAOPS, Jaipur*.

[47] OpenNebula.org. *Planning the Installation*. Disponible en: *http://docs.opennebula.org/4.4/design_and_installation/building_your_cloud/plan*

[48] GitHub. *LXCone. Installation & Configuration Guide.* Disponible en: *https://github.com/OpenNebula/addon-lxcone/blob/master/Guide.md*

[49] OpenNebula.org. (marzo del 2016). *OpenNebula 4.14 Documentation*. Disponible en: *http://docs.opennebula.org/4.14/*

[50] OpenNebula.org. (abril del 2016). *Virtual Machine Definition File* Disponible en: http://docs.opennebula.org/4.14/user/references/template

[51] GitHub. (abril del 2016). *AppMarket*. Disponible en: *https://github.com/OpenNebula/addon-appmarket*

[52] GitHub. (abril del 2016). *AppMarket Installation and Configuration*. Disponible en: *https://github.com/OpenNebula/addon-appmarket/blob/master/doc/installation_and_configuration.md*

[53] E. K. Mena Maldonado, A. C. Guerrero Alemán, and I. Bernal Carrillo, "Implementación de un prototipo de Cloud Computing de modelo privado para ofrecer infraestructura como servicio (IaaS)," 2010.

[54] G. Reese, *Cloud application architectures: building applications and infrastructure in the cloud*: " O'Reilly Media, Inc.", 2009.

[55] J. García Núñez, J. Hernández Sánchez, and D. Molina Aranda, "An implementation of the Sun Cloud API for the OpenNebula Toolkit," 2010.

[56] OpenNebula.org. (marzo del 2016). *Advanced Contextualization*. Disponible en: *http://docs.opennebula.org/4.14/user/virtual_machine_setup/cong*

GLOSARIO

Amazon EC2 *Amazon Elastic Compute Cloud* es una parte central de la plataforma de cómputo en la nube de la empresa Amazon.com denominada (AWS). Permite a los usuarios alquilar computadores virtuales en los cuales poder ejecutar sus propias aplicaciones.

Amazon S3 *Simple Storage Service*, es un almacenamiento de archivos en línea servicio web ofrecido por *Amazon Web Services*. Ofrece almacenamiento a través de servicios web interfaces

Amazon Web Site Es una colección de servicios de computación en la nube (también llamados servicios *web*) que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por *Amazon.com*

Android Sistema operativo basado en el *kernel* de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, y también para relojes inteligentes, televisores y automóviles.

Apache Es una licencia de *software* libre permisiva creada por la *Apache Software Foundation* (ASF). Requiere la conservación del aviso de derecho de autor y el descargo de responsabilidad, pero no es una licencia *copyleft*, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

API Interfaz de programación de aplicaciones, es un conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción. Son usadas generalmente en las bibliotecas.

cgroups Grupos de control de Linux, permiten definir jerarquías en las que se agrupan los procesos de manera que un administrador puede definir con gran detalle la manera en la que se asignan los recursos (no solo tiempo de atención de CPU, sino también I/O y memoria principal) o llevar la contabilidad de los mismos.

CPU Unidad Central de Procesamiento o procesador, es el componente principal del ordenador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.

DevOps Acrónimo inglés de *development* (desarrollo) y *operations* (operaciones), que se refiere a una metodología de desarrollo de software que se centra en la

	comunicación, colaboración e integración entre desarrolladores de *software* y los profesionales de operaciones en las tecnologías de la información (IT)
DNS, Domain Name System	Sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.
DHCP, Dynamic Host Configuration Protocol	Protocolo de red que permite a los clientes de una red IP obtener sus parámetros de configuración automáticamente.
Grid Computing	Permite utilizar de forma coordinada todo tipo de recursos (entre ellos cómputo, almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado
HPC, High performance Computing	Computación de alto rendimiento, es una herramienta muy importante en el desarrollo de simulaciones computacionales a problemas complejos.
HTML, HyperText Markup Language	El lenguaje de marcas de hipertexto hace referencia al lenguaje de marcado para la elaboración de páginas web.
IBM, International Business Machines	Empresa multinacional que fabrica y comercializa *hardware* y *software* para computadoras, y ofrece servicios de infraestructura, alojamiento de Internet, y consultoría en una amplia gama de áreas relacionadas con la informática, desde computadoras centrales hasta nanotecnología.
ISO	Organización Internacional de Normalización, es una organización para la creación de estándares internacionales compuesto por diversas organizaciones nacionales de estandarización.
JavaScript	Es un lenguaje de programación interpretado, dialecto del estándar *ECMAScript*. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
kernel	Es el principal responsable de facilitar a los distintos programas acceso seguro al *hardware* de la computadora o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema, también conocido como núcleo.
loop device	En sistemas operativos *Unix*, es un *pseudo*-dispositivo que hace que se pueda acceder a un fichero como un dispositivo de bloques.
LVM, Logical Volume Manager	Es una implementación de un administrador de volúmenes lógicos para el *kernel* Linux.
Mashup (aplicación web	En desarrollo web, aplicación que usa y combina datos, presenta tonalidad procedentes de una o más fuentes para crear nuevos servicios.

híbrida)	
Proxmox Entorno Virtual	Es un entorno de virtualización de servidores de código abierto. Permite desplegar y gestionar máquinas virtuales y contenedores.
Rackspace	Compañía de gestión de computación en la nube de Estados Unidos.
REST, Representational State Transfer	La Transferencia de Estado Representacional es una técnica de arquitectura *software* para sistemas hipermedia distribuidos como la *World Wide Web* (www).
Ruby	Lenguaje de programación interpretado, reflexivo y orientado a objetos.
SCP, Secure Copy	Medio de transferencia segura de archivos informáticos entre un host local y otro remoto o entre dos hosts remotos, usando el protocolo *Secure Shell* (SSH).
Smartphones	Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades semejantes a una minicomputadora y conectividad que un teléfono móvil convencional.
SLA, Service Level Agreement	Contrato escrito entre un proveedor de servicio y su cliente con objeto de fijar el nivel acordado para la calidad de dicho servicio.
SOAP, Simple Object Access Protocol	Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.
SSH (Secure Shell)	Es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red.
SUSE	Distribución de Linux, se considera que sea una de las más sencillas de instalar y administrar.
Utility computing	Suministro de recursos computacionales, como puede ser el procesamiento y almacenamiento, como un servicio medido similar a las utilidades públicas tradicionales (como la electricidad, el agua, el gas natural o el teléfono).
VT-X Vanderpool	Tecnología de virtualización de Intel
XML-RPC	Protocolo de llamada a procedimiento remoto que usa *XML* para codificar los datos y *HTTP* como protocolo de transmisión de mensajes

ANEXOS

Anexo I.....Arquitectura de *Cloud Computing*

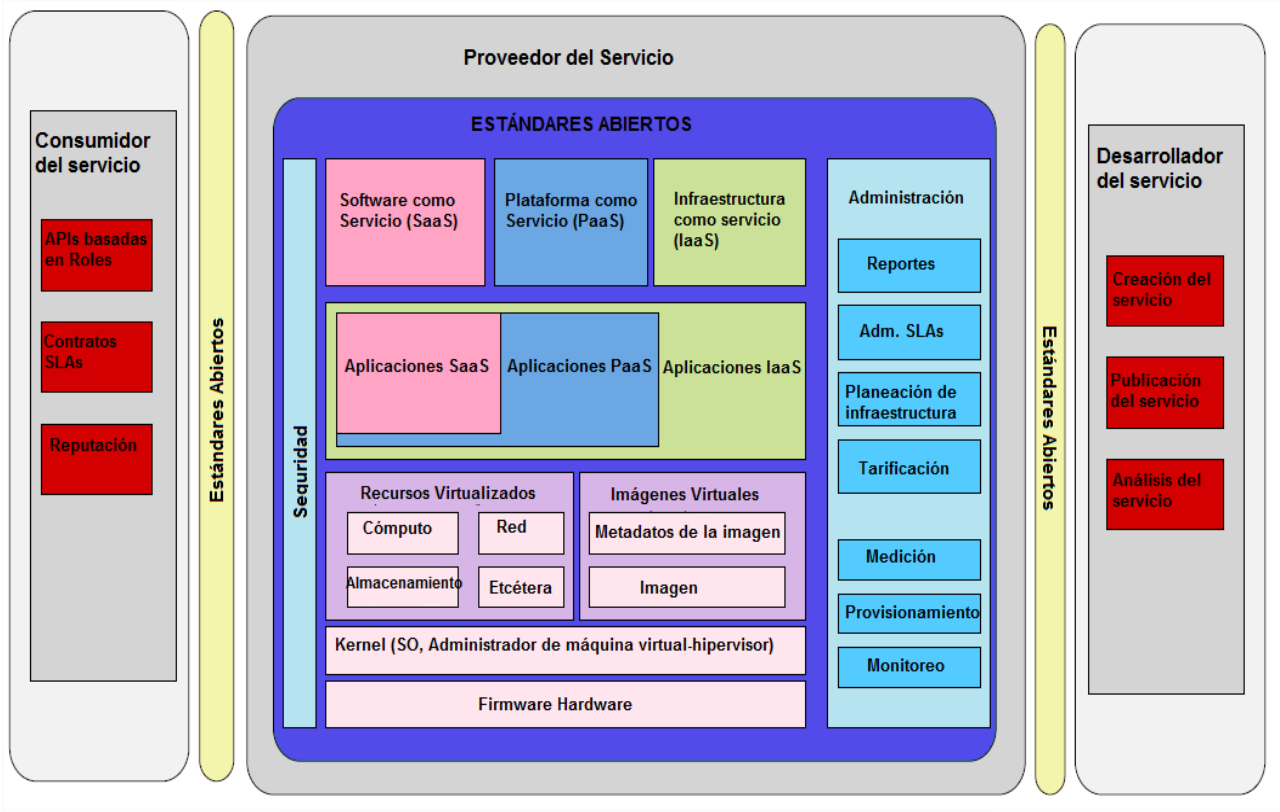

Figura I.1 Taxonomía del modelo *Cloud Computing* [53].

En el diagrama de la Figura I.1 del proveedor de servicios, la capa inferior de la pila es el *firmware* y *hardware* en el que se basa todo lo demás. Por encima del *firmware* está el núcleo de *software* (*kernel*), ya sea el sistema operativo o el gestor de la máquina virtual (*Virtual Machine Manager*, VMM) que aloja la infraestructura por debajo de la nube. Los recursos virtuales y las imágenes incluyen los servicios en la nube de computación básicos

tales como la potencia de procesamiento, almacenamiento y herramienta de gestión (*middleware*). Las imágenes virtuales se controlan por el VMM, estas incluyen tanto las propias imágenes, así como los metadatos necesarios para su gestión.

La capa de gestión es crucial para las operaciones del proveedor de servicios. A nivel bajo, la gestión requiere de: medición para determinar quién utiliza los servicios y en qué medida, el aprovisionamiento para determinar cómo se asignan los recursos a los consumidores, y monitoreo para el seguimiento del estado del sistema y sus recursos.

A alto nivel, la gestión incluye la facturación para recuperar los costos, la gestión de SLAs para asegurar que los términos del servicio acordado entre el proveedor y el consumidor se cumplan, y la presentación de informes a los administradores para llevar mayor control acerca del proceso.

La seguridad y los estándares se aplican a todos los aspectos de las operaciones del proveedor de servicios. Un conjunto bien definido de normas y estándares abiertos simplifican las operaciones de la nube del proveedor y la interoperabilidad con las mismas de otros proveedores.

El desarrollador de servicio, ubicado en la parte derecha de la Figura I.1, crea, edita y supervisa el servicio prestado por las nubes [7].

Durante la creación de servicios, el análisis de éstos, implica la depuración remota para probar el servicio antes de su publicación para los consumidores. Una vez que el servicio es publicado, el análisis permite a los desarrolladores supervisar el rendimiento de su servicio y posteriormente hacer cambios si fuera necesario [54].

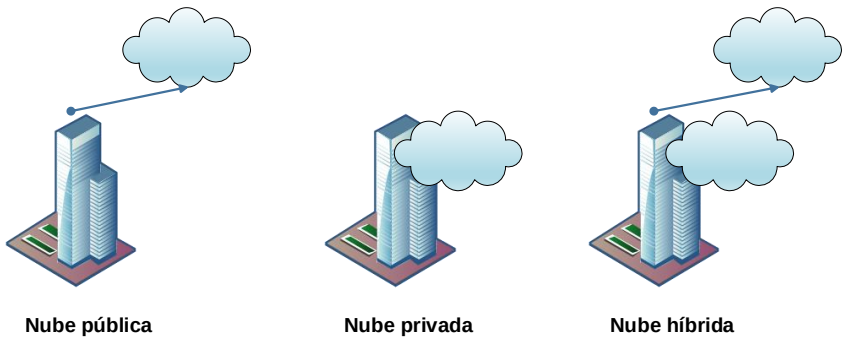

Figura I.2 Modelos de implementación de *Cloud Computing.*

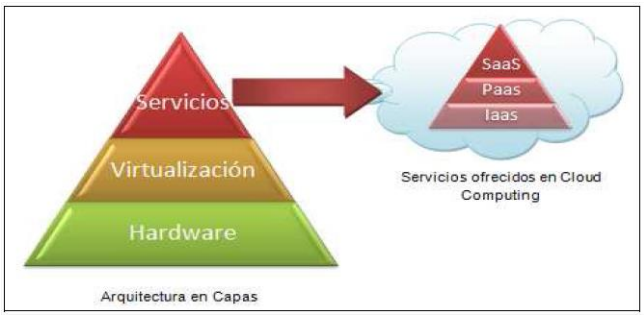

Figura I.3 Arquitectura en capas y modelo de servicio en *Cloud Computing* [53].

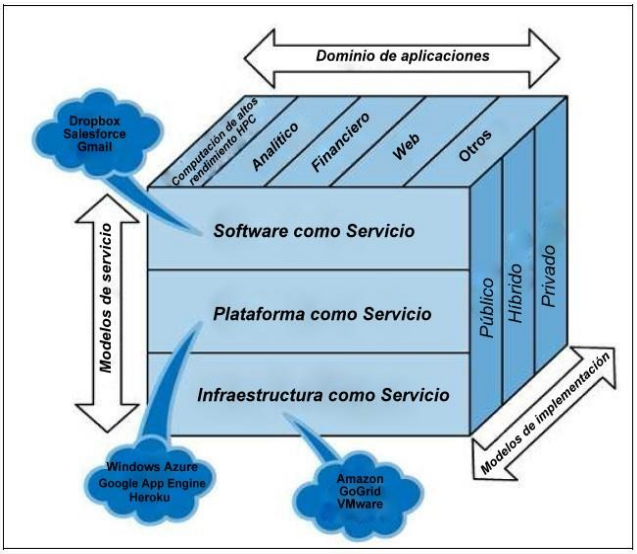

Figura I.4 Relación entre los servicios, aplicaciones y tipos de Nubes [53].

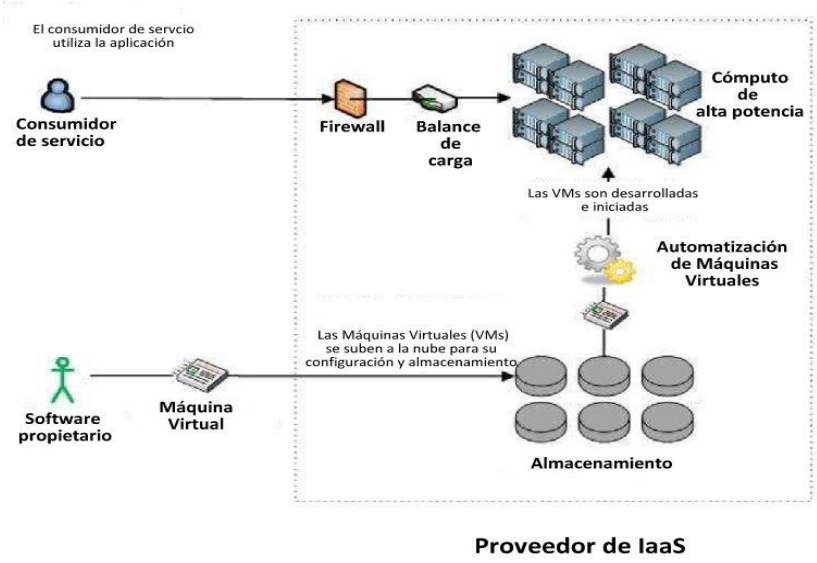

Figura I.5 Esquema de una Infraestructura como Servicio [55].

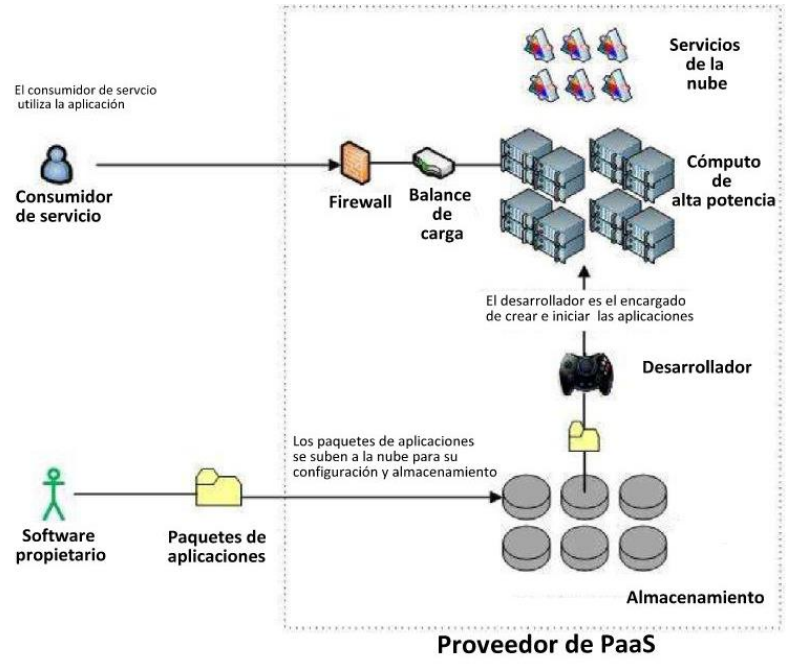

Figura I.6 Esquema de una Plataforma como Servicio [55].

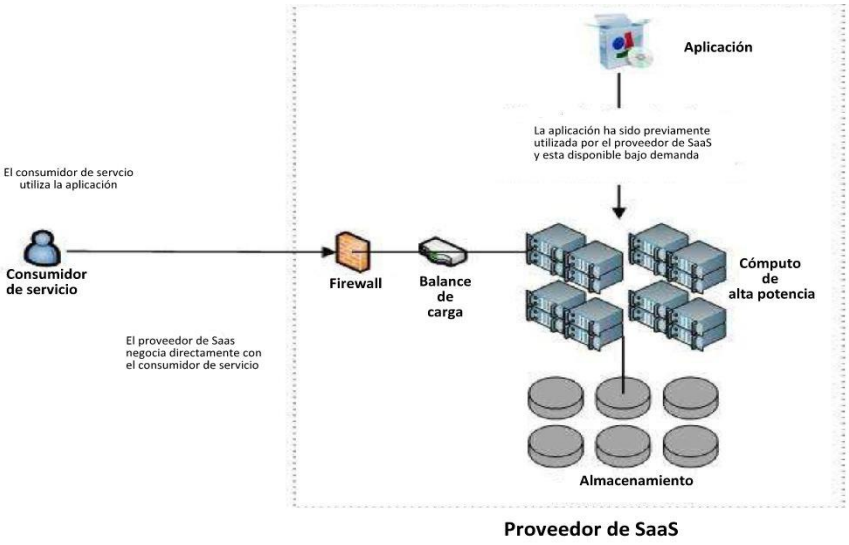

Figura I.7 Esquema de un *Software* como Servicio [55]**.**

Anexo II ..... Virtualización.

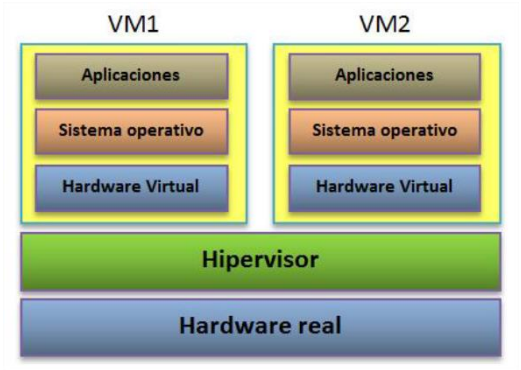

Figura II.1 Hipervisor del tipo sin sistema anfitrión.

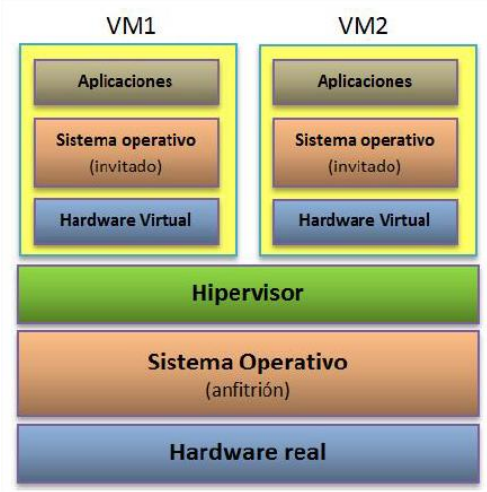

Figura II.2 Hipervisor del tipo con sistema anfitrión**.**

Tabla I.1 Desventajas de la virtualización.

Desventajas	Descripción
Necesidad de un mayor espacio de almacenamiento	Se debe hacer un replanteamiento y una correcta planificación sobre el almacenamiento, teniendo en cuenta que se requieren copias de seguridad, pruebas, etcétera
Hay sistemas no virtualizables o que no tienen soporte del fabricante en entornos virtualizados	Existen problemas de compatibilidad con el hardware virtualizado, un ejemplo de ello son las tarjetas gráficas, ni todos los SO pueden ser virtualizados
Preparación previa para la administración de virtualización	Aparentemente es muy simple pero sin una adecuada preparación puede dar verdaderos problemas.
Menor Rendimiento que los servidores físicos	En la virtualización se introduce una capa intermedia (el hipervisor) la cual gestiona las peticiones de los servidores virtualizados, lo que reduce el rendimiento
Avería del computador anfitrión	Un daño o fallo en un host afecta a los demás sistemas virtualizados

Anexo III.....Configuración de OpenNebula con hipervisor KVM en Debian/Ubuntu

Los pasos mostrados a continuación son válidos tanto para los sistemas operativos Debian y Ubuntu a menos que se indique lo contrario. Se toma como convención que los comandos que comiencen con # son desde el usuario *root*, por otra parte las que inicien con $, desde el usuario *oneadmin*.

A lo largo de la instalación se realizan dos funciones separadas: interfaz de administración (*Frontend)* y Nodos**.** El servidor *Frontend* ejecuta los servicios de OpenNebula, y en los nodos se ejecutan las máquinas virtuales. Es un requisito indispensable ejecutar las VMs en anfitriones con extensiones de virtualización. Para probar si el anfitrión es compatible con las extensiones de virtualización (en este caso KVM), se ejecuta:

```
# grep -E 'svm|vmx' /proc/cpuinfo
```

Si el comando anterior no devuelve ninguna salida es probable el servidor no soporte extensiones de virtualización.

Instalación del *Frontend*

Paso 1: Instalar los repositorios

Añadir la llave para el repositorio de OpenNebula y luego la entrada correspondiente del repositorio según el sistema operativo empleado, en este caso Ubuntu. La configuración de los repositorios se encuentra disponible en:

http://redtic.uclv.cu/dokuwiki/reposwiki#opennebula

Los repositorios de la UCLV poseen la instalación de OpenNebula 14.04 para Debian Jessie y Ubuntu 14.04 LTS.

Para Ubuntu 14.04 LTS se procede de la siguiente manera:

```
# wget -q -O- http://repos.uclv.cu/keys/opennebula.key | apt-key add –
# echo "deb http://repos.uclv.edu.cu/opennebula/4.14/Ubuntu/14.04/ \
stable opennebula" \  > /etc/apt/sources.list.d/opennebula.list
```

En caso de emplear Debian para instalar el *Frontend*, solo hay que cambiar la línea que hace referencia al repositorio por:

```
deb http://repos.uclv.edu.cu/opennebula/4.14/Debian/8 stable opennebula
```

Paso 2: Instalar los paquetes requeridos

Con el objetivo de actualizar el sistema e instalar los servicios requeridos, ejecutar:

```
# apt-get update
# apt-get upgrade
# apt-get install opennebula opennebula-sunstone nfs-kernel-server
```

Paso 3: Configurar e iniciar los servicios

Existen dos procesos principales que se deben iniciar, el demonio principal de OpenNebula: *oned*, y la interfaz gráfica de usuario: *sunstone*.

Sunstone sólo escucha en la interfaz local (*loopback*) por defecto, por razones de seguridad. Para cambiar este comportamiento, se modifica el archivo "*/etc/one/sunstone-server.conf*", donde se cambia la línea *host:* 127.0.0.1 a: *host*: 0.0.0.0. Luego se inicia el servicio *sunstone*:

```
# /etc/init.d/opennebula-sunstone start
```

Paso 4: Configuración de Sistema de Archivo de Red (*Network File System*, NFS)

Se exporta el directorio "*/var/lib/one/*"desde el *Frontend* a los nodos, para esto se agrega la siguiente línea a archivo "*/etc/exports/*".

```
/var/lib/one/      *(rw,sync,no_subtree_check,root_squash)
```

A continuación se reinicia el servicio:

```
# service nfs-kernel-server restart
```

En caso de emplear varios *datastores* en servidores NFS distintos se deben de compartir cada uno por separado hacia los nodos e incluirlos en el *Frontend*. Ejemplo compartir *"/var/lib/one/datastore/<datasotore_id>"*.

Paso 5: Configurar la clave pública SSH

OpenNebula necesita contraseñas SSH desde cualquier nodo (incluye la interfaz) a cualquier otro nodo. Para hacerlo, se ejecutan los siguientes comandos desde el usuario *oneadmin*:

```
# su – oneadmin
$ cp ~ / .ssh / id_rsa.pub ~ / .ssh / authorized_keys
```

Luego se añade el siguiente fragmento al archivo "*~/.ssh/config"* para desactivar las alertas de autenticidad del nodo al conectarse por ssh y se ajustan los permisos necesarios:

```
$ cat << EOT > ~/.ssh/config
Host *
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
EOT
$ chmod 600 ~/.ssh/config
```

Instalación de los nodos

Paso 1: Instalación de los repositorios

Se repite el paso 1 de la instalación del *Frontend*, lo que ahora desde el nodo que se va a instalar.

Paso 2: Instalar los paquetes necesarios

Con el objetivo de actualizar el sistema e instalar los servicios requeridos, ejecutar:

```
# apt-get update
# apt-upgrade
# apt-get install opennebula-node nfs-common bridge-utils
```

Paso 3: Configuración de la red

Se necesita una segunda interfaz red, normalmente *eth1*, conectada a un puente (*bridge*). El nombre del puente debe ser el mismo en todos los nodos, en este caso *br0*.

La configuración de red requerida es mostrada a continuación:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
       address 10.12.112.141
       network 10.12.112.0
       netmask 255.255.255.0
       broadcast 10.12.112.255
       gateway 10.12.112.254

       dns-nameservers 10.12.1.50 10.12.1.51
       dns-search uclv.edu.cu uclv.cu

auto br0
iface br0 inet static
       address 10.12.114.1
       network 10.12.114.0
       netmask 255.255.255.0
       broadcast 10.12.114.255
       gateway 10.12.114.254
       bridge_ports eth1
       bridge_fd 9
       bridge_hello 2
       bridge_maxage 12
       bridge_stp off
```

Después de realizados los cambios reiniciar el servidor:

```
# reboot
```

Paso 4: Configuración de NFS

Se monta el recurso compartido en el paso 4 de la instalación del *Frontend* en el directorio "*/var/lib/one*" de cada uno de los nodos, para ello se añade lo siguiente a archivo "*/etc/fstab"*:

```
10.12.112.140:/var/lib/one/      /var/lib/one/    nfs \
soft,intr,rsize=8192,wsize=8192,noauto
```

Donde la dirección IP 10.12.112.140 es la del *Frontend*, para aplicar los cambios:

```
# mount /var/lib/one/
```

Para asegurar que cada vez que se reinicien los nodos se monte correctamente el recurso NFS especificado anteriormente, agregar la siguiente línea en el archivo "/*etc/rc.local*":

```
mount /var/lib/one/
```

Paso 5: Configuración de Qemu

Verificar que el contenido del archivo "*/etc/libvirt/qemu.conf*" se ajuste a:

```
user  = "oneadmin"
group = "oneadmin"
dynamic_ownership = 0
```

En caso de modificarlo se reinicia el servicio *libvirt* para aplicar los cambios.

Para Ubuntu 14.04 LTS:

```
# service libvirt-bin restart
```

Para Debian Jessie:

```
# systemctl restart libvirtd
```

Acceso a la interfaz web Sunstone

Para acceder a la interfaz *web* Sunstone (figura III.1), se especifica en el navegador la dirección IP del servidor *Frontend* y el puerto 9869, en este caso: *http://10.12.112.140:9869*. La contraseña del usuario *oneadmin* se encuentra localizada "*~/.one/one_auth"*, y es generada de forma aleatoria en cada instalación.

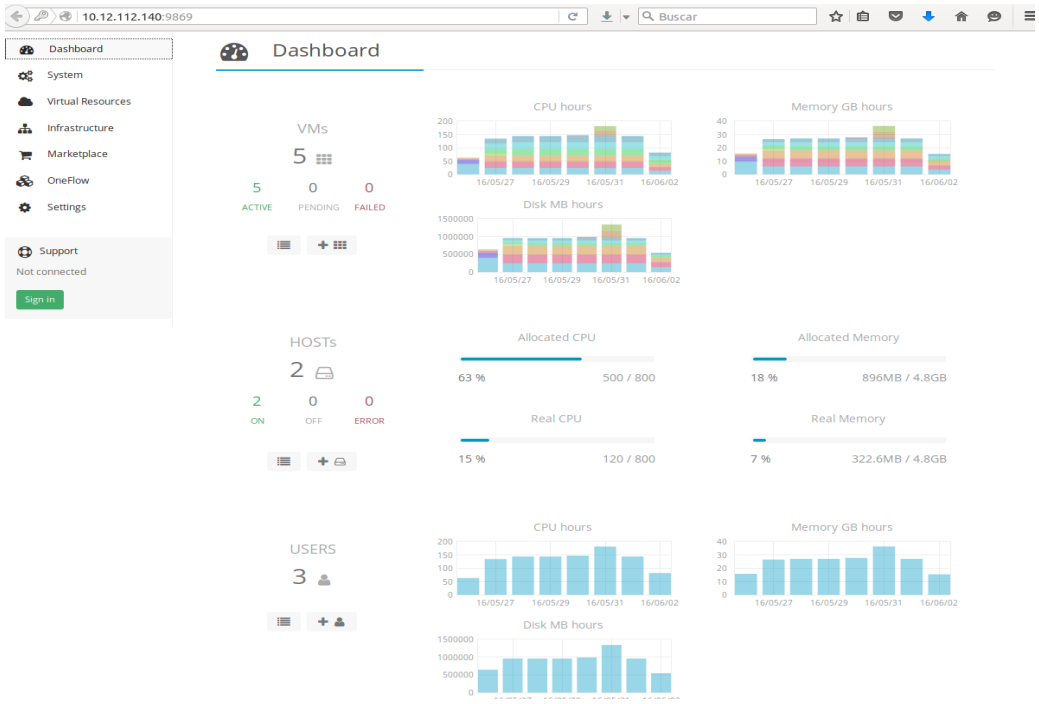

Figura III.1 Interfaz Sunstone.

Agregar los nodos el *Frontend*

Siempre se interactúa con OpenNebula desde el usuario *oneadmin*, para acceder a través del mismo se ejecuta:

```
# su – oneadmin
```

Antes de unir nodos se comprueba que el archivo *"/etc/hosts" c*ontenga el nombre del *Frontend* y de los nodos (esto es necesario hacerlo para el *Frontend* y cada nodo que se va a agregar), por ejemplo:

```
10.12.112.140       frontend.uclv.edu.cu      frontend
10.12.112.141       nebula1.uclv.edu.cu       nebula1
10.12.112.142       nebula2.uclv.edu.cu       nebula2
10.12.112.145       nebula3.uclv.edu.cu       nebula3
10.12.112.146       nebula4.uclv.edu.cu       nebula4
```

Paso 1: Añadir un nodo

Para correr las VMs, primero de deben registrar los nodos de OpenNebula. Se ejecuta el siguiente comando para cada uno de los nodos. Se sustituye *nebula1* por el nombre de cada nodo, ejemplo nebula1, nebula2, etc. El controlador de red a emplear es *dummy*, pero se puede utilizar cualquiera de los mencionados en la sección 2.3.6 del capítulo 2.

```
$ onehost create nebula1 -i kvm -v kvm -n dummy
```

Las opciones que hay que pasar al agregar un nodo se describen a continuación:

- --*im/-i*: Información del controlador. Opciones válidas*: kvm, xen, vmware, ec2, ganglia, dummy*.
- --*vm/-v*: Controlador de VMs. Opciones válidas: *kvm, xen, vmware, ec2, dummy*.
- --*net/-n*: *C*ontrolador de red. Opciones válidas: *802.1Q, dummy, ebtables, fw, ovswitch, vmware*.

Una vez que se crea el anfitrión se comprueba que esté activo con el comando *onehost list*. Si falla es probable que se tenga algún problema en la configuración de ssh, por lo que se puede verificar el archivo "*/var/log/one/oned.log"*.

Agregar los recursos virtuales

Una vez que se ha agregado el nodo a OpenNebula, se procede a crear la red de virtualización y la plantilla o plantillas para las máquinas virtuales.

Paso 1: Creación de la red de virtualización

Para crear la red o redes, primero se crea la plantilla de red, por ejemplo *network114KVM.one* con el siguiente contenido:

```
NAME = "red114KVM"
BRIDGE = br0

# Context attributes
NETWORK_ADDRESS = "10.12.114.0"
NETWORK_MASK    = "255.255.255.0"
DNS             = "10.12.1.50"
GATEWAY         = "10.12.114.254"

AR = [
TYPE = IP4,
IP = 10.12.114.10,
SIZE = 5
]
```

La dirección IP es la primera del rango de direcciones que se va a asignar a la red y el tamaño es en dependencia de las necesidades, en este caso es una red de 5 IPs. A continuación se crea la red a partir de las especificaciones en el archivo anterior:

```
$ onevnet create network114KVM.one
```

Es posible comprobar si la red fue creada satisfactoriamente con el siguiente comando:

```
$ onevnet list
```

Paso 2: Creación de la imagen para las VMs

Siempre que se desee crear una imagen se necesita aclarar, el nombre, el camino donde se encuentra la imagen que puede ser local o desde internet (en este caso especificar ella URL completa de la imagen o descargarla desde el *Marketplace*), el controlador de imagen, y que almacén de datos (*datastore*) se empleará para almacenarla. La descripción es opcional y desde la interfaz *web* Sunstone puede agregarse o/y modificarse.

A través del comando *oneimage* se procede a crear la imagen:

```
$ oneimage create --name "CentOS-7.1_x86_64" \
--path "/srv/KVM_images/centos-7.1.qcow2c" --driver qcow2 --datastore default \
--description "This image is based off a CentOS 7.1 cloud image with the
OpenNebula contextualization package."
```

Se tiene que esperar hasta que la imagen esté lista para usarse, se supervisa su estado (columna *STAT*) mediante el comando *oneimage list,* si todo marcha bien debe mostrarse algo como se muestra a continuación:

USER	GROUP	NAME	DATASTORE	SIZE	TYPE	PER	STAT	RVMS
oneadmin	oneadmin	CentOS-7_x86_64	default	919M	OS	No	rdy	0

Paso 3: Creación de la plantilla para las VMs

En OpenNebula las VMs son definidas a través de una plantilla, la cual puede ser creada desde la interfaz *web* o desde consola como se muestra a continuación:

```
$ onetemplate create --name "CentOS-7.1" --cpu 1 --vcpu 1 --memory 256 \
--arch x86_64 --disk "CentOS-7.1_x86_64" --nic "red114KVM" --vnc --ssh \
--net_context
```

Es importante aclarar el nombre de la red y de la imagen creada debe de coincidir con el nombre que se le pase a la opción *"--nic"* y *"--disk"* respectivamente del comando *onetemplate*. La opción *"--net_context"* permite activar la red de las imágenes

contextualizadas del *Marketplace*, esta opción es obligada para poder activar la red en las VMs.

Cuando se emplea un entorno heterogéneo de hipervisores (KVM y LXC), cada plantilla creada hay que especificarle como requerimiento que emplee uno u otro tipo de hipervisor. Desde la interfaz web en las plantillas de KVM, en la sección *Scheduling -> Placement* especificar:

```
HYPERVISOR=\"kvm\"
```

Para las plantillas de hipervisores LXC:

```
HYPERVISOR=\"lxc\"
```

Desde la CLI es posible crear la plantilla con todos los parámetros requeridos para este caso:

```
$ onetemplate create --name "CentOS-7.1" --cpu 1 --vcpu 1 --memory 256 \
--arch x86_64 --disk "CentOS-7.1_x86_64" --nic "red114KVM" --vnc --ssh \
--net_context --dry > CentOS-7.1.tmpl
```

Luego editar el archivo *"CentOS-7.1.tmp"*, agregar los requerimientos del planificador y el tipo de hipervisor como se muestra:

```
NAME="CentOS-7.1-test"
OS = [
  ARCH = "x86_64" ]
CPU=1.0
VCPU=1
MEMORY=128
DISK=[
  IMAGE="CentOS-7.1_x86_64"
]
NIC=[
  NETWORK="red114KVM"
]
GRAPHICS=[ TYPE="vnc", LISTEN="0.0.0.0" ]
CONTEXT=[
  SSH_PUBLIC_KEY="$USER[SSH_PUBLIC_KEY]",
  NETWORK = "YES"
]
## Scheduler requeriments
SCHED_REQUIREMENTS = "HYPERVISOR=\"kvm\""
## Hypervisor type
HYPERVISOR="kvm"
```

Finalmente crear la plantilla a partir del archivo "*CentOS-7.1.tmpl"*:

```
$ onetemplate create CentOS-7.1.tmpl
```

Por motivos de seguridad las imágenes de máquinas virtuales que se descargan del *Marketplace* de OpenNebula no poseen contraseña de *root*, solo es posible acceder con claves ssh. Se sigue esta política para evitar que un usuario pueda acceder a la VM de otro, si este no ha cambiado la contraseña. Para ello estas imágenes traen instalado el paquete de red *one-context*, que se encarga de habilitar la red y modificar la clave ssh del usuario *root* en el arranque [56].

Para que esto funcione tiene que estar definida la variable *SSH_PUBLIC_KEY* del usuario que arranque la máquina (por ejemplo *oneadmin*) o incluir esta variable dentro de la sección *CONTEXT* del *template* de la máquina virtual, por lo que hay que añadir la llave ssh a la plantilla de usuario, primero es necesario copiar la salida del siguiente comando en el portapapeles:

```
$ cat ~/.ssh/id_rsa.pub
```

Luego editar la plantilla del usuario *oneadmin* y agregar la variable *SSH_PUBLIC_KEY* con el contenido del comando anterior:

```
$ EDITOR=vi oneuser update oneadmin
SSH_PUBLIC_KEY=¨ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAA……¨
```

Creación de las máquinas virtuales

Para ejecutar una VM a partir de una plantilla creada solo es necesario especificar el nombre de la plantilla, así como el nombre del VM:

```
$ onetemplate instantiate "CentOS-7.1" --name "CentOS_VM1"
```

Se comprueba que la VM funcione correctamente con el comando:

```
 $ onevm list
```

El ciclo normal de inicio de una VM es de *PENDING* a *PROLOG* y finalmente *RUNNING*. Esto puede ser apreciado en la columna *STAT* al ejecutar el comando anterior. Si la máquina virtual falla, se comprueba el registro en el archivo: "*/var/log/one/<VM_ID>.log"*.

Anexo IV..... Configuración de OpenNebula con KVM en Debian con *Open vSwitch*

Si se desea emplear el controlador de red OVS, se siguen todos los pasos descritos en el Anexo III, excepto los paquetes a instalar en los nodos y la configuración de red.

1) En los nodos no se debe instalar el paquete *bridge-utils*, este paquete se sustituye por *openvswitch-switch.*

2) Configuración de red

   En este caso no se configuran los *bridges* de Linux como se explica en el Anexo III, *Open vSwitch* se encarga de gestionarlos, por lo que el archivo de configuración de red queda de la siguiente forma:

```
auto lo
 iface lo inet loopback

 auto eth0
 iface eth0 inet static
        address 10.12.112.141
        network 10.12.112.0
        netmask 255.255.255.0
        broadcast 10.12.112.255
        gateway 10.12.112.254

        dns-nameservers 10.12.1.50 10.12.1.51
        dns-search uclv.edu.cu uclv.cu
```

3) Crear un *bridge* OVS.

```
# ovs-vsctl add-br br0
```

4) Añadir la interfaz *eth1* al *bridge.*

```
# ovs-vsctl add-port br0 eth1
```

   Cuando se agrega *eth1* al puente OVS, se pierde la conectividad en dicha interfaz, de ahí que es un requisito indispensable tener al menos dos interfaces: una de administración y otra para virtualización.

   Para comprobar se utiliza el comando:

```
# ovs-vsctl show
```

5) Se une el nodo al *Frontend* especificando el controlador de red *ovswitch*.

```
$ onehost create nebula4 -i kvm -v kvm -n ovswitch
```

6) En la configuración de red especificar el controlador de red *ovswitch*, en caso de querer aislar las redes, activar la opción VLAN="YES" y especificar los IDs de las VLANs.

Independientemente se pueden tener mezcla de hipervisores unos con *bridge* de Linux y otros con OVS en OpenNebula siempre que se especifique el nombre del *bridge* a usar con las variables en la configuración de la red virtual (VNET):

```
BRIDGE="nombre de tu bridge linux"
BRIDGE_OVS="nombre de tu bridge OVS"
```

Anexo V ..... Configuración de OpenNebula con LXC en Debian/Ubuntu

Instalación del *Frontend*

Los pasos 1,2 y 3 coinciden con la instalación del *Frontend* del Anexo III.

Paso 4: Configuración de NFS

Se exporta "*/var/lib/one/*"y "*/var/log/one*" desde el *Frontend* a los nodos, para esto se agregan las siguientes líneas al archivo *"/etc/exports*":

```
/var/lib/one/ *(rw,sync,no_subtree_check,no_root_squash,crossmnt,nohide)
/var/log/one/ *(rw,sync,no_subtree_check,no_root_squash,crossmnt,nohide)
```

Luego se reinicia el servicio NFS:

```
# service nfs-kernel-server restart
```

Paso 5: Configurar la clave pública SSH

Este punto también coincide con el de la guía descrita en el Anexo III.

Paso 6: Copiar y ajustar los permisos del *driver* LXC

Primeramente es necesario descargar los archivos desde el repositorio *GitHub* del controlador LXC, los mismos se encuentran disponibles en la siguiente dirección: https://github.com/OpenNebula/addon-lxcone.

Para clonar el repositorio desde la consola:

```
# git clone https://github.com/OpenNebula/addon-lxcone.git
```

De forma alternativa el repositorio puede ser descargado como *zip*.

Luego acceder al directorio *addon-lxcone* y copiar la directorio *lxc* en el directorio "*/var/lib/one/remotes/vmm*" del *Frontend.*

Luego copiar los directorios *lxc.d* y *lxc-probes.d*, que se encuentran dentro del directorio *im*, en el directorio "*/var/lib/one/remotes/im*" del *Frontend.*

Se ajusta los permisos de usuarios y grupos:

```
# chown -R oneadmin:oneadmin /var/lib/one/remotes/vmm/lxc \
```

```
/var/lib/one/remotes/im/lxc.d /var/lib/one/remotes/im/lxc-probes.d
# chmod -R 755 /var/lib/one/remotes/vmm/lxc /var/lib/one/remotes/im/lxc.d \
/var/lib/one/remotes/im/lxc-probes.d
```

Paso 7: Agregar la configuración de los nuevos drivers a OpenNebula

La información referente a los nuevos controladores que gestionará OpenNebula se realiza desde el archivo *"/etc/oned/oned.conf*":

Luego localizar la parte inferior de la sección "***Information Driver Configuration"*** y adicionar:

```
#-------------------------------------------------------------------------------
# LXC Information Driver Manager Configuration
# -r number of retries when monitoring a host
# -t number of threads, i.e. number of hosts monitored at the same time
#-------------------------------------------------------------------------------
       IM_MAD = [
       name = "lxc",
       executable = "one_im_ssh",
       arguments = "-r 3 -t 15 lxc" ]
#-------------------------------------------------------------------------------
```

También debajo de **"*Virtualization Driver Configuration"*** añadir:

```
#-------------------------------------------------------------------------------
# LXC Virtualization Driver Manager Configuration
# -r number of retries when monitoring a host
# -t number of threads, i.e. number of actions performed at the same time
#-------------------------------------------------------------------------------
     VM_MAD = [ name = "lxc",
     executable = "one_vmm_exec",
     arguments = "-t 15 -r 0 lxc",
     type = "xml" ]
#-------------------------------------------------------------------------------
```

Reiniciar el servicio de OpenNebula.

```
# service opennebula restart
```

Por defecto, el servicio *opennebula-sunstone n*o se inicia automáticamente, para modificar esto, se adiciona la siguiente línea al archivo "*/etc/rc.local":*

```
service opennebula-sunstone start
```

Instalación de los nodos

Paso 1: Configurar los repositorios

Se repite el Paso 1 de la instalación del *Frontend*, lo que ahora desde el nodo que se va a instalar.

Paso 2: Instalación de los paquetes

Con el objetivo de actualizar el sistema e instalar los servicios requeridos, ejecutar:

```
# apt-get update
# apt-get upgrade
# apt-get install opennebula-node nfs-common bridge-utils xmlstarlet \
libpam-runtime bc at libvncserver0 libjpeg62 lxc
```

Es útil instalar el paquete *SVNCterm* para habilitar el servicio de VNC en los contenedores LXC, para ello acceder al directorio donde se clonó del repositorio de *LXC-one* y ejecutar:

```
# dpkg -i svncterm_1.2-1_amd64.deb
```

Paso 3: Configuración de la red

Este paso coincide con la configuración de la red de los nodos de KVM.

Paso 4: Configurar el NFS

Para crear el directorio "/*var/log/one*" si se ejecuta el siguiente comando:

```
# mkdir -p /var/log/one
```

Se adicionan las siguientes líneas a "/*etc/fstab*":

```
10.12.112.140:/var/lib/one/         /var/lib/one/   nfs  \
soft,intr,rsize=8192,wsize=8192,auto

10.12.112.140:/var/log/one/    /var/log/one/    nfs \
soft,intr,rsize=8192,wsize=8192,auto
```

Luego se montan los directorios especificados anteriormente:

```
# mount /var/lib/one
# mount /var/log/one
```

En ocasiones es posible que no se monten correctamente los directorios mencionados al reiniciar el servidor por lo que se recomienda agregar los dos comandos anteriores en el archivo *"/etc/rc.local*".

Paso 5: Asignar los privilegios necesarios para el usuario *oneadmin*

Se añade el usuario *oneadmin* el archivo *sudoers*, y se habilita para que ejecute comandos desde *root* sin contraseña.

Para ello se ejecuta el comando *visudo* y se agrega, justamente debajo de línea

"*root ALL=(ALL:ALL) AL*", lo siguiente:

```
oneadmin ALL= NOPASSWD: ALL
```

Finalmente escribir la siguiente instrucción para permitir al usuario *oneadmin* ejecute los *scripts* que están en el directorio del contenedor:

```
# chmod +rx /var/lib/lxc
```

Paso 6: Activar la capacidad límite de memoria

Se chequea que la capacidad de memoria del *cgroup* esté disponible:

```
# cat /proc/cgroups| grep memory | awk '{ print $4 }'
```

El "0" significa que la capacidad no está disponible y el "1" lo contrario.

Para activar la capacidad de límite de memoria (*memory limit capability*) se adiciona el argumento *cgroup* al *grub*. Para ello se añade dentro del archivo "*/etc/default/grub"* en la directiva GRUB_CMDLINE_LINUX, los parámetros *cgroup_enable=memory* y *swapaccount=1*, por ejemplo*:*

```
GRUB_CMDLINE_LINUX="cgroup_enable=memory swapaccount=1"
```

Luego se actualiza la configuración de *grub* mediante*:*

```
# update-grub
```

Paso 7: Añadir dispositivos de lazo

Cada imagen del sistema de archivo que usa LXC a través de este controlador requiere de un dispositivo de lazo (*loop device*). Como el límite por defecto es 8, se necesita

modificarlo. Por lo que se debe de editar el archivo "*/etc/modprobe.d/local-loop.conf*" y agregar la siguiente línea:

```
options loop max_loop=64
```

Para activar el lazo automáticamente, se escribe *loop* al final del archivo "*/etc/modules"*.

Se reinicia el nodo para habilitar todos los cambios realizados en los pasos anteriores:

```
# reboot
```

Crear una imagen para los contendores LXC

Paso 1: Crear una imagen mediante LXC

Se desea crear una imagen de 3 GB dentro de un contenedor de Linux, donde el archivo de la imagen se localiza en *"/var/lib/lxc/<name>/rootdev*" y su nombre es el del contenedor. Para ello:

```
# lxc-create -t debian -B loop --fssize=3G -n name
```

Este paso requiere Internet o modificar la plantilla a usar en "/*usr/share/lxc/templates*".

Paso 2: Configurar el contenedor

Primero copiar la clave de "*root*" al finalizar el comando *lxc-create*, el cual devuelve algo como:

```
Root password is '3KyNv10m', please change !
```

Luego asegurarse que en el archivo "*/var/lib/lxc/<name>/config"* esté la siguiente línea:

```
write lxc.autodev = 1
```

La opción anterior permite llenar el directorio "*/dev*" al iniciar el contenedor, lo cual es requerido en sistemas que usan *systemd*.

Puede darse el caso que al iniciar el contenedor no tenga la red activa, para ello es necesario agregar el archivo "*/var/lib/lxc/<name>/config"* la configuración de red siguiente*:*

```
# ETH0
lxc.network.type = veth
lxc.network.flags = up
lxc.network.name = eth0
lxc.network.link = br0
```

```
lxc.network.ipv4 = 10.12.114.9/24
lxc.network.ipv4.gateway = 10.12.112.254
```

Tanto la red como la puerta de enlace deben ser ajustadas según el caso.

Iniciar el contenedor:

```
# lxc-start -n name -d
```

Si se desea ver su estado:

```
# lxc-info -n name
```

Para listar todos los contenedores:

```
# lxc-ls -f
```

Paso 3: Acceder a la consola del contenedor

El siguiente comando nos permite acceder al contenedor y personalizarlo según las necesidades del usuario:

```
# lxc-console -n name
```

Nota: para salir de la consola presionar Ctrl-A seguido de Q.

Puede resultar más conveniente el siguiente comando para acceder al contendor, ya que no hay que especificar la contraseña:

```
# lxc-attach -n name
```

Una vez dentro en caso de usar repositorios locales se deben de configurar primeramente:

```
# cat << EOT > /etc/apt/sources.list
deb http://repos.uclv.edu.cu/debian jessie main non-free contrib
deb http://repos.uclv.edu.cu/debian jessie-updates main non-free contrib
deb   http://repos.uclv.edu.cu/debian-security   jessie/updates   main   non-free
contrib
EOT
# apt-get update
```

Debido a que la plantilla generada es muy básico, se recomienda instalar los siguientes paquetes:

```
# apt-get install tasksel
# tasksel install standard
# apt-get install mc openssh-server
```

Una vez finalizado todos los cambios se procede a detener el contenedor:

```
# lxc-stop –n name
```

Paso 4: Ajustar la imagen en un sector de 512 bytes

Las imágenes que se crean con LXC no se montan bien, ya sea en un dispositivo de lazo o un administrador de volúmenes lógico (LVM), en caso que se use LVM. Esto ocurre porque el tamaño de imágenes creadas con LXC no es múltiplo de 512 Bytes. Para solucionar esto se procede a crear una imagen vacía que sea múltiplo de 512 bytes y a continuación se clona la imagen creada con el comando *lxc-create* hacia la nueva imagen.

Los pasos consisten en:

1) Crear una imagen vacía:

Esto se puede hacer por varias vías, se escoge la siguiente:

```
# qemu-img create -f raw image_name <SIZE_IN_MB>M
```

Para seleccionar el tamaño correcto de la imagen multiplique 1024 por la cantidad de GB deseados, ejemplo: 3GB = 1024MB x 3 = 3072MB.

2) Clonar el contenido de la imagen creada con el comando *lxc-create* en la imagen vacía creada con el comando *qemu-img*:

```
# dd if=/var/lib/lxc/**name**/rootdev of=image_name bs=512 conv=notrunc
```

Agregar los recursos virtuales

Se accede a la interfaz *web* desde la dirección *http://10.12.112.140:9869* y la contraseña del usuario *oneadmin* se encuentra en el archivo *“var/lib/one/.one/one_auth”*.

Paso 1: Subir la imagen creada con LXC a OpenNebula a través de la interfaz gráfica Sunstone

NOTA: Sunstone ofrece varias formas para subir la imagen al servidor de OpenNebula: especificando el camino (remoto o local) o subiéndola a través de un formulario vía *web*. En caso de que Sunstone no pueda determinar el tamaño de la imagen al especificarla de forma local, se recomienda subirla primero a un servidor *web* y especificar la URL donde se encuentra publicada.

Para crear una imagen se deben especificar los campos, ver Figura IV.1:

- *Name* (Nombre).
- *Type* (Tipo). Seleccionar Sistema Operativo (SO).
- *Imagen Location* (Localización de la imagen). En caso que la imagen esté en un servidor *web*, seleccionar "*Provide a path"*, y especificar la URL.
- *Datastore*. Lugar donde se almacenará la imagen subida. Por defecto se emplea el predeterminado.

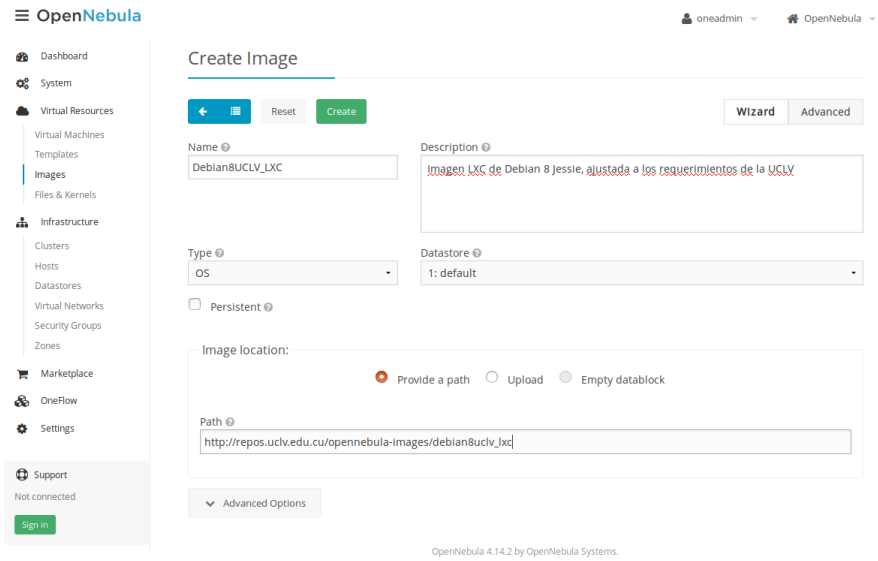

Figura IV.1 Crear una imagen en Sunstone.

Paso 2: Crear un nodo

Para crear un nodo se necesitan especificar las siguientes características, como se muestra en la Figura IV.2:

- *Type* (Tipo). Seleccionar ***Custom***.
- Escribir la dirección IP del nodo donde fue instalado y configurado LXC. También se acepta el nombre del nodo, o el DNS en caso de ser configurado previamente.
- Dentro de ***Drives***
    - *Virtualization* (Virtualización). Seleccionar ***Custom***
    - *Information* (Información). Seleccionar ***Custom***
    - Custom VMM_MAD. Escribir ***lxc***
    - Custom IM_MAD. Escribir ***lxc***

En este caso se usa el controlador de red por defecto de OpenNebula, *dummy*.

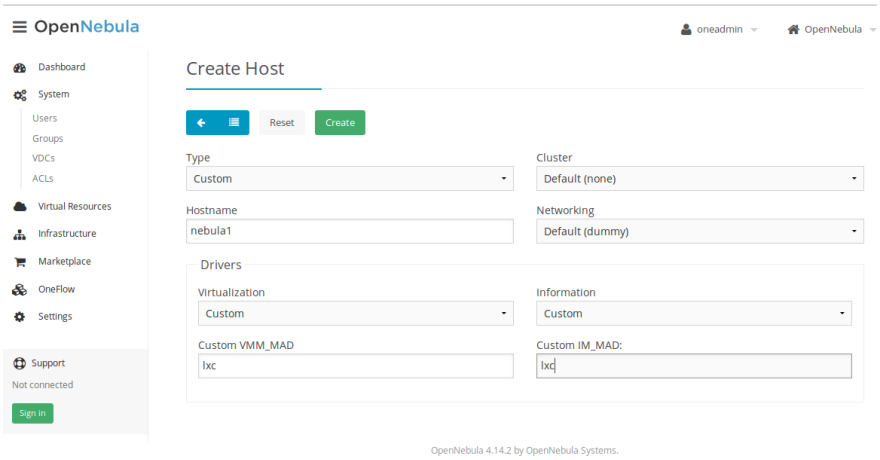

Figura IV.2 Crear un nodo LXC en Sunstone.

Paso 3: Crear una red virtual

Para crear una red virtual se necesitan especificar las siguientes características, como se muestra en:

- Dentro de *General* (General):
  - Nombre
- Dentro de *Configuration* **(**Configuración**):**
  - *Bridge* (Puente). Escribir el nombre del puente previamente creado br0, en este caso.
- Dentro de *Address* **(**Dirección**):**
  - Primera dirección IP. Dirección donde comienza el rango (*pool*) que se va a crear.
  - Tamaño. Cantidad de direcciones IP que OpenNebula puede asignar a partir de la primera especificada.
- Dentro de *Context* **(**Contextualización**):**
  - Añadir el *Gateway* (puerta de enlace), DNS, máscara de red y dirección d red.

Una vez que se especifiquen todos estos cambios queda como se muestra en la Figura IV.3.

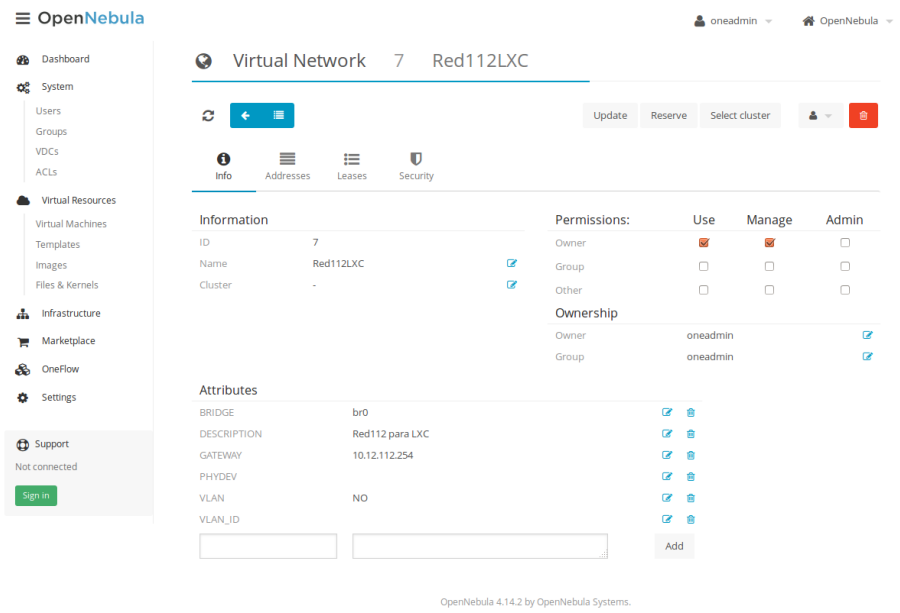

Figura IV.3 Red virtual LXC en Sunstone.

Paso 4: Crear la plantilla para el despliegue de los contenedores

Para crear una plantilla se deben especificar los campos siguientes:

- Dentro de General**:**
  - Nombre
  - Memoria
  - CPU
- Dentro de *Storage* (Almacenamiento)*:*
  - Seleccionar en el *Disk* 0 (Disco 0) la imagen LXC previamente subida a OpenNebula.
  - Es posible agregar varios discos LVM y sistemas de archivos. Estos discos ser montados dentro del contenedor en el directorio *"/media/DISK_ID*".
- Dentro de *Network* (Red):
  - Seleccione la interfaz de red o interfaces que desea adjuntarle al contenedor.
- Dentro de *Input/Output* (Entrada/Salida) (En caso de que VNC sea requerido)
  - Seleccionar VNC en la sección *Graphics*.
- Dentro de *Scheduling* (Planificación):

- Seleccionar el nodo o los nodos LXC en los que se quiera desplegar los contenedores para esta plantilla. Esto es requerido en entornos donde coexistan nodos de hipervisores diferentes a LXC. También es posible especificar que se despliegue en cualquier nodo que emplee como hipervisor LXC, para ello la expresión regular sería: *HYPERVISOR = \"lxc\"*.
- La opción anterior puede ser definida en la configuración de la plantilla mediante la directiva "*SCHED_REQUIREMENTS*". Ejemplo para emplear solo los nodos LXC:

  ```
  SCHED_REQUIREMENTS = "HYPERVISOR = \"lxc\""
  ```

  Mientras que para un nodo específico:

  ```
  SCHED_REQUIREMENTS="ID=\"5\""
  ```

Una vez creada la plantilla queda de la siguiente forma:

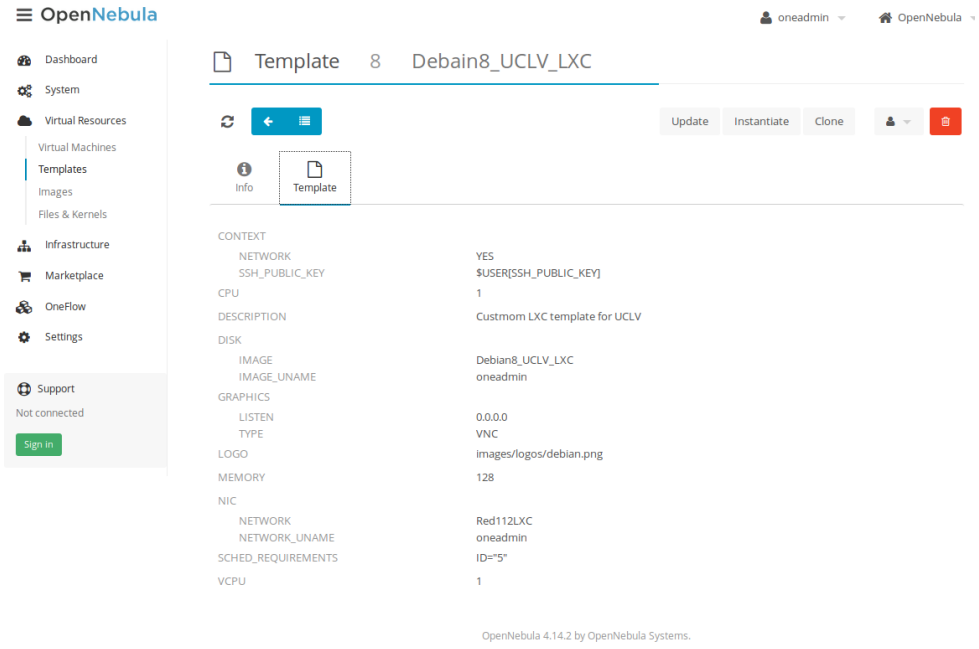

Figura IV.4 Plantilla para una VM LXC en Sunstone.

Desplegar el contenedor

Desde la interfaz Sunstone en ***Virtual Resources*** → ***Virtual Machines*** → ***ADD***, seleccionar la plantilla creada, luego se selecciona la VM y se da clic en el botón ***Create*** desde el menú.

Para que el contenedor inicie automáticamente (útil si se presentan fallas eléctricas), es necesario ajustar en su archivo de configuración "*/var/lib/lxc/<name>/config*" la siguiente línea como se muestra:

```
lxc.start.auto = 1
```