Proyecto Final de Máster en Software Libre

"Estado del arte en soluciones de virtualización. Sistemas gestores de Cloud: OpenNebula"

Administración de redes y sistemas operativos

Autor: José Antonio Montes Serena

Consultor: Jordi Massaguer Pla

Tutor externo: Antonio Rodil Garrido

Empresa colaboradora: ARTIC S.L.

Fecha: Diciembre de 2013

Licencia:

Resumen del proyecto:

En el presente proyecto se ha abordado la tarea de acercar las tecnologías existentes de plataformas de gestión de infraestructuras ofrecidas en la nube (**Cloud Management Platform**, aka **CMP** [1]) al mundo empresarial. En concreto, se ha desplegado una solución de explotación de infraestructuras privadas en la nube (**IaaS** [2]) enfocada a la gestión de un **datacenter** virtualizado, utilizando para ello soluciones completamente basadas en Software Libre.

De entre las opciones disponibles, se ha escogido **OpenNebula** [3], por ser una solución considerada como agnóstica de la tecnología utilizada, ampliamente adoptada por la comunidad científica, académica y empresarial, y con marcada orientación a la gestión de **datacenters** virtualizados, aunque su alta flexibilidad y versatilidad permiten darle otros usos diferentes dentro del ámbito de los **CMP**.

Para realizar el proyecto se ha realizado un estudio lo más extenso posible de la solución escogida, analizando todas las opciones tecnológicas ofrecidas por el entorno, para adoptar la configuración que mejor se adapta a nuestra solución con orientación empresarial.

A continuación, se ha planteado el diseño de la solución, de tal forma que sea escalable a nivel horizontal, y se encuentre dotada de redundancia frente a puntos únicos de fallo.

Para comprobar la viabilidad y validez de las opciones adoptadas, se ha instalado toda la solución en una maqueta, sobre la que se han realizado todas las pruebas necesarias, como paso previo a la etapa de instalación sobre el entorno de producción.

Una vez probadas todas las opciones sobre la maqueta, se ha procedido a la instalación de la solución definitiva sobre el entorno real de producción, utilizando el **hardware** disponible para ello.

A continuación se ha realizado la configuración y personalización de la plataforma, para poder realizar las diferentes pruebas que han permitido validar el entorno de producción.

Por último, se ha realizado la configuración necesaria para desplegar el sistema como entorno de producción, mediante la preparación de cuentas de usuario, recursos de red, imágenes, plantillas de configuración de instancias, etc.

Todos los pasos de instalación y configuración realizados durante el proyecto han quedado recogidos en dos archivos de comandos y salidas que se adjuntan como anexos aparte de esta memoria:

- El anexo 1 recoge todos los comandos y salidas ejecutados y comentados, realizados sobre la instalación de la maqueta.
- El anexo 2 recoge los comandos y salidas ejecutados y comentados, sobre las instalación del entorno de producción.

Índice de contenidos

Introducción

Hoy en día, las empresas que tienen su infraestructura **IT** alojada en sus propias dependencias, o en proveedores comerciales de alojamiento en Internet, se enfrentan con frecuencia a problemas de ahorro de costes, mantenimiento, y escalabilidad. Surge entonces la necesidad de acudir a soluciones de virtualización, que permitan compartir y gestionar los recursos físicos globales disponibles, optimizando y economizando sus prestaciones. Pero aún así, estas soluciones de virtualización per se, aplicadas a pequeña escala y de forma artesanal (trabajando sobre una imagen o máquina virtual en cada operación), no son lo suficientemente escalables ni ágiles como para desplegar y gestionar los servicios necesarios requeridos en algunos entornos, ni tampoco permite optimizar los recursos humanos empleados en la administración de los sistemas. Es en este contexto cuando aparecen los servicios de explotación de infraestructuras en la "nube" (**cloud**), llamada así porque trata de desligar la dependencia de la infraestructura física subyacente con los servicios virtualizados que ofrece. El caso más conocido de empresa que ofrece este tipo de servicios es **Amazon** con sus **Amazon Web Services** (**AWS**) [4], que marca la tendencia actual en el mercado.

Por otra parte, en el mundo del software libre aplicado a la nube, y en especial a la explotación de infraestructuras, también surgen varias iniciativas que tratan de cubrir el amplio abanico de posibilidades que ofrece el espectro [5], compitiendo con los grandes proveedores comerciales en cuanto a las funcionalidades ofrecidas.

Pero en el mundo empresarial, este tipo de tecnologías y soluciones sigue siendo desconocido y poco utilizado. Nuestra misión en el presente proyecto consistirá en acercar este tipo de soluciones **Open Source** a la pequeña y mediana empresa, describiendo y abordando un caso de aplicación real para implementarlo sobre un entorno en producción.

Objetivos

El objetivo del trabajo consistirá en realizar un análisis teórico y practico de implantación de una solución, lo más completa posible, de explotación de servicios **IaaS** (**Infrastructure as a Service** [2]) en la nube. Dicha implantación y estudio tendrá un enfoque enteramente empresarial, de tal forma que permita a la empresa donde se realizarán las prácticas, desplegar sus servicios de **infraestructura virtualizada** a través de la solución implantada, atendiendo tanto su demanda interna, como la de sus clientes de **hosting**.

Se utilizará **OpenNebula** [3] como la solución **Open Source** escogida para la implantación del **CMP** (**Cloud Management Platform**), debido a su alta flexibilidad y orientación para desplegar los servicios de **datacenter virtualizado** demandados por la empresa (véase el cuadrante mostrado en la primera referencia). Además, se pretende que la nube de carácter privado (con infraestructura local), pueda ser escalable, cuando la demanda requerida no pueda ser abastecida con los recursos locales, utilizando eventualmente los recursos públicos ofertados por **AWS** (**Amazon Web Services** [4]). De esta forma, la solución a implantar tendrá la categoría de nube híbrida, al mismo tiempo que se rentabilizarán al máximo los recursos físicos disponibles para la nube privada.

Con el fin de que la solución implantada sea totalmente explotable, se instalarán sistemas de monitorización, se trabajarán con diferentes aspectos en pos de la optimización y ajuste de la instalación en producción, como la preparación de las imágenes para las máquinas virtuales, la personalización de dichas instancias mediante plantillas y procedimientos de contextualización, optimización de los recursos de red para montar las redes virtuales que conecten entre si las instancias activas, etc.

Además se realizarán pruebas de rendimiento que demuestren la escalabilidad y validez de la plataforma como entorno de producción.

Por último, se realizarán comparativas de los resultados obtenidos por las pruebas de rendimiento, interpretando los resultados.

1. Estudio de viabilidad

1.1 Necesidades y requisitos del cliente

Withintic * Es una empresa del sector de las **TIC**, que se dedica al desarrollo de soluciones a medida basadas en **Software Libre**.

Withintic posee infraestructura propia en sus dependencias, donde mantiene sus actividades de desarrollo e implementación de soluciones a media.

No satisfecho con su linea de negocio actual, **Withintic** desea ampliar su catálogo de servicios ofreciendo servicios de **hosting** para sus clientes, aprovechando los recursos a nivel infraestructuras y conectividad de red actualmente disponibles en la sala técnica de sus dependencias.

Actualmente la empresa administra sus propios sistemas para atender a la demanda interna (los servidores corporativos que proporcionan los servicios internos de correo, **DNS**, **web**, servidores de desarrollo, etc.). Pero de cara a ofrecer los servicios de **hosting** a clientes, la operativa actual aplicada a la demanda interna no resulta válida para ofrecerla a sus clientes de forma eficiente.

La empresa necesita crear una solución de explotación de infraestructura que le permita desarrollar los servicios de **hosting** de forma ordenada, escalonada, estable, y al ritmo de la demanda de sus clientes, optimizando los recursos invertidos, y sin menoscabo de la calidad del servicio.

Es aquí donde aparece la necesidad de montar una solución completa de **datacenter**, utilizando las tecnologías de virtualización disponibles (aka el **datacenter** virtualizado), con el fin de atender la demanda de sus clientes, optimizando al máximo los recursos.

Con la idea de aprovechar el potencial de los recursos humanos de **IT**, y minimizar los costes iniciales y recurrentes de la solución, además de la posibilidad de personalizarla y adaptarla a las necesidades de la empresa, para distinguirse como elemento diferenciador de cara a sus competidores, se desea que la solución adoptada sea basada en Software Libre

* N.A: El nombre de la empresa es ficticio, para representar a una empresa genérica del sector de las TICs donde aplicar la solución.

1.2 Análisis de la situación actual

Withintic posee en sus dependencias una sala técnica completamente acondicionada para instalar la infraestructura necesaria de servidores y elementos de red, con los siguientes elementos:

- Sistema de alimentación continua de doble suministro (A y B).
- Sistema de aire acondicionado a través de falso suelo con pasillos de aire caliente y frio intercalados.
- **Racks** de **19”** disponibles, con doble suministro y regletas de alimentación.
- Bandejas de cableado estructurado: alimentación **AC**, fibras ópticas, y cableado de red categoría **5E** por pasos independientes.
- Repartidores ópticos (**ODF**) y **patch panels** categoría **5E**.
- Sistema de extinción de incendios por espuma seca, especial para equipos eléctricos.
- Control de acceso restringido exclusivamente para personal especializado.
- Monitorización de los equipos alojados en la sala, por parte del personal técnico 24x7.
- Monitorización de los sistemas de alimentación, aire acondicionado, y extinción de incendios, por parte del personal de seguridad del edificio, fuera del horario de oficina.
- Servidores industriales de alto rendimiento y eficiencia energética.
- Conectividad a internet con **BGP** a través de varios proveedores utilizando sus propios recursos asignados a través de **RIPE** (**AS** propio y direccionamiento **IPv4** asignado). La conectividad a internet se consigue mediante un par de **routers BGP** manteniendo una sesión **IBGP** sobre **OSPF** entre ambos, los cuales hablan a nivel **EBGP** con los proveedores mediante interfaces **gigabit ethernet** con fibra óptica monomodo, balanceando el ancho de banda proporcionalmente entre ambos **routers**.

Este pequeño **ISP** montado a medida es lo que le permite plantearse ofrecer, como ampliación de su catálogo de servicios, el **hosting** a sus clientes. Como el espacio es limitado, se plantea que la solución pueda ser escalable utilizando infraestructura externa, en la nube.

Como el servicio de **hosting** no está aún implantado, no existe una situación actual de partida sobre la que haya que migrar los servicios a la nueva solución, aparte de la infraestructura anteriormente descrita.

1.3 Definición de Requisitos del sistema

El sistema a implantar para ofrecer los servicios de **hosting** a los clientes deberá permitir los siguientes requisitos:

- La plataforma debe de ser abierta, y compatible con los estándares de referencia existentes en el mercado.
- La plataforma debe de ser redundante, y protegida frente a caídas de un único punto de fallo.
- Debe permitir la monitorización integral, tanto de los recursos de la plataforma, como de las máquinas (virtuales) alojadas de los clientes, para poder detectar cuellos de botella y adelantarse a problemas de escalabilidad.
- Debe de permitir la gestión y mantenimiento de las imágenes correspondientes a las máquinas alojadas de los clientes. Dichas imágenes (llamadas instancias), deben de estar centralizadas sobre un repositorio compartido.
- Debe de permitir la selección de imágenes con instalaciones ya preparadas, en función del sistema operativo y aplicaciones base deseadas, por parte de los administradores de la plataforma.
- Debe de permitir la personalización de dichas imágenes para poder adaptarlas a diferentes configuraciones según los requerimientos solicitados por el cliente (memoria, disco duro, **CPUs**, interfaces de red, esquema de conectividad, etc.).
- Debe tener diferentes niveles de acceso, en función del perfil del usuario, y los recursos de red (p.e. un administrador debería poder acceder a toda la plataforma, mientras que un supervisor, tendría derechos de monitorización, y por último un cliente final podría gestionar exclusivamente sus máquinas virtuales, para pararlas, reiniciarlas, acceder desde consola, etc.).
- Debe de proporcionar diferentes interfaces de acceso y monitorización.
- Debe de permitir el **accounting** de los recursos utilizados, para poder implantar la modalidad de pago por uso.
- Debe poder escalar usando infraestructuras públicas cuando la demanda de recursos en momentos pico así lo requieran (**cloudbursting** [6]). Preferiblemente sobre **AWS**.
- Debe de permitir la interconexión múltiple y distribuida a nivel de red entre las máquinas de los clientes, aplicando reglas y medidas de seguridad de bastionado de las máquinas de los clientes (es decir, aislamiento de las **LANs** de interconexión utilizadas para cada grupo de máquinas de los clientes).

- Debe permitir el redimensionado dinámico de los recursos de las instancias de los clientes.
- Debe de permitir las migraciones en vivo de las instancias de los clientes, desde una máquina física a otra diferente del mismo **cluster**.
- Debe permitir la integración con otros sistemas de provisión, monitorización, etc. utilizando **APIs** estándares y abiertas (**SNMP**, **AWS**, etc.).
- Debe de ser escalable, flexible en configuración, eficiente, y robusto.
- Debe soportar diferentes opciones de virtualización (**KVM** [7], **XEN** [8]), y permitir la virtualización de sistemas operativos e incluso arquitecturas diversas.
- Debe proporcionar servicios de apoyo dedicados y personalizados a grupos de instancias, como servidores **DHCP**, **NTP**, **NAT**, **routing** con salida a internet, etc. [9]

1.4 Estudio de alternativas de solución

Atendiendo al hecho importante de que la solución tiene que ser basada completamente en software libre, a continuación se enumeran las siguientes opciones de **CMP** (**Cloud Management Platforms**) existentes en el mercado:

1. **Eucalyptus** [10] es una plataforma orientada fundamentalmente al despliegue de provisión de infraestructura bajo demanda, de forma similar a como lo hace **Amazón Web Services** (aka **AWS**), pero contando con infraestructura local, en lugar de pública. Es altamente escalable, robusto, e implementa completamente el **API** de **Amazon**. Fue uno de los primeros sistemas de este tipo en aparecer en el mercado.
2. **OpenStack** [11] es una plataforma orientada a la provisión de infraestructura bajo demanda utilizando infraestructura local, aunque permite escalar a infraestructura pública a través de **RackSpace** [12]. Es una plataforma bastante robusta, altamente distribuida y escalable, pero su **API** es completamente incompatible con el **API** de **Amazon**.
3. **CloudStack** [13] es una plataforma a caballo entre la virtualización de **datacenters** y la provisión de infraestructura bajo demanda. Corre principalmente bajo **Java**, y ha sido cedida por **Citrix** [14] a la **Apache Foundation** [15], donde migró su licencia de **GPLv3** [16] a **ALv2** [17]. Su **API** también es compatible con el **API** de **Amazon**, y recientemente ha pasado a la categoría de **Top-Level Project** [18] en la **Apache Software**

Foundation, si bien le queda aún recorrido para conseguir la flexibilidad y madurez de las opciones anteriores.

4. **OpenNebula** [3] es una plataforma orientada a la virtualización de **datacenters** utilizando infraestructura local, aunque permite escalar sobre infraestructura pública a través del **API** de **Amazon**. Es altamente escalable, robusta, flexible, madura, y con desarrollo muy activo.

Todos los sistemas anteriormente descritos están basados en software libre, con lo que el coste de licencias por uso de las plataformas es algo que no se contempla.

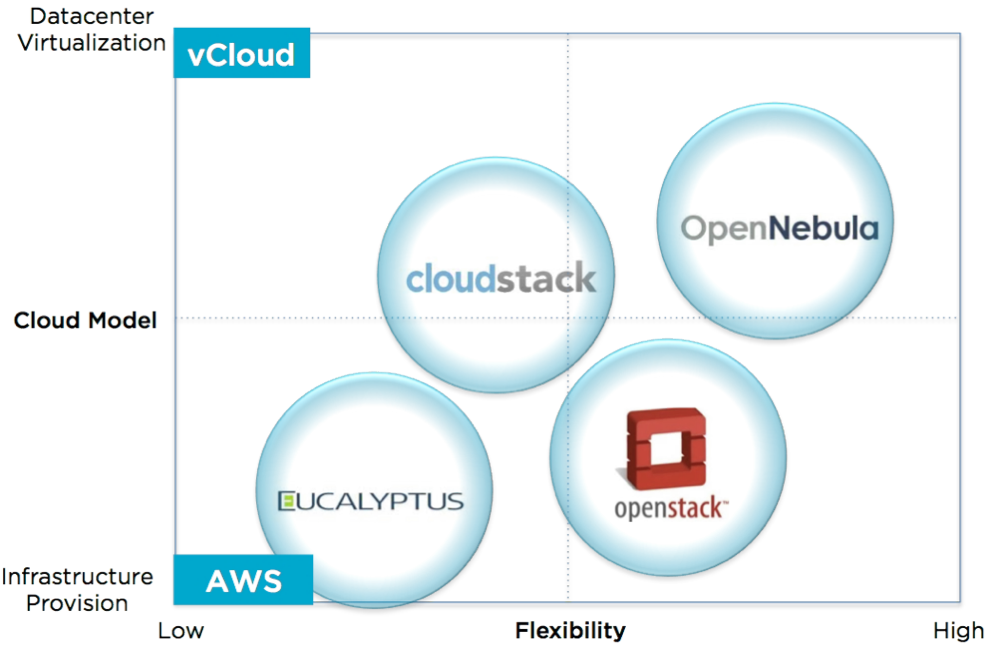
1.5 Valoración y elección de las posibles soluciones

Después de describir en el punto anterior las opciones alternativas existentes en el mercado como **Software Libre**, pasamos a valorar cada una de las opciones para escoger la solución más adecuada para nuestro proyecto:

- **Eucalyptus** es una solución con fuerte orientación a la provisión de infraestructura sobre equipamiento privado, a través del **API** de **Amazon** (de hecho se muestra como una clara alternativa a los servicios de **Amazon** sobre infraestructura local). Nuestro proyecto en particular requiere una solución de virtualización de **datacenter**, donde disponemos de las herramientas de administración necesarias para gestionar el **datacenter** junto con las instancias de los clientes. Por tanto esta solución no es la más idónea.
- **OpenStack** es una solución orientada principalmente a la provisión de infraestructura, con una clara vocación diferenciadora en cuanto a la implementación de su **API** de provisión, totalmente incompatible con **Amazon** (en eso se desmarca de las otras tres opciones), y compatible con **RackSpace**, permitiendo montar una estructura de nube híbrida sobre **RackSpace**, pero no sobre **Amazon**. Como uno de los requerimientos de nuestro cliente es que la solución adoptada permita escalar eventualmente sobre **Amazon**, esta opción no resulta tampoco válida.
- **CloudStack** sí soporta el **API** de **Amazon**, pero aún no es ni tan robusta ni tan flexible como las otras opciones, y su orientación no está demasiado orientada a la gestión de **datacenters** virtualizados, con lo que tampoco podemos tomarla como el mejor candidato.
- **OpenNebula** es la opción con orientación a la virtualización de **datacenters** más completa y flexible. Además soporta todos los requerimientos definidos en el apartado de requisitos, y permite

el escalado sobre la infraestructura de **Amazon**. Por otra parte dispone de una comunidad de desarrollo muy activa, el foro está bastante alimentado y soportado por la comunidad, la documentación es abundante y detallada, los paquetes de instalación se encuentran disponibles sobre los repositorios de las principales distribuciones de **GNU/Linux**, y certifica una serie de distribuciones y versiones sobre las que el sistema corre con garantías [19].

En el siguiente diagrama se puede observar un cuadro con las diferentes ubicaciones de cada solución anteriormente descritas, atendiendo a su grado de especialización y flexibilidad:



Fuente: http://blog.opennebula.org/?p=4042

Nosotros elegiremos **OpenNebula**, como la mejor opción para montar la solución requerida por **Withintic**, debido a que reúne todos los requisitos demandados, al mismo tiempo que garantiza una alta flexibilidad y escalabilidad, unida una alta estabilidad, en la solución del **datacenter** virtualizado para los servicios de **hosting**.

2. Análisis del sistema

2.1 Definición del sistema

Para desplegar la solución del **datacenter** virtualizado basado en **OpenNebula**, necesitaremos implantar los siguientes elementos:

1. Un repositorio de ficheros compartidos en red. Dicho repositorio se utilizará para contener las imágenes utilizadas para generar las instancias, las imágenes de las instancias generadas (esto nos permitirá realizar migraciones de instancias en vivo), y ficheros de diversa índole (plantillas, **kernels**, ficheros de configuración de aplicaciones, etc.), utilizados para la personalización de las instancias.
2. Un sistema centralizado de gestión (**frontend**), desde donde se administra todos los recursos de nuestro **datacenter** virtualizado. El **frontend** es donde se ejecutan los procesos de gestión y se conserva toda la información de administración de las instancias en el (los) **datacenter**(s). Además es donde se soporta el servidor web con el interfaz de administración. Aunque su tarea es fundamental, e incluso es capaz de controlar múltiples **datacenters** (soporta la opción de gestión **multi-tier**), en realidad no necesita una gran cantidad de recursos, con lo que en algunas instalaciones se configura directamente sobre el servidor a cargo del repositorio de ficheros, o incluso como una imagen virtualizada.
3. Varios nodos, sobre los que se ejecutan las instancias virtuales de los usuarios. Estos nodos pueden correr una o varias supervisoras de virtualización diferentes (**KVM**, **XEN**, **VMWare** [20]), y son gestionados desde el **frontend** a través de acceso remoto por red, utilizando el **API** estándar proporcionado por **libvirt** [21]. Los nodos son el principal “músculo” de nuestro **datacenter** virtualizado, e interesa que haya varios para dotar a la solución con las funcionalidades de redundancia y reparto de carga.
4. Un sistema externo de monitorización de la plataforma (opcional), para poder monitorizar los parámetros, tanto de los equipos físicos que forman el **datacenter**, como de las instancias virtualizadas de los usuarios. **OpenNebula** incorpora su propio sistema interno de monitorización, pero permite la integración con sistemas ampliamente utilizados de monitorización de **clusters** (como **Ganglia** [22]), por lo que conviene mencionarlo aquí.
5. Los nodos, el **frontend**, y el repositorio de ficheros, y el sistema de monitorización, deben de estar interconectados entre si mediante enlaces de datos de alta velocidad, redundancia, y

baja latencia. Esto se consigue mediante la instalación de **lan switches** de alto rendimiento (interfaces **gigabit**, o incluso de **10G**, ya soportados por algunos servidores), con técnicas de **bonding**, **LACP**, **802.1Q**, **QinQ**, etc., que permiten interconectar y aislar a nivel 2 las diferentes redes de los equipos físicos, así como de las instancias o grupos de instancias de los usuarios. En el caso de los nodos, se hace necesario establecer interconexiones dedicadas para transportar y aislar las redes de las instancias que comparten los mismos segmentos de red (**VLAN**), incluso encontrándose en máquinas físicas (nodos) dispersas, lo que garantiza la redundancia de los nodos, y las migraciones en vivo de las instancias entre nodos.

2.2 Requisitos exactos del proyecto

Los requisitos del proyecto se pueden desglosar atendiendo a las siguientes categorías:

- **Requisitos legales:** dado que el **datacenter** virtualizado albergará las instancias de los usuarios, conteniendo información personal y confidencial, así como las cuentas de acceso de todos los usuarios (tanto internos como externos) con acceso a la plataforma, se debe cumplir todas las normativas vigentes en protección de datos y acceso a la información de carácter personal (**LODP** [23]).
- **Requisitos de propiedad intelectual y licencias:** El proyecto debe basarse completamente en **Software Libre**, teniendo en cuenta las inversiones necesarias, tanto en la implantación inicial como en las posteriores actualizaciones de mantenimiento.
- **Requisitos de acceso único:** Todos los datos de gestión y acceso de la plataforma, así como las imágenes y las instancias de los usuarios, se almacenarán en un servidor de ficheros que hará la función de repositorio centralizado de la información. Sobre dicho servidor y la información contenida en él, deberán de aplicarse los permisos y medidas de acceso adecuados para cada usuario, con el fin de proteger los datos almacenados de accesos indebidos.
- **Requisitos de seguridad del sistema:** La plataforma deberá estar protegida frente a accesos externos no autorizados provenientes de internet, así como de posibles accesos físicos no autorizados a los servidores. Para ello se recurrirá a aplicar medidas de bastionado de los servidores físicos, así como de los elementos de red implicados en la solución (**lan switches**, **routers**, y **firewalls**).
- **Requisitos de gestión de backups:** la plataforma deberá contar con un sistema de **backups** automatizado y diversificado, para

evitar la pérdida de información de los usuarios, así como garantizar la integridad de los datos del sistema.

- **Requisitos de monitorización y supervisión de la plataforma:** la solución deberá incluir herramientas de monitorización y recogida centralizada de **logs** para garantizar el buen funcionamiento de la plataforma, así como la detección temprana de posibles problemas.

2.3 Establecimiento de requisitos

Desde el punto de vista de los **administradores**, estos son los requisitos a tener en cuenta, que complementan los mencionados en los puntos anteriores:

1. Posibilidad de administrar las imágenes con las distribuciones y aplicaciones base en un repositorio centralizado.
2. Posibilidad de establecer permisos y perfiles diferentes a cada usuario de acceso al sistema.
3. Posibilidad de contextualizar las imágenes dinámicamente con plantillas, atendiendo a cada tipo de cliente y entorno.
4. Posibilidad de realizar migraciones en vivo de las instancias entre los nodos.
5. Posibilidad de administrar cuotas de uso a cada instancia, y también a cada usuario, limitando el máximo de recursos a asignar.
6. Posibilidad de monitorizar, tanto los procesos y recursos consumidos por la plataforma, como los de las instancias o grupos de instancias pertenecientes a los usuarios.
7. Posibilidad de escalar horizontalmente la plataforma, sin impacto en el servicio.
8. Posibilidad de escalado vertical de la plataforma, utilizando recursos localizados en **AWS**, bajo condiciones de **cloudbursting**, de forma dinámica.
9. Posibilidad de reasignar recursos sobre las instancias en uso, sin impacto en el servicio.
10. Posibilidad de realizar **snapshots** [24] sobre las instancias activas de los usuarios.
11. Protección frente a fallos de la plataforma, como caída de uno de los nodos, etc.
12. Posibilidad de automatizar procesos, y expandir las posibilidades que ofrece la plataforma, mediante **APIs** disponibles para lenguajes de programación de **script** (**Ruby** [25], y/o **Python** [26]).

Desde el punto de vista de los **usuarios** con acceso a la plataforma, estos son los requisitos a tener en cuenta, que complementan los mencionados en los puntos anteriores:

1. Posibilidad de seleccionar las imágenes con las distribuciones y aplicaciones base disponibles en el repositorio centralizado.
2. Posibilidad de gestionar las instancias asociadas a su cuenta de usuario (crear, destruir, clonar, arrancar, parar, reiniciar, realizar **snapshots**, etc.).
3. Posibilidad de redimensionar los recursos de las instancias en ejecución, mientras se lo permita las cuotas de recursos asignadas a su cuenta.
4. Posibilidad de monitorizar en todo momento los recursos utilizados por sus instancias en ejecución.
5. Posibilidad de acceder a las instancias mediante consola.
6. Posibilidad de realizar todas las operaciones (incluido el acceso por consola) a través de una interfaz gráfica amigable, preferiblemente un interfaz web.
7. Tener garantías de la integridad y protección de los datos albergados sobre la plataforma.

2.4 Definición de interfaces de usuario

Los interfaces de usuario disponibles son los proporcionados a través del **frontend**, y se pueden dividir en tres partes principalmente:

1. El interfaz de administración gráfico a través de la web, o **WebGUI**. Por medio de este interfaz, podemos gestionar íntegramente toda la plataforma, pudiendo realizar las tareas de provisión, mantenimiento, creación de nuevas instancias, monitorización de los recursos del sistema, e incluso acceso (por **websocket** [27]) a la consola de las instancias en ejecución. Es uno de los puntos fuertes más atractivos de la solución, y además permite la creación de varios usuarios con perfiles y privilegios diferentes, por lo que incluso se podría dar acceso a los clientes finales con ciertas garantías de seguridad, para que gestionen sus recursos, asignados previamente por nosotros a través de cuotas.
2. El interfaz de administración por línea de comandos. A través de la línea de comandos, con el usuarios de administración de la plataforma desde el **frontend**, se pueden realizar a bajo nivel todas las opciones de mantenimiento y administración de la plataforma, como la creación de nuevos nodos, redes, plantillas, etc. así como todas las opciones soportadas por el interfaz **WebGUI**.

3. Varios interfaces **API** para gestión y provisión externa. Tales como los interfaces **API** de **SNMP** [28], **EC2** [29], etc. Nota: aquí no especificamos los interfaces **API** de despliegue de servicios sobre infraestructura pública, como **EC2** (**AWS**) y **OCCI** [30] (estándar abierto).

2.5 Especificación del plan de pruebas

En nuestro proyecto, abordaremos la puesta en servicio de una plataforma desde cero, por lo que deberemos realizar los siguientes tipos de pruebas:

- **Pruebas unitarias:** Se realizarán pruebas sobre cada uno de los elementos que componen la plataforma, verificando su correcto funcionamiento:
    1. Acceso remoto automatizado a todas las máquinas y desde todas las máquinas que integran la plataforma desde el usuario de administración de **OpenNebula**.
    2. Acceso desde todos los servidores (**frontend** y nodos) al servidor de ficheros centralizado con los usuarios y permisos correctos.
    3. Pruebas de conectividad y aislamiento de cada una de las redes y **VLANs** utilizadas por la plataforma.
    4. Pruebas de acceso y gestión remota desde el **frontend** hacia los nodos.
    5. Pruebas de monitorización de los servidores y procesos de la plataforma.
    6. Pruebas de funcionamiento de los interfaces de usuario.
- **Pruebas de sistema:** se comprobará que los diferentes componentes y procesos que integran la plataforma interactuan entre si de forma correcta.
- **Pruebas de integración:** se realizarán pruebas con casos de uso sobre la plataforma en su conjunto, que incluirán la gestión de imágenes, contextualización de estas, pruebas de conectividad de red entre las instancias, migración de instancias de un nodo a otro, etc.
- **Pruebas de escalabilidad:** se probará la plataforma bajo condiciones de estrés para medir su rendimiento y ponderar su escalabilidad.
- **Pruebas de aceptación:** Se realizarán pruebas de aceptación por parte de los administradores y usuarios del sistema, para confirmar que efectivamente la plataforma cumple con los requerimientos

recogidos en los apartados anteriores, y queda dispuesta para su uso en producción.

Además, la propia solución (**OpenNebula**) ha pasado rigurosos controles de **QA** (**Quality Assurance**) [31], mediante sistemas de integración continua (**Jenkins** [32]), y políticas de certificación e interoperabilidad con los diferentes elementos que componen la plataforma, desde los sistemas operativos hasta las **hipervisoras** soportadas, para garantizar la viabilidad y robustez de la solución. Para conocer más información sobre este tema se puede consultar la sección de **QA** en la página web de **OpenNebula** [33].

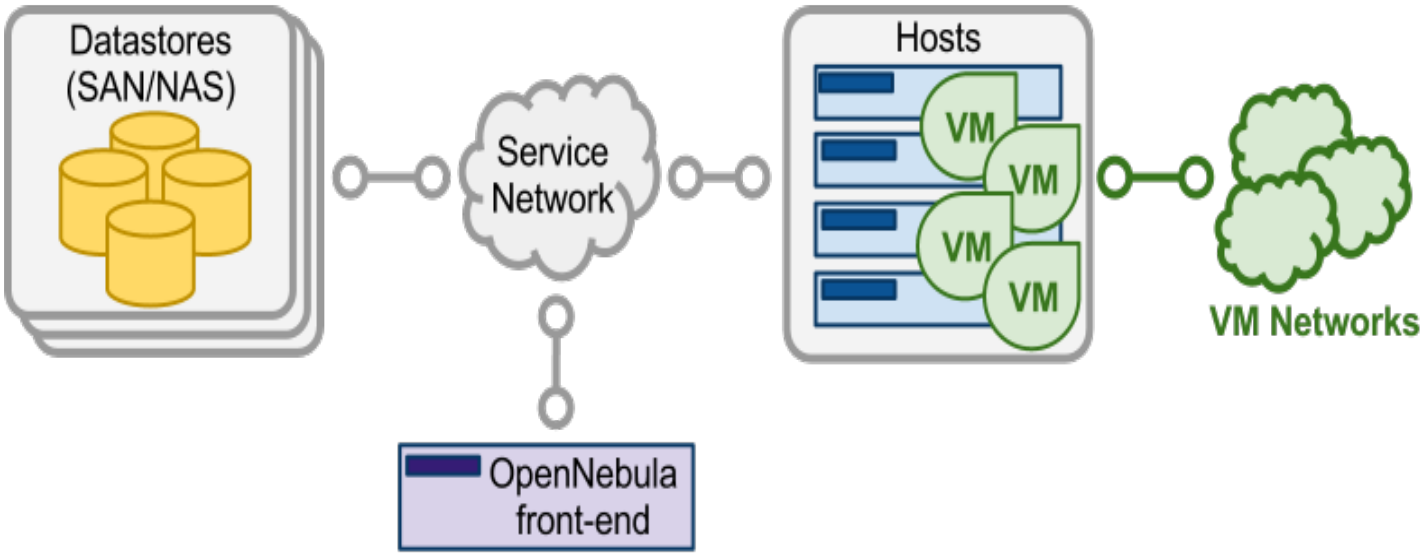
3. Diseño de la solución

3.1. Arquitectura del sistema

En los siguientes puntos realizaremos un repaso pormenorizado por la arquitectura de **OpenNebula**, que será la solución a implantar para nuestro **datacenter** virtualizado.

3.1.1. Arquitectura funcional

En la siguiente imagen se describen los bloques funcionales que representarán los elementos correspondientes a la estructura física de nuestra plataforma:



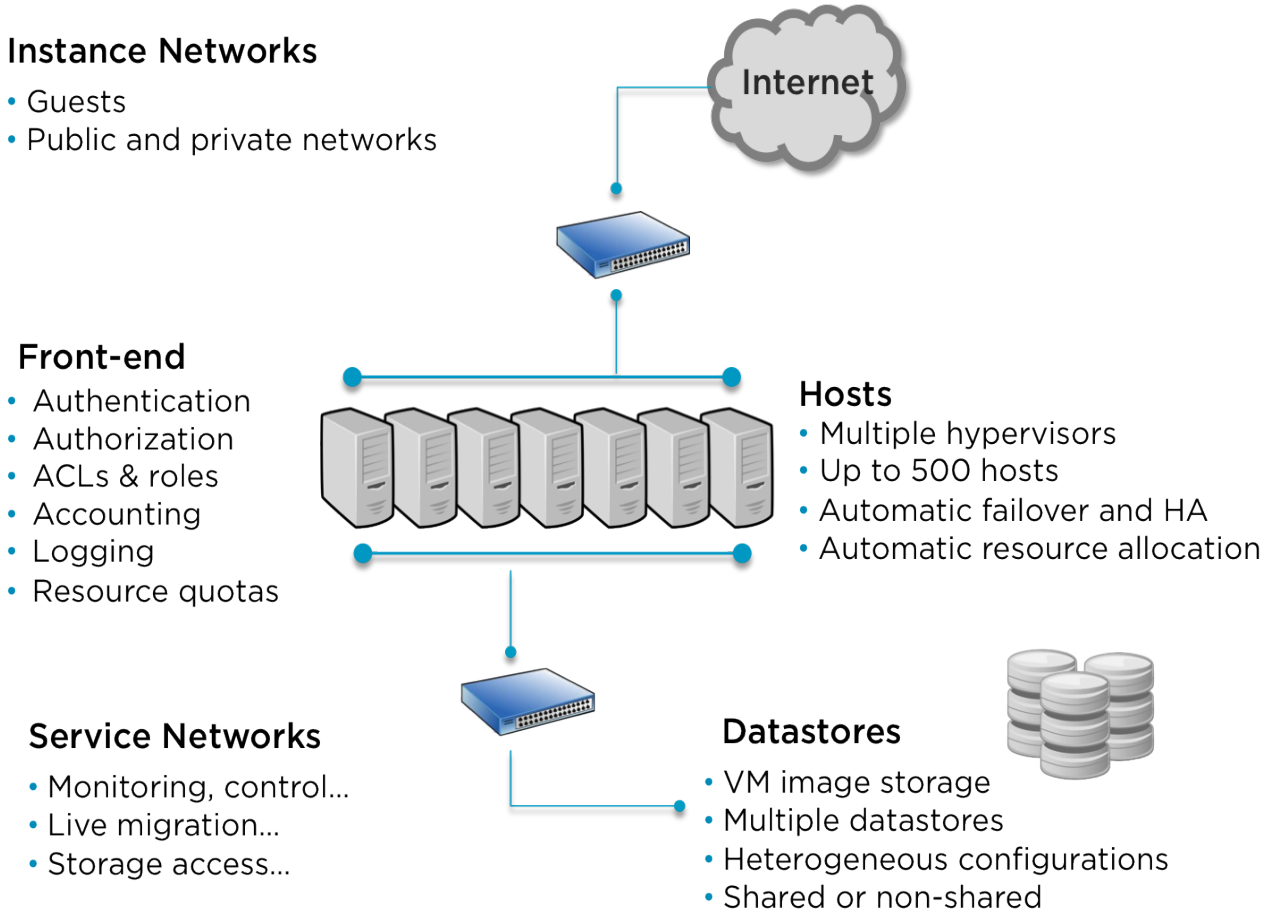
Fuente: http://opennebula.org/_detail/documentation:rel3.4:one_high.png

Los componentes básicos del sistema **OpenNebula**, según la imagen anterior son los siguientes:

- El **Front-end**: es el servidor donde se ejecutan todos los servicios de gestión, provisión, monitorización, acceso, y administración de la plataforma. Es el cerebro del sistema.
- Los **Hosts**: son los servidores de virtualización donde se ejecutarán las máquinas virtuales (**VM**) proporcionadas por la plataforma.
- Los **Datastores**: son los repositorios donde se almacenan las imágenes de las máquinas virtuales. Los **datastores** pueden contener imágenes en varios formatos, así como ficheros de configuración, **kernels**, volúmenes de almacenamiento, etc.

- El **Service Network**: es la red de servicio que permite interconectar los **Hosts** con los **Datastores** y el **Frontend**. Todos los nodos que integran el sistema deben de poder interconectarse directamente entre sí a través del **Service Network**.
- Las **VM Networks**: es la infraestructura física de red que permite soportar los servicios de red dedicados (**VLANs**) entre las máquinas virtuales (**VM**).

Otra forma de ver la estructura anterior es mediante el siguiente esquema funcional, sin atender a su estructura física:

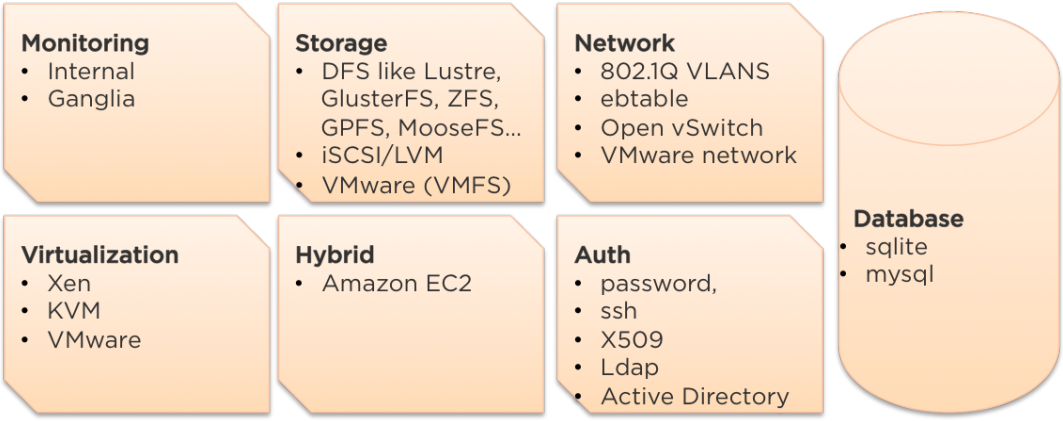


Fuente: http://opennebula.org/_media/overview_operators.png

3.2. Descripción de los subsistemas

La plataforma puede descomponerse a nivel funcional en varios subsistemas, cada uno de ellos desplegando una serie de servicios, necesarios para la administración y funcionamiento de la solución. Estos subsistemas, a su vez soportan varias opciones o tecnologías de uso, las cuales explicaremos en los siguientes apartados para poder decidir la mejor opción para nuestro caso de aplicación.

En la siguiente imagen podemos apreciarlo gráficamente:



Fuente: http://opennebula.org/_media/overview_builders.png

3.2.1 Subsistema de virtualización

El subsistema de virtualización, es el componente encargado de comunicarse con las hipervisoras (**hypervisor** [34]) instaladas en los **Hosts** de virtualización, y de gestionar las acciones requeridas durante el ciclo de vida de las máquinas virtuales (**VM**).

OpenNebula soporta actualmente las siguientes hipervisoras:

1. **Xen** [8]: esta hipervisora soporta servicios de paravirtualización [35] y virtualización completa [36]. Es muy potente, segura, y eficiente, pero requiere que tanto el **host** como las máquinas virtuales, tengan un **kernel** especialmente adaptado para poder ejecutarse.
2. **KVM** [7]: esta hipervisora está soportada de forma nativa por el **kernel** de **Linux**, y permite ejecutar máquinas virtuales con virtualización completa para diferentes sistemas operativos, sin que sea necesario realizar ningún tipo de adaptación en el huesped. Para ello necesita que las **CPUs** del **host** soporten la extensión de virtualización **hardware** (llamada **VT-x** por **Intel** y **V** por **AMD**) [37], y que dicha funcionalidad se encuentre habilitada en la **BIOS** del servidor, lo que no siempre es posible. Con el fin de acelerar el acceso de entrada y salida de los periféricos en

las máquinas virtuales, **KVM** también soporta la paravirtualización de ciertos dispositivos (red, disco, memoria, y tarjeta gráfica) mediante los controladores **VirtIO** [38] correspondientes, soportados de base en los sistemas operativos modernos, incluido **Windows** [39]. El tipo de virtualización resultante se llama **híbrida** [40].

3. **VMware** [20]: es un sistema de virtualización propietario, cuya hipervisora es capar de ejecutarse directamente sobre el hardware del **host** sin necesidad de instalación del sistema operativo. Su gestión está soportada por **OpenNebula**, por medio de una pasarela especial [41], para las hipervisoras **VMware ESX** y **VMware Server**.

Para nuestra solución, nosotros optaremos por utilizar **KVM**, como solución de hipervisora instalada en los **hosts**. Los criterios para tomar esta decisión son:

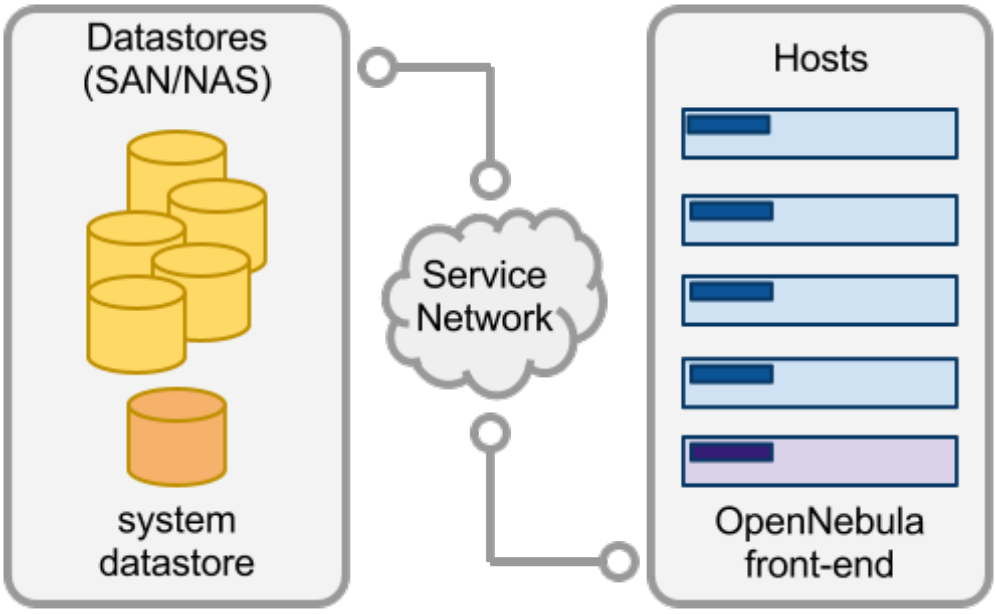
- La extensión (también llamada aceleración) de virtualización por **hardware**, está ampliamente soportada por los procesadores y las **BIOS** en los servidores actuales.
- Los sistemas operativos actuales que pueden correr como huéspedes en los servidores de virtualización, incorporan de base los controladores **VirtIO** para los dispositivos paravirtualizables soportados por **KVM**, con lo que se consigue la virtualización híbrida, con un rendimiento de la máquina virtual similar al que tendría de ejecutarse directamente sobre la máquina física.
- No es necesario adaptar los sistemas operativos huésped para que puedan ejecutarse sobre las hipervisoras **KVM**. Permite la ejecución de sistemas operativos diferentes al del **host**, y además, mediante **QEMU** [42], permite la ejecución en modo emulación de sistemas con arquitecturas diferentes al **host** (como el caso de dispositivos **Android**), o virtualización anidada (la máquina virtual es capaz a su vez de ejecutar instancias virtualizadas mediante **KVM**).
- **KVM** permite realizar tareas de gestión muy potentes, como migración de imágenes en vivo de un **host** a otro, captura de **snapshots** de las imágenes en ejecución, reglas de filtrado de red entre las máquinas virtuales, etc.

OpenNebula utiliza el **API** con la librería **libvirt** [21] instalada en los **hosts**, para poder comunicarse y gestionar las hipervisoras. Para ello utiliza dos módulos diferentes:

- El **VM_MAD**: es el módulo de administración de la hipervisora, y es el responsable de gestionar las máquinas virtuales en los hosts.
- El **IM_MAD**: es el módulo de gestión de información de la hipervisora, y es el responsable de recoger y procesar la información sobre el estado de las máquinas virtuales, reportadas por las hipervisoras instaladas en los **hosts**.

3.2.2 Subsistema de almacenamiento

El subsistema de almacenamiento, mediante sus diferentes tipos de **datastores**, cumple la función de repositorio centralizado de las imágenes y ficheros utilizadas por las máquinas virtuales en su ejecución. Especial mención merece el **System Datastore**, utilizado por el la plataforma para contener las imágenes de las máquinas virtuales en ejecución. Los **datastores** se comunican con los nodos (los **hosts** y el **front-end**) a través de la red de servicio (**service network**), como se puede apreciar en la siguiente imagen:



Fuente: http://opennebula.org/_detail/documentation:rel3.4:datastoreoverview.png

OpenNebula soporta los siguientes tipos de **datastores**, coexistiendo varios tipos simultáneamente sobre la misma plataforma:

- **System**: como hemos comentado anteriormente, es el encargado de contener las imágenes de las máquinas virtuales en ejecución. A su vez se soportan varios formatos para dichas imágenes (copias completas de la imagen original, deltas de ficheros **qcow** [43], o sistemas de ficheros en formato **raw** [44]).
- **File-System**: las imágenes son almacenadas en ficheros tipo **raw** y proporcionadas por un servidor **SAN/NAS** [45] a los **hosts**.
- **iSCSI/LVM**: las imágenes se almacenan en un servidor como volúmenes lógicos y se presentan como **targets iSCSI** [46] a los **hosts**.
- **Vmfs** [47]: es el sistema de almacenamiento utilizado por las hipervisoras **VMware**. Este **datastore** no puede ser montado en el servidor **front-end**, ya que no es compatible con **Linux**.
- **Ceph**: es utilizado para almacenar imágenes utilizando el sistema de almacenamiento distribuido **Ceph** [48].
- **Files**: es un **datastore** especial para almacenar ficheros, en lugar de imágenes, como **kernels**, **ramdisks** [49], o ficheros de contextualización, empleados por las imágenes.

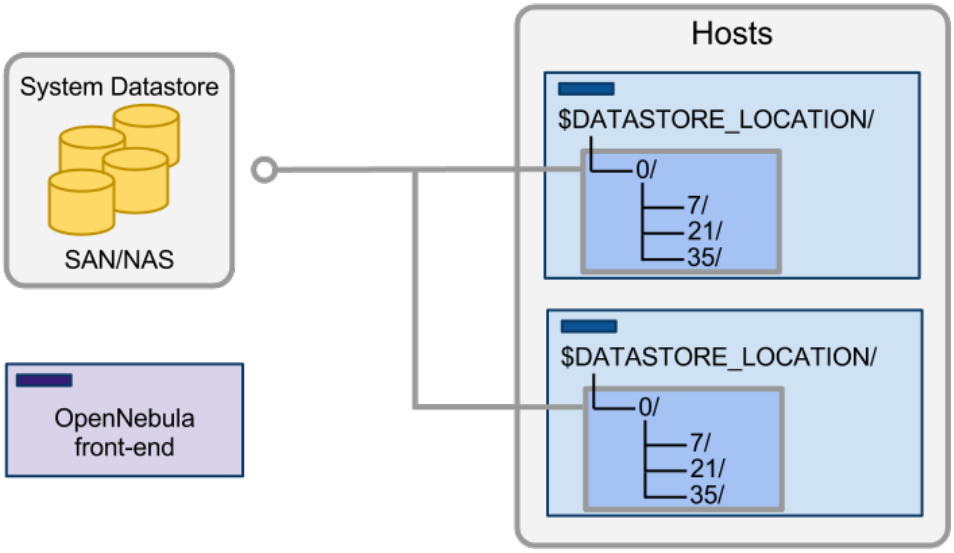
Por otra parte, las imágenes son transferidas desde los **datastores** a los **hosts** mediante los controladores de gestión de transferencia (**Transfer Manager**, o **TM**), los cuales realizan operaciones a bajo nivel específicas para cada **datastore**, que deben de ser debidamente configuradas en cada **host**. Existen varias formas de transferir las imágenes de los **datastores** a los **hosts**:

- **Shared**: el **datastore** está exportado en un sistema de ficheros compartido en la red de servicio (por ejemplo, mediante **NFS** [50]).
- **SSH**: las imágenes son copiadas desde el **datastore** a los **hosts** utilizando el protocolo **SSH** [51]. Este sistema ofrece muy bajo rendimiento y añade sobrecarga a los **hosts**, debido a que la información viaja encriptada y tunelizada a través de **SSH**.
- **ISCSI**: los **hosts** aceptan los **targets** de los **datastores** abriendo y cerrando sesiones dinámicamente.
- **Vmfs** [52]: las imágenes son copiadas mediante el uso de **vmkfstools** (las herramientas de gestión de sistemas de ficheros de **VMware**).

- **Qcow**: es un controlador específico para manejar el formato de imágen **qcow** [43] y aprovechar las posibilidades de **snapshot** que ofrece ese formato de fichero.
- **Ceph**: es el controlador especial para gestionar la **RDBs** de **Ceph** en los nodos distribuidos.

Para nuestra solución, nosotros utilizaremos el sistema **shared** para el **system datastore**, ya que ello reduce el tiempo de despliegue de las máquinas virtuales, al mismo tiempo que permite las migraciones en vivo de las máquinas virtuales entre los **hosts**. El sistema de compartición de ficheros que utilizaremos será **NFS**, por su alto rendimiento y estabilidad, además de ser un sistema ampliamente conocido, lo que acorta el tiempo de aprendizaje por parte de los administradores de la plataforma.

El sistema de almacenamiento queda representado por la siguiente imagen:



Fuente: http://opennebula.org/_detail/documentation:rel3.4:shared_system.png

3.2.3 Subsistema de red

El subsistema de red permite de una forma escalable y segura que las máquinas virtuales distribuidas en el **datacenter** puedan interconectarse entre sí a través de sus interfaces de red, por medio de segmentos de red (**VLANs**) independientes, proporcionando al mismo tiempo su conexión con el exterior (cuando ello se estime necesario), mediante diversos mecanismos o controladores. Esto permite que cada máquina virtual pueda tener acceso a varias redes, públicas o privadas, definidas mediante plantillas (**templates**) de red.

El administrador de la plataforma deberá de tener en cuenta que si bien esto supone una funcionalidad muy potente, deberá complementarlo con mecanismos de seguridad que restrinjan el acceso a la red exclusivamente para las máquinas virtuales deseadas, impidiendo el acceso al resto de máquinas virtuales del sistema, y viceversa.

Para ello **OpenNebula** dispone de diferentes controladores de redes virtuales, que pueden ser asociados cuando el **host** (i.e. el servidor de virtualización) es definido en la plataforma:

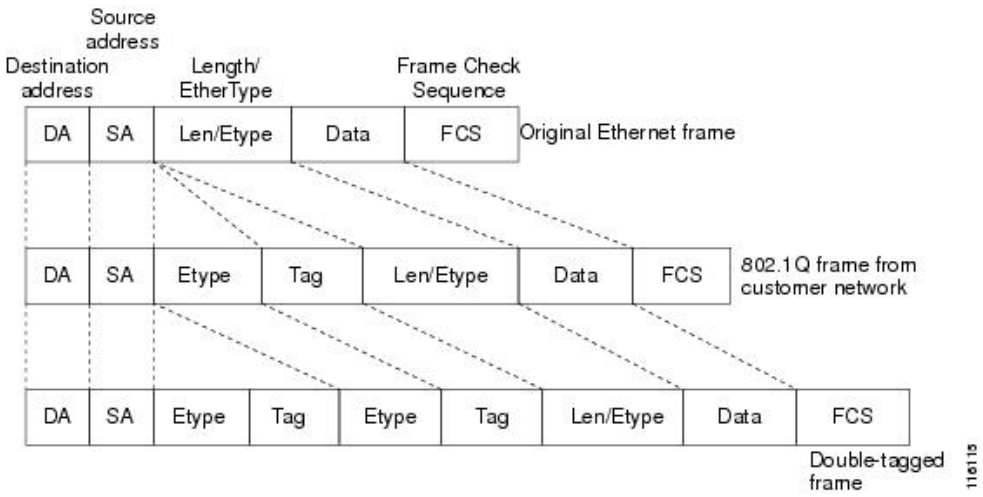
- **Dummy**: Es el controlador de red por defecto, el cual no realiza ninguna operación real en el sistema cuando la red es definida, ni siquiera aplica reglas de **firewall** [53] a los interfaces definidos cuando la máquina virtual es instanciada con sus interfaces de red.
- **FW**: este controlador permite aplicar reglas de **firewall** sobre los interfaces de red, cuando estos son creados en la máquina virtual instanciada, pero no aplica ninguna regla de aislamiento de red a nivel 2, por lo que no impide ataques realizados a nivel 2 (como tormentas de **broadcast** [54], **MAC spoofing** [55], etc.).
- **802.1Q**: Este controlador permite el transporte y encapsulación por un mismo puerto físico de red de hasta **4096 VLANs** [56], mediante el etiquetado de **VLANs** por el protocolo **802.1Q** [57], con lo que cada **VLAN** queda completamente aislada del resto a nivel 2, aunque para ello sea necesario contar con **lan switches** que soporten dicha funcionalidad, y que sean configurados adecuadamente para permitir interconectar los **hosts** entre sí (mediante *troncales* o puertos en modo **trunk**), sin afectar al resto de la red.
- **Ebtables**: Este controlador permite el aislamiento de **VLANs** mediante el uso de reglas **ebtables** [58] (similar a las **iptables** [59] pero trabajando a nivel 2 con **MACs**). Para este controlador no es necesario una configuración hardware especial, aunque tiene el problema de que no aísla las **VLANs** a nivel **IP**, con lo que habría que aplicar reglas de filtrado **IP** adicionales sobre las máquinas virtuales.
- **OvSwitch**: Este controlador permite el aislamiento y transporte a nivel 2 de **VLANs** mediante la solución **Open vSwitch** [60], el cual permite aplicar reglas de filtrado, **QoS**, etc. Es una solución muy completa y potente, aunque su instalación y puesta a punto puede resultar un poco compleja y delicada.
- **VMware**: es el controlador especial para poder utilizar la infraestructura de red de **VMware** con el fin de proporcionar una red aislada **802.1Q** compatible con las máquinas virtuales instanciadas en las hipervisoras **VMware**.

La siguiente tabla muestra la matriz de compatibilidades entre las hipervisoras y los controladores de red soportados por la plataforma:

	Firewall	Open vSwitch	802.1Q	ebtables	VMware
KVM	Yes	Yes	Yes	Yes	No
Xen	Yes	Yes	Yes	Yes	No
VMware	No	No	No	No	Yes

Fuente: http://opennebula.org/documentation:rel4.2:nm

En nuestra solución aplicaremos las técnicas de transporte y encapsulado de **VLANs** por **802.1Q**, al ser una tecnología estándar, madura, muy probada, con varios años de experiencia de uso por parte del alumno en su labor profesional diaria, y totalmente soportada por el **kernel** de **Linux**. En el caso de que los **hosts** se encuentren separados físicamente entre sí, y sea necesario interconectar las redes mediante **802.1Q** utilizando **lan switches** compartiendo un enlace de transporte común, se requerirán técnicas especiales de doble etiquetado de **VLAN**, también llamada **QinQ**, por medio del protocolo **802.1ad** [61], donde a través de un mismo **trunk** compartido, se pueden transportar hasta **4096 trunks** agregados. Para ello la trama **ethernet** lleva dos etiquetas diferentes (llamadas **tags**): la **S-tag (Service Tag)**, que es la etiqueta exterior, y la **C-tag** (**Customer Tag**) que es la etiqueta interior. Un esquema de la trama resultante podemos verlo en la siguiente ilustración:



Fuente: http://www.cisco.com/en/US/i/100001-200000/110001-120000/116001-117000/116115.jpg

Para poder utilizar **802.1Q** en los **lan switches**, la mayoría de los equipos de gama media (con cierta capacidad de procesamiento de conmutación de paquetes) ya lo soportan (hay que tener en cuenta que trabajaremos con varios interfaces de red agregados a nivel físico para conseguir cierto nivel de escalabilidad en la plataforma, lo que implica un equipamiento **hardware** potente). Sin embargo los **lan switches** con capacidades de **802.1Q-in-Q** ya son algo más caros, y son utilizados a nivel de **ISPs** para proporcionar servicios de transporte transparente de nivel 2 por **VLAN** punto a punto a clientes, que a su vez, utilizan la encapsulación **802.1Q** en dichos circuitos para optimizar el transporte de **VLANs** punto a punto facilitado por el proveedor.

3.2.4 Subsistema de monitorización

OpenNebula soporta la monitorización de los **hosts** y máquinas virtuales a través de su controlador **IM** (**Information Manager**), configurado para trabajar con la hipervisora en cada **host** definido en la plataforma. Pero por otra parte, **OpenNebula** admite también su integración con otros sistemas de monitorización ampliamente utilizados en administración de **clusters**, como **Ganglia** [62]. Para ello habría que definir el **IM** de los **hosts** para que reporten los datos recogidos al nodo de gestión de **Ganglia**, utilizando el controlador especial proporcionado por **OpenNebula**.

En nuestra solución, nosotros utilizaremos **Ganglia** como sistema de monitorización global de la infraestructura que compone la solución (aka el cluster de virtualización), mientras que usaremos el sistema de monitorización nativo de **OpenNebula** para recoger y presentar las estadísticas de uso de los recursos utilizados por cada usuario, gracias a su funcionalidad integrada de mostrar sólo la información correspondiente a cada cuenta de usuario mediante vistas personalizadas.

3.2.5 Subsistema de autenticación

OpenNebula viene por defecto dotado con un sistema interno de autenticación mediante **usuario/password** junto con un sistema de autorización basados en listas de accesos por permisos (**ACL** [63]). Por otra parte, apelando a su alta flexibilidad, también soporta varios sistemas de autenticación externos, aplicados sobre los diferentes módulos de acceso al sistema.

Nosotros haremos uso del sistema interno de autenticación, soportado por defecto, ya que es lo suficientemente completo como para proporcionarnos las funcionalidades requeridas de autenticación autorización, y **ACLs**.

3.2.6 Subsistema de base de datos

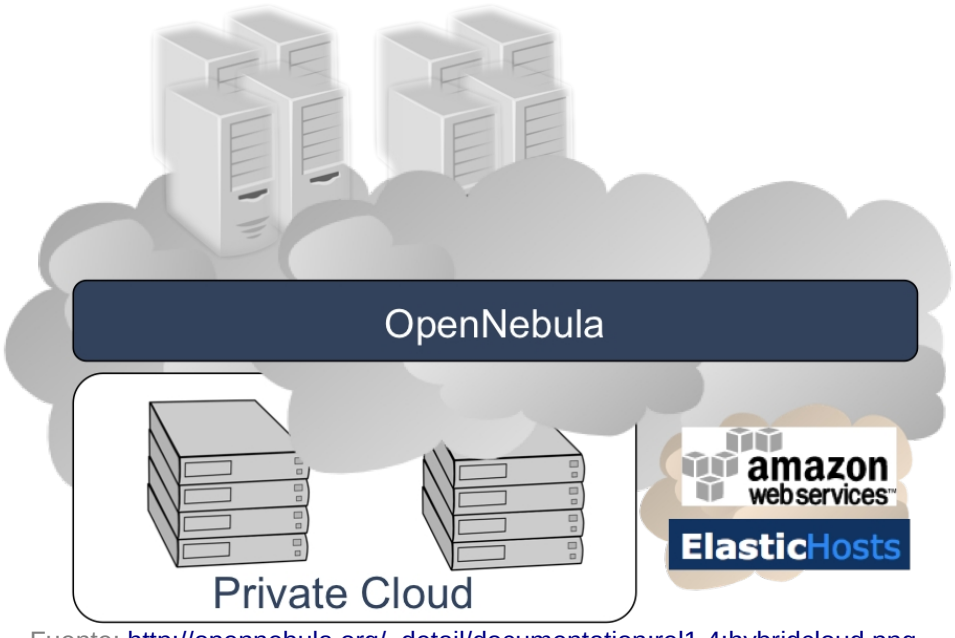
OpenNebula soporta dos sistemas de bases de datos diferentes, para poder almacenar toda la información de gestión de la plataforma de forma centralizada:

- **SQLite** [64]: es la base de datos soportada por defecto.
- **MySQL** [65]: pensada para entornos de alta disponibilidad del **Frontend**, o con conexión remota distribuida.

En nuestra solución, utilizaremos **SQLite** como motor de base de datos, ya que es el que viene soportado por defecto, y para los requerimientos de la plataforma es más que suficiente.

3.2.7 Subsistema de escalado híbrido

OpenNebula proporciona sistemas de escalado híbrido, que permiten que los recursos asignados por la infraestructura local (privada) del **datacenter**, en condiciones pico de carga, pueda llegar a escalar sobre infraestructura pública de red, a través de un **API** estándar (en este caso el **API** de **Amazon EC2** [66]), permitiendo de esta forma establecer la arquitectura de una nube híbrida, como la que se muestra en el siguiente dibujo:



Fuente: http://opennebula.org/_detail/documentation:rel1.4:hybridcloud.png

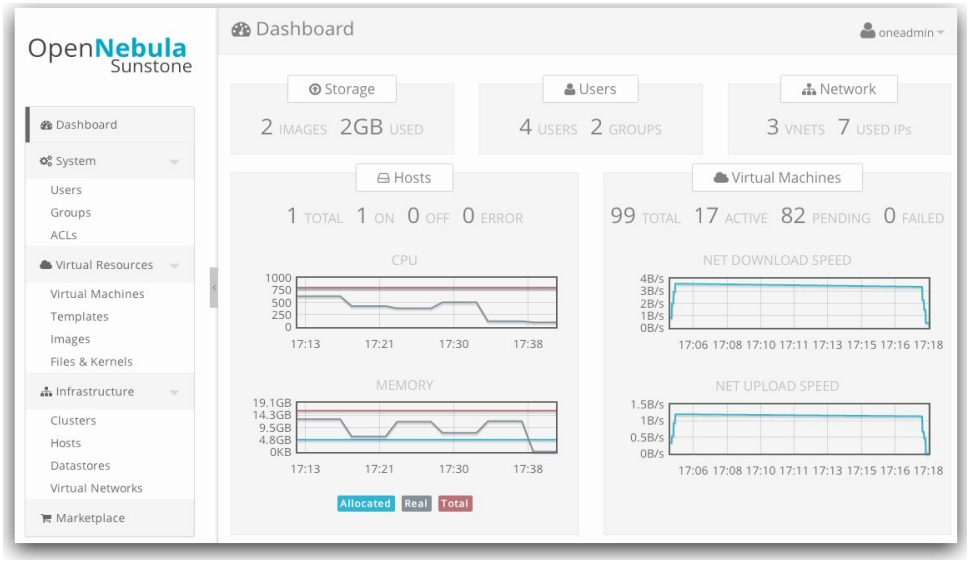
3.2.8 Subsistema de administración de la plataforma

OpenNebula soporta varias opciones de gestión de los recursos de la plataforma, disponiendo incluso de **APIs** estándares (**EC2**, **OCCI**) para proporcionar servicios **IaaS** públicos a terceros (tal y como hacen **Amazon**, **Rackspace**) a la manera de **Eucalyptus** (i.e. utilizando la infraestructura privada local). Como nuestro proyecto aborda el despliegue y administración de un **datacenter** virtualizado, trataremos los dos modos de administración principales soportados por la plataforma:

- **CLI** [67]: Este método (línea de comando) permite proporcionar al administrador de la plataforma un control total para desplegar, configurar, monitorizar y gestionar todos los aspectos soportados por el sistema. **OpenNebula** proporciona un juego de herramientas de línea de comando bastante completo y potente, por lo que será el método de administración a bajo nivel preferido por el administrador, dada su versatilidad y potencia.
- **Sunstone** [68]: Es el interfaz gráfico de gestión vía **web** de la plataforma, permitiendo en la práctica casi todas las opciones posibles de gestión de la plataforma, y la cual soporta varias vistas, según el perfil del usuario sea el de administrador, administrador de **datacenter** virtualizado, o usuario. Las vistas pueden ser completamente adaptadas mediante plantillas de configuración, y asignadas a usuarios o grupos de usuarios, de tal forma que tengamos un control total de las funcionalidades que pretendamos proporcionar a cada tipo de perfil.

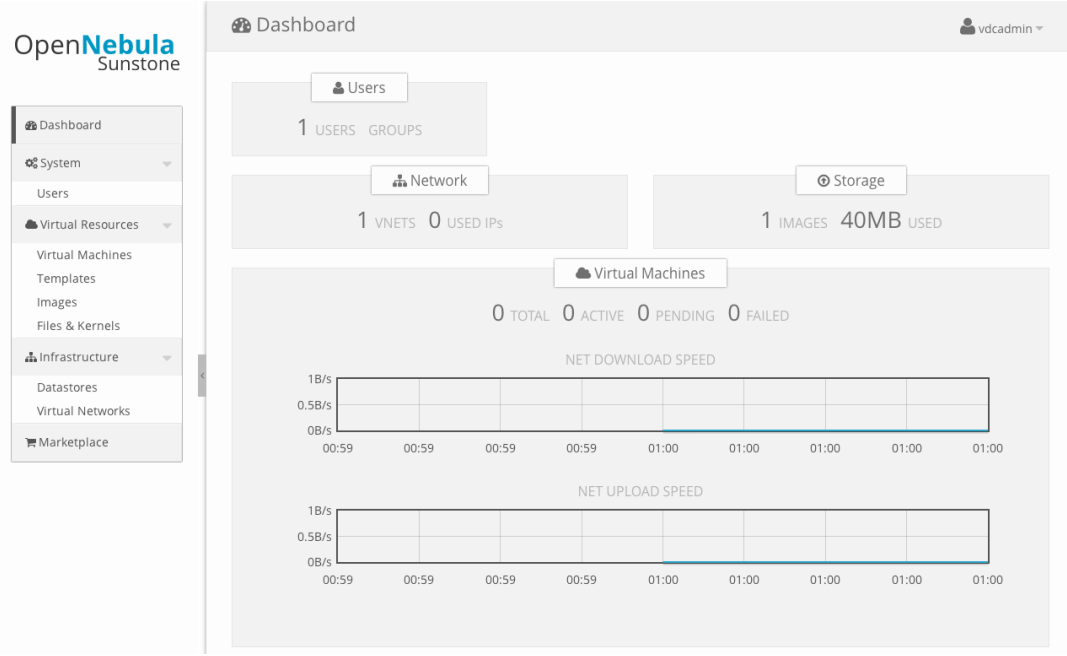
En las siguientes imágenes mostraremos el aspecto que presentan los diferentes tipos de vistas:

Vista de Administrador del sistema:



Fuente: http://opennebula.org/_media/admin_view.jpg

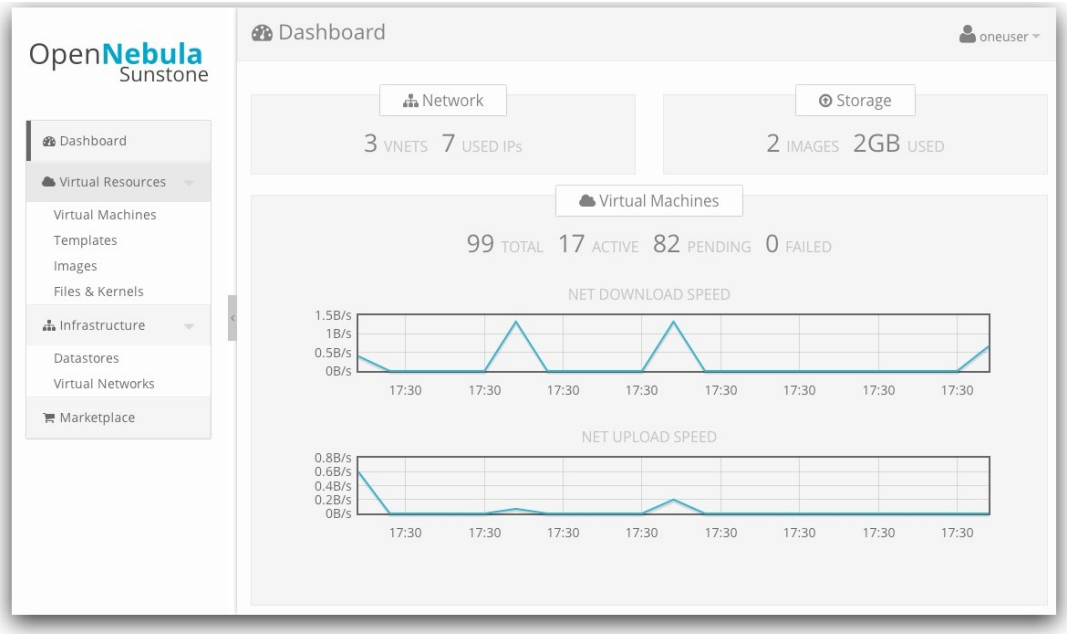
Vista del administrador de **datacenters**:



Fuente: http://opennebula.org/_media/vdcadmin_view.png

Nota: en nuestro caso, como solo vamos a administrar un único **datacenter** virtualizado (no estamos contemplando una solución **Multi-tier**), la vista del administrador del **datacenter** coincidirá con la del administrador de la plataforma).

Vista de usuario:



Fuente: http://opennebula.org/_media/user_view.jpg

3.3 Especificación de estándares, normas de diseño, y construcción

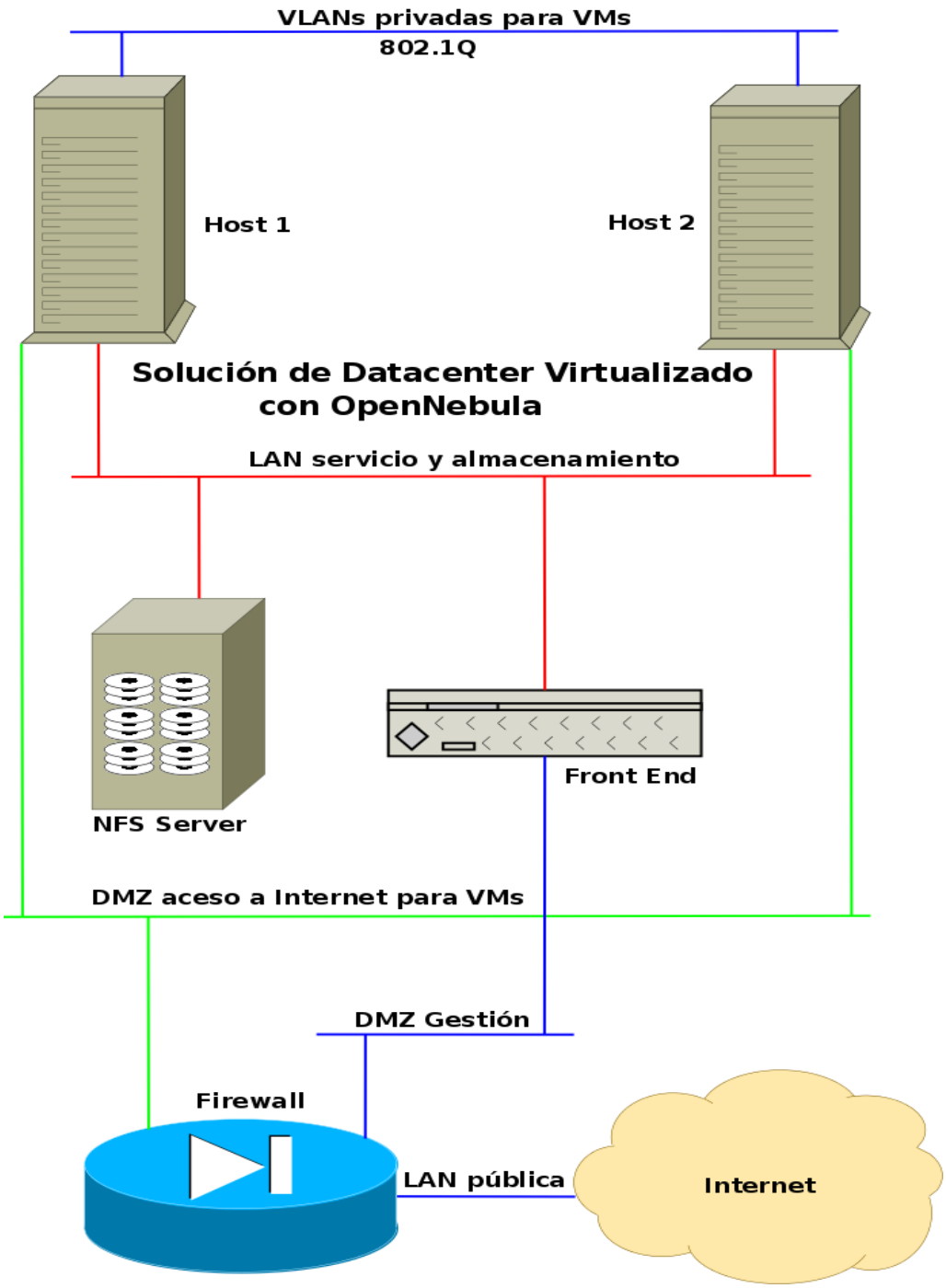
En nuestro proyecto, utilizaremos los siguientes estándares y herramientas, todas ellas con la filosofá del **Software Libre**:

- Documentos de diseño: todos los documentos de diseño utilizarán formatos y especificaciones abiertas, que sean compatibles multiplataforma para que se pueda trabajar con ellos sobre diferentes sistemas operativos. Como formato de adoptado se utilizará **ODF** [69], el cual está ampliamente adoptado como estándar y permite exportar los documentos directamente a otros formatos para su consulta (**XML**, **PDF**, etc.).
- Diagramas de diseño: aplicando el mismo criterio anterior, utilizaremos **Dia** [70], como herramienta de diseño de diagramas, debido a su versatilidad para trabajar con diferentes modelos de diagramas (soporta incluso **UML** [71]), y su portabilidad a otros sistemas operativos.
- Herramientas de documentación técnica, seguimiento de incidencias, tareas de mantenimiento, y gestión de proyectos: se utilizará como plataforma unificada de documentación y gestión del proyecto a **Redmine** [72] por ser una plataforma ampliamente utilizada, versátil, potente, que abarca todas las funcionalidades requeridas para poder desarrollar y mantener un proyecto de estas características, y que cuenta con gran experiencia de uso por parte del alumno en el desempeño diario de su trabajo profesional. Sin duda, esta herramienta resulta clave para desplegar, documentar, y mantener proyectos de este tipo.

4. Implantación de la solución

4.1. Diagrama del sistema a implantar

Después de haber descrito cada uno de los subsistemas con las opciones posibles, vamos a pasar a la fase de implementación de la solución diseñada, la cual se puede apreciar en el siguiente diagrama:



En el diagrama anterior, hemos tratado de utilizar el equipamiento mínimo necesario para poder implantar una solución orientada a dar servicios en producción, permitiendo la escalabilidad horizontal de la solución, conforme se detecte que su dimensionado se va quedando limitado, respecto al crecimiento de los servicios desplegados. Esto se traduce en el listado mostrado en la siguiente tabla:

Nombre	Modelo	Características
Host1	Dell R710	12 CPUs/64G RAM/600G RAID10
Host2	Dell R710	12 CPUs/64G RAM/600G RAID10
Frontend	Dell R610	4 CPUs/8G RAM/600G RAID10
NFS Server	Dell R720XD	4 CPUs/8G RAM/15TB RAID10
LAN switch VMs	Cisco 3560X	24 puertos Gigabit Ethernet RJ45
LAN switch Servicio	Cisco 3560X	24 puertos Gigabit Ethernet RJ45
Firewall	Cisco ASA5520	8 puertos Gigabit Ethernet RJ45

4.1.2. Descripción del sistema físico empleado

Siguiendo el diagrama y el listado de equipos necesarios, comentamos las funciones de cada uno, para que tenga sentido con la descripción funcional desarrollada en los puntos anteriores:

- Los **Hosts 1 y 2** son los responsables de soportar las hipervisoras **KVM**, para mantener las máquinas virtuales del **datacenter**. Las necesidades fundamentales de estos equipos son **CPUs**, memoria, y puertos de red, para soportar con ancho de banda suficiente las tres redes necesarias.
- El **Frontend** es el responsable de ejecutar todos los procesos de administración de **OpenNebula**, así como el servidor **Sunstone** para la administración por interfaz **web**.
- El **servidor NFS** es el responsable de proporcionar los ficheros de las imágenes instanciadas correspondientes a las máquinas virtuales que están ejecutándose en los **hosts 1** y **2**. Además contiene las imágenes base necesarias para crear nuevas instancias. No necesita tener capacidad de **CPU** ni memoria, pero si alta densidad de interfaces de red de alto rendimiento, puesto que todos los accesos a disco duro de las máquinas virtuales en ejecución de la plataforma lo harán directamente sobre el servidor **NFS**.

- El **LAN switch VMs** es el responsable de interconectar los dos **hosts** entre si mediante **trunks** con encapsulación **802.1Q**, para poder desplegar entre los **hosts** las **VLANs** de los usuarios, permitiendo la comunicación entre máquinas virtuales situadas en diferentes **hosts** través de las **VLANs** dedicadas. Para esta red se utilizaran al menos 4 puertos en **bonding** [73] con **LACP** [74] en cada **host**, formando un **port-channel** [75] configurado en el lado del **LAN switch**.
- El **LAN switch de servicio** es el responsable de interconectar entre sí el **Frontend**, los dos **hosts**, y el servidor **NFS**, en una red exclusiva para el tráfico de gestión de la plataforma, y de acceso a los **datastores** por **NFS**. Para ello se utilizaran varios (4) puertos en **bonding** con **LACP** en cada servidor, formando un **port-channel** configurado en el lado del **LAN switch**.
- El **firewall** de salida a internet gestionará a su vez tres redes diferentes:
    1. La **DMZ** [76] de salida a internet de las máquinas virtuales que se están ejecutando en los **hosts**. Podemos asumir un par de puertos por cada **hosts** para esta función. La **DMZ** tiene direccionamiento público asignado, para aquellas máquinas virtuales que así lo demanden.
    2. La **DMZ** de gestión del **Frontend**, la cual usará un puerto aparte dedicado en el **firewall**. Es muy importante que esta red de gestión se encuentre en una **DMZ** completamente aislada del resto, ya que sino correríamos el riego de ver como nuestra plataforma es atacada por una o varias máquinas virtuales comprometidas de alguno de nuestros clientes.
    3. La red pública (insegura) de salida a internet. En este caso podemos utilizar un par de interfaces de red para salida al exterior, ya que no esperamos un consumo de ancho de banda a internet superior a **2G**.

En el caso de las **DMZs**, se ha decidido aprovechar los recursos de puertos de red que brinda el **firewall** seleccionado, para evitar tener que instalar otro **LAN switch** dedicado a esta labor (por motivos de seguridad, los **LAN switches** de las **DMZs** deberían de estar completamente aislados y no utilizados para otras redes, ni siquiera otras **DMZs**, ya que un problema en la configuración podría agrupar puertos inesperados en la misma **VLAN**, comprometiendo toda la plataforma y los servicios de los clientes). En el caso de la **DMZ** de gestión, un único puerto directamente conectado entre el **Frontend** y el **firewall** será suficiente.

Todo el equipamiento necesario para esta solución se puede alojar en un único **rack** de **19”** con doble regleta de alimentación.

4.2 Fases de la implantación

En este apartado comentaremos las fases de implantación necesarias para dejar la plataforma instalada y lista para realizar las pruebas. Podemos describir las siguientes fases:

4.2.1. instalación y conexionado físico de los equipos

Siguiendo el diagrama y el listado de equipos necesarios, hemos procedido con el desempaquetado, e instalación física de los equipos en los **racks** asignados, comprobando que tenemos todos los materiales en perfecto estado, y que no falta nada necesario para continuar con las siguientes fases.

Una vez instalados los equipos en los **racks**, hemos continuado con los cableados (alimentación, tierra, y cableado de red), cada uno tendido de forma ordenada y por la vertical correspondiente, para no entorpecer el crecimiento futuro dentro de cada **rack**, y evitar interferencias electromagnéticas.

Una vez completados y revisados todos los pasos realizados, de acuerdo al diagrama de conexiones, procedimos con la siguiente fase.

4.2.2. Configuración y gestión del equipamiento de red

Siguiendo un orden correcto, debemos de comenzar con el encendido y configuración del equipamiento de red. Procedimos a encender y configurar, de acuerdo al esquema de red, los dos **lan switches**, el **firewall**, y el core de la red (no incluido en este proyecto, pero ya implantado en nuestro **datacenter** físico, para poder conectar el puerto público del **firewall** a **internet**), con el fin de montar la capa de conectividad, necesaria en las posteriores fases de instalación de los servidores. Es muy importante señalar, que debemos poner los medios para tener gestión remota de todos los elementos de la red de forma segura (i.e. los dos **LAN switches** y el **firewall**), a través de redes privadas dedicadas de gestión.

Una vez completada y revisada toda la configuración, y comprobado el correcto acceso remoto de gestión en los equipos de red implicados, hemos abordado la siguiente fase.

4.2.3. Instalación del sistema operativo y configuración básica de red en los servidores

Siguiendo el orden preestablecido, continuamos con la instalación del sistema operativo en los 4 servidores utilizados. En nuestra solución, hemos adoptado la misma distribución de sistema operativo para los cuatro equipos: **CentOS 6.4** a **64 bits**. Para ello configuramos y dejamos armarse el sistema **RAID 10** [77] por la controladora hardware en cada equipo (en el servidor **NFS** se tardó varias horas, debido a la capacidad de disco empleada). Hemos optado por **RAID 10** porque las controladoras hardware de los sistemas actuales ya lo soportan, y porque demuestra ser mas segura frente a fallos simples. Una vez montado el **RAID** en los equipos, hemos realizado una instalación estándar de **CentOS 6.4** utilizando **LVM** [78] en lugar de particiones físicas (en realidad hemos definido una partición física para el directorio **/boot**, y otra que engloba el resto de la capacidad disponible en cada máquina para el **LVM**).

Nota: eventualmente, para facilitar las tareas de configuración y puesta a punto de los equipos, hemos conectado un puerto de red de cada equipo a una **LAN** privada con servidor **DHCP** y salida al exterior, lo que nos ha permitido actualizar los sistemas una vez completada la instalación, incluyendo el servidor **NFS**, que en condiciones normales de producción no tendrá salida al exterior.

Una vez completadas las tareas de instalación y actualización básicas de los paquetes del sistema operativo (configuración de las cuentas de administración, puesta en hora de los equipos, etc.), proseguimos con la configuración de los interfaces físicos de red, de tal forma que al acabar la tarea todos los equipos se vean entre si con sus **IPs** definitivas, en cada una de las redes definidas según la solución diseñada. Este paso es muy importante, y es el que garantiza una vez completado que no haya problemas en las siguientes fases de la instalación.

4.2.4. Instalación de los repositorios y paquetes necesarios para los servicios de la plataforma

Una vez comprobada la perfecta conectividad de las máquinas entre si, y manteniendo la salida al exterior a través de la red privada empleada en la fase anterior, procedimos a instalar los repositorios y paquetes necesarios en cada servidor:

- En el servidor **NFS** sólo son necesarios los paquetes correspondientes al servicio **NFS** y la configuración de los interfaces de red por **bonding** con **LACP**.
- En el caso de los dos **hosts** y el **Frontend**, es necesario instalar el repositorio **EPEL** [79] de **Fedora** [80], y el repositorio de los

paquetes de **OpenNebula**, disponibles desde los servidores de **C12G Labs** [81].

Una vez instalados los paquetes correspondientes a cada servidor, procedimos con la siguiente fase de configuración.

4.2.5. Configuración preliminar de los servicios de la plataforma en los servidores

En esta fase comenzamos con la configuración preliminar de los servicios en cada uno de los servidores. Hemos creado las cuentas de usuario, permisos, preparado los ficheros de configuración de los **daemons**, automatizado el arranque automático de estos al reiniciar la máquina, automatizado el acceso al usuario **oneadmin** entre todas las máquinas a través de la red de servicio sin necesidad de utilizar el **password**, montado las unidades de disco en el servidor **NFS** y los **hosts** y **Frontend**, configurado los **bridges** y el soporte por **VLAN 802.1Q** en los **hosts**, etc.

Nota: todos estos pasos detallados de instalación y configuración de la solución se facilitan en forma de materiales entregables en los **anexos 1** (entorno de pruebas) [82] y **2** (entorno de producción) [83] de este proyecto. Con el fin de no redundar y extender innecesariamente la memoria, sólo se detallarán a modo de resumen funcional algunos de los puntos más representativos, dejando los detalles de todos los comandos ejecutados en los anexos citados.

4.2.6. Pruebas unitarias de la plataforma

Con una configuración inicial para pruebas, se han realizado las pruebas unitarias que demuestran el correcto funcionamiento de cada servicio en la plataforma. A modo de resumen, podemos mencionar las siguientes:

- Comprobación de los sistemas físicos de los servidores: discos duros, memoria, **CPU**, interfaces de red, fuentes de alimentación, etc., así como de los **lan switches**, **routers**, y demás equipamiento empleado en la solución.
- Comprobación de la correcta activación de la extensión de la aceleración **hardware** para virtualización por **KVM** en la **BIOS** de los **hosts**.
- Comprobación de la perfecta conectividad a nivel 2 y 3 de todos los puertos de red utilizados en la plataforma.
- Comprobación de la correcta carga de los módulos del **kernel** para virtualización, configuración de interfaces de red, **VLANs**, **bridges**, **802.1Q**, etc. en los **hosts**.

- Comprobación de los permisos y cuentas de usuario necesarios para correr el software de la solución.
- Comprobación del correcto arranque de los procesos de **OpenNebula**, sin detectar errores en los ficheros de **log** correspondientes a los procesos, y al sistema en general.

Una vez completada esta fase con éxito, procedimos a realizar los cambios de configuración necesarios para acometer el siguiente paso en las pruebas.

4.2.7. Pruebas de interoperabilidad

A partir de los cambios preparados en la configuración en el paso anterior, procedimos con las pruebas de interoperabilidad, para comprobar que todos los componentes y servicios ubicados en cada servidor, interaccionan correctamente con el resto de componentes de la plataforma. A modo de resumen podemos citar los siguientes puntos:

- Comprobación del acceso remoto por **ssh** sin **password** entre todos los servidores empleados, con la cuenta de administrador del sistema, **oneadmin**.
- Comprobación del acceso remoto por **sudo** desde el servidor **Frontend** a los **hosts**.
- Comprobación de las rutas correctas de las aplicaciones utilizadas por la solución, en los ficheros de configuración de **OpenNebula**.
- Comprobación del correcto acceso y permisos por **NFS** desde los **host** y el **Frontend** a los **datastores** compartidos.
- Comprobación de la correcta ejecución de los comandos remotos lanzados por el **Frontend** sobre los **hosts** de virtualización para poder instanciar las imágenes de las máquinas virtuales, realizar las operaciones del ciclo de vida de estas, configurar/desconfigurar los interfaces de red, **VLANs**, **bridges**, etc. sin errores de permisos o de ejecución de las herramientas del sistema operativo empleadas para ello.
- Comprobación del correcto acceso desde el interfaz web **Sunstone** a las consolas de las máquinas virtuales por **VNC**, a través del servidor **proxy** por **websockets**.
- Comprobación del correcto funcionamiento de los sistemas de autenticación que soporta **OpenNebula**, necesarios para poder operar tanto desde el **Sunstone** como desde el **API** de comandos **online**.

Una vez completada esta fase, modificamos los cambios necesarios en la configuración para poder realizar las pruebas de integración en la siguiente fase.

4.2.8. Pruebas de integración

A partir de los cambios preparados en la configuración en el paso anterior, procedimos con las pruebas de integración. A modo de resumen se mencionan algunos ejemplos:

- Preparación de **datastores** para trabajar con imágenes de tipo **qcow2**.
- Realización de pruebas con imágenes persistentes y no persistentes.
- Preparación de imágenes contextualizadas para poder probarlas y utilizarlas posteriormente en la plataforma.
- Preparación y uso de **templates** para definir **datastores**, imágenes, redes, instancias, cuotas de usuario, etc.
- Utilización de instancias **router** contextualizadas para proporcionar el acceso de las máquinas virtuales al exterior.
- Creación de cuentas de usuario, asignación de cuotas y permisos, etc.
- Asignación de vistas para los diferentes perfiles de usuario en el **Sunstone**.
- Realización de operaciones sobre el ciclo de vida de las imágenes instanciadas, tanto persistentes como no persistentes.
- Instalación de **Ganglia** como sistema de monitorización de la solución.
- Migración de la plataforma completa a direccionamiento público, una vez terminadas las pruebas de integración.

Una vez completada esta fase, modificamos los cambios necesarios en la configuración para poder realizar las pruebas de escalabilidad en la siguiente fase.

4.2.9. Pruebas de escalabilidad

A partir de los cambios preparados en la configuración en el paso anterior, procedimos con las pruebas de escalabilidad y rendimiento, con la finalidad de comprobar que los sistemas llegan a los límites esperados, sin degradación del servicio, al mismo tiempo que podemos registrar los parámetros de consumo de recursos en la plataforma de forma precisa.

Para poder escalar la plataforma, se han adoptado una serie de medidas para agilizar la fase de despliegue de las instancias:

- Se han realizado mejoras en los **scripts** utilizados por el **datastore raw** para poder clonar las imágenes en el **datastore** compartido sin necesidad de transferir la imagen a través de los interfaces de red durante el proceso de clonación.
- Se ha creado un **datastore** de tipo **qcow** para poder instanciar imágenes de tipo **qcow2**, usando los ficheros diferenciales para almacenar los deltas con los cambios sobre la imagen original.
- Se han modificado los parámetros del **scheduler** para agilizar el despliegue y las operaciones realizadas sobre la plataforma al mayor ritmo posible.

Una vez realizados estos cambios se han realizado dos pruebas diferentes para probar el rendimiento y escalabilidad de la solución:

- Se ha invocado **1500** instancias de una imagen reducida de **GNU/Linux**, que sólo requiere **64M** de **RAM**, **0.1 CPUs**, y **40M** de disco duro, para comprobar el rendimiento del despliegue de imágenes hasta el límite posible, con los parámetros del **scheduler** modificados para la prueba.
- Después de eliminar las instancias anteriores, se ha devuelto los parámetros del **scheduler** a los valores originales que tenía por defecto, y se ha realizado otra prueba de invocar **240** instancias de **CentOS 6.5 a 64 bits**, con **0.5 CPUs**, **512M** de **RAM**, y **3.2G** de disco duro cada una.

Los resultados y conclusiones obtenidos de ambas pruebas quedan recogidos en el **anexo 2** (apartado **4.4.1**) [83].

Una vez completada esta fase dejamos la configuración de preproducción para la realización de las pruebas de aceptación.

4.2.10. Pruebas de aceptación

A partir de los cambios preparados en la configuración en el paso anterior, procedimos con las pruebas de aceptación, confirmando que la plataforma proporciona toda la funcionalidad esperada para el entorno de producción. A continuación se detallan algunas de las operaciones realizadas:

- Preparación de las cuentas de usuario.
- Asignación de cuotas de uso por usuario y por defecto.
- Preparación de plantillas de uso personalizadas para las redes de cada usuario (red con acceso a internet mediante el **router** virtual, y red privada para comunicar las instancias del usuario entre si).

- Preparación de imágenes contextualizables de uso para los usuarios en los **datastores raw** y **qcow**.
- Preparación de las instancias de los **routers** virtuales para cada usuario, para permitir la salida a internet mediante **NAT** y **port forwarding**.
- Preparación de plantillas de imágenes de uso para los usuarios, incluyendo plantillas de uso general que permiten instanciar una imagen con direccionamiento público a internet, gracias a la aplicación de las cuotas de limitación de uso asignadas a cada usuario.
- Comprobación del correcto aislamiento de de las **VLANs** de usuario, incluso solapando direccionamiento **IP** y **MACs**.
- Comprobación de las vistas individualizadas y permisos de cada usuario, a través del **Sunstone** y del **API** de comandos **online**.
- Comprobación del acceso remoto y por consola de cada usuario a sus respectivas instancias, usando sus credenciales.
- Comprobación que las operaciones realizadas sobre el ciclo de vida de las instancias para cada usuario a través del **Sunstone** y del **API** de comandos **online** funcionan correctamente.

Después de completar las pruebas de aceptación, la plataforma ha quedado dispuesta para ser entregada en producción.

4.3 Plan de Formación

A pesar de que la plataforma ya estaría preparada para desplegar los servicios en producción, aún queda la parte más importante del proyecto: la transferencia de conocimiento al personal responsable de mantener y gestionar la solución implantada.

Lo mas aconsejable es implicar desde el primer momento a los responsables de administrar la plataforma, haciéndoles participar en todas las fases de instalación, configuración, pruebas, etc. Al fin y al cabo ellos son administradores de sistemas, están familiarizados con las tecnologías empleadas, y ellos serán los responsables de asegurar la continuidad del proyecto, remplazando equipos, o ampliándolos en el futuro.

Después de haber trabajado codo con codo con las personas de mayor perfil técnico y responsables de administrar y mantener la plataforma, se hace aconsejable definir unos perfiles de formación, para el resto de empleados en plantilla que tengan que ver con la solución de servicios en el **datacenter** virtualizado.

4.4 Conclusiones personales al final del proyecto

Este proyecto me ha permitido explorar y abordar las tecnologías disponibles en el campo de la virtualización, y en especial, en el campo de la virtualización de **datacenters**, y su gestión integral a través de **OpenNebula**.

Las prácticas desarrolladas demuestran su viabilidad como solución perfectamente aplicable al entorno empresarial, para permitir desplegar servicios de **IaaS** sobre infraestructura privada de forma eficiente, a nivel de costes de infraestructura, mantenimiento, consumo eléctrico, escalabilidad, funcionalidad, y recursos humanos. Sin lugar a dudas, representa un gran avance en este campo.

La solución utilizada (**OpenNebula**) ha supuesto un gran acierto desde el comienzo del proyecto, porque ha confirmado su condición de **vendor-agnostic**, su facilidad de configuración y uso, su amplio abanico de posibilidades y tecnologías soportadas para administración de los recursos disponibles (almacenamiento, red, autenticación, **APIs** de gestión, etc.), su gran flexibilidad para adaptarla a las necesidades de cada entorno, y sobre todo, porque permite darle el enfoque de plataforma de gestión de **cloud** orientada a la virtualización de **datacenters** para el ámbito empresarial, que era el objetivo del presente trabajo. Y todo ello utilizando exclusivamente soluciones de Software Libre, con lo que la solución ofrece un sólido apoyo y reconocimiento por parte de la comunidad, una larga madurez como proyecto, y amplias expectativas de futuro.

Aunque la solución escogida facilita bastante las tareas de mantenimiento y la curva de aprendizaje de uso de la plataforma, conocer y dominar las tecnologías subyacentes requiere un nivel de experiencia bastante elevado y en continua formación, si se quiere aprovechar al máximo las posibilidades existentes que ofrece la solución, y sobre todo, permitir su escalado sin problemas de rendimiento en el servicio. Este ha sido uno de los principales retos afrontados por el alumno durante el desarrollo de las prácticas.

Por otra parte, y aunque la solución cuenta con amplia aceptación y madurez, se puede comprobar por los **roadmap**, tanto del proyecto **OpenNebula** como de sus alternativas existentes en el mercado, que es una materia en constante y rápida evolución (sólo en el año 2013 han salido 3 versiones estables de **OpenNebula**), no sólo añadiendo nuevas funcionalidades y opciones tecnológicas soportadas, sino también solucionando problemas existentes en las versiones anteriores. Esto he podido comprobarlo durante las prácticas, con los problemas de escalado experimentados en los procesos de monitorización de las instancias, o la imposibilidad de aplicar múltiples grupos a los permisos de los recursos asignados a los usuarios. Estos dos problemas mencionados quedan solventados en la nueva versión

de **OpenNebula**, (la **4.4**, llamada **Retina**), aparecida hace sólo unos días [84], y se espera la versión **4.6** para Marzo de 2014. Todo avanza muy deprisa, y mantenerse al día exige dedicación completa en la materia. Sin duda nos tocará presenciar unos avances espectaculares en los próximos años en el campo del **cloud computing**.

Sólo me queda comentar que la oportunidad de haber realizado este proyecto ha supuesto un reto importante, lo que a su vez me ha aportado una experiencia complementaria muy enriquecedora a mi carrera profesional. Me siento muy agradecido por ello, y es algo que espero poder aplicar en el futuro, en el desempeño de mi trabajo.

Referencias

[1]http://www.gartner.com/it-glossary/cloud-management-platforms

[2]http://en.wikipedia.org/wiki/IaaS#Infrastructure_as_a_service_.28IaaS.29

[3]http://opennebula.org/

[4]http://aws.amazon.com/

[5]http://blog.opennebula.org/?p=4042

[6]http://searchcloudcomputing.techtarget.com/definition/cloud-bursting

[7]http://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine

[8]http://en.wikipedia.org/wiki/Xen

[9]http://en.wikipedia.org/wiki/Internet_protocol_suite

[10]http://www.eucalyptus.com/

[11]http://www.openstack.org/

[12]http://www.rackspace.com/

[13]http://cloudstack.apache.org/

[14]http://en.wikipedia.org/wiki/Citrix

[15]http://en.wikipedia.org/wiki/Apache_foundation

[16]http://www.gnu.org/licenses/gpl-3.0.html

[17]http://en.wikipedia.org/wiki/Apache_License

[18]http://en.wikipedia.org/wiki/Cloudstack

[19]http://opennebula.org/about:why

[20]http://en.wikipedia.org/wiki/Vmware

[21]http://en.wikipedia.org/wiki/Libvirt

[22]http://en.wikipedia.org/wiki/Ganglia_%28software%29

[23]http://es.wikipedia.org/wiki/LOPD

[24]http://en.wikipedia.org/wiki/Snapshot_%28computer_storage%29

[25]http://en.wikipedia.org/wiki/Ruby_%28programming_language%29
[26]http://en.wikipedia.org/wiki/Python_%28programming_language%29
[27]http://en.wikipedia.org/wiki/Websocket
[28]http://en.wikipedia.org/wiki/SNMP
[29]http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud
[30]http://en.wikipedia.org/wiki/Open_Cloud_Computing_Interface
[31]http://en.wikipedia.org/wiki/Software_quality_management
[32]http://en.wikipedia.org/wiki/Jenkins_%28software%29
[33]http://opennebula.org/software:testing
[34]http://en.wikipedia.org/wiki/Hypervisor
[35]http://en.wikipedia.org/wiki/Paravirtualization
[36]http://en.wikipedia.org/wiki/Full_virtualization
[37]http://en.wikipedia.org/wiki/Hardware-assisted_virtualization
[38]http://www.linux-kvm.org/page/Virtio
[39]http://www.linux-kvm.org/page/WindowsGuestDrivers/Download_Drivers
[40]http://en.wikipedia.org/wiki/Hardware-assisted_virtualization#cite_note-5
[41]http://cloudcomputing.sys-con.com/node/2276400
[42]http://en.wikipedia.org/wiki/QEMU
[43]http://en.wikipedia.org/wiki/Qcow
[44]http://en.wikipedia.org/wiki/IMG_%28file_format%29
[45]http://en.wikipedia.org/wiki/Network-attached_storage
[46]http://en.wikipedia.org/wiki/Iscsi
[47]http://en.wikipedia.org/wiki/VMFS
[48]http://en.wikipedia.org/wiki/Ceph_%28storage%29
[49]http://en.wikipedia.org/wiki/Ramdisk
[50]http://en.wikipedia.org/wiki/Network_File_System
[51]http://en.wikipedia.org/wiki/Secure_Shell
[52]http://en.wikipedia.org/wiki/VMFS
[53]http://en.wikipedia.org/wiki/Firewall_%28computing%29
[54]http://en.wikipedia.org/wiki/Broadcast_storm
[55]http://en.wikipedia.org/wiki/MAC_spoofing
[56]http://en.wikipedia.org/wiki/Virtual_LAN
[57]http://en.wikipedia.org/wiki/802.1q
[58]http://ebtables.sourceforge.net/

[59]http://en.wikipedia.org/wiki/Iptables

[60]http://openvswitch.org/

[61]http://en.wikipedia.org/wiki/802.1ad

[62]http://ganglia.sourceforge.net/

[63]http://en.wikipedia.org/wiki/Access_control_list

[64]http://en.wikipedia.org/wiki/SQLite

[65]http://en.wikipedia.org/wiki/Mysql

[66]http://docs.aws.amazon.com/AWSEC2/latest/APIReference/Welcome.html

[67]http://opennebula.org/documentation:archives:rel4.2:cli

[68]http://opennebula.org/documentation:archives:rel4.2:sunstone

[69]http://en.wikipedia.org/wiki/OpenDocument

[70]http://es.wikipedia.org/wiki/Dia_%28programa%29

[71]http://en.wikipedia.org/wiki/Unified_Modeling_Language

[72]http://es.wikipedia.org/wiki/Redmine

[73]http://en.wikipedia.org/wiki/Link_aggregation

[74]http://en.wikipedia.org/wiki/Link_Aggregation_Control_Protocol

[75]http://es.wikipedia.org/wiki/EtherChannel

[76]http://en.wikipedia.org/wiki/DMZ_%28computing%29

[77]http://en.wikipedia.org/wiki/RAID_10#RAID_1.2B0

[78]http://en.wikipedia.org/wiki/Logical_volume_management

[79]http://fedoraproject.org/wiki/EPEL

[80]http://fedoraproject.org/

[81]http://c12g.com/

[82] “Anexo 1: Configuración del entorno de pruebas”: **TFM-Anexo1_Adm_de_redes_y_sistemas_operativos_Jose_Antonio_Montes_20131231_V1.5.pdf**

[83] “Anexo 2: Configuración del entono de producción”: **TFM-Anexo2_Adm_de_redes_y_sistemas_operativos_Jose_Antonio_Montes_20131231_V1.5.pdf**

[84]http://blog.opennebula.org/?p=5633