UNIVERSIDAD DE CANTABRIA Facultad de Ciencias

Proyecto Fin de Carrera

DESPLIEGUE DE UN ENTORNO IAAS DE COMPUTACIÓN EN LA NUBE PRIVADO

Deploying a Private IAAS Cloud Computing Environment

Para acceder al título de

INGENIERO EN INFORMÁTICA

Autor: Mariano Benito Hoz
Director: Enrique Vallejo Gutiérrez
Codirector: Roberto Hidalgo Cisneros

Septiembre - 2013

Agradecimientos

Ahora que tocan a su fin con la presentación de este Proyecto, me gustaría dedicar unas lineas a todas las personas que, de un modo u otro, han estado presentes en estos cinco años de carrera.

En primer lugar, me gustaría darles las gracias al director de mi proyecto y a uno de sus compañeros de departamento, por su dedicación, por sus consejos, por su profesionalidad y por haberme sabido guiar hasta este punto.

En segundo lugar, gracias al codirector del proyecto, por haberme propuesto un tema que despertó mi interés el día que me abrió la puerta de su oficina y por no haberla cerrado desde entonces.

En tercer lugar, quisiera expresarle mi afecto a mi padre, porque aunque el camino vivido, tanto en la cercanía como en la lejanía nunca ha sido fácil, sé que siempre contaré con todo su apoyo.

En cuarto lugar, a las personas con las que he podido compartir tantos momentos a lo largo de estos años de carrera.

Por ultimo, y más importante, mis gracias más sinceras a mi abuela, por haberme acompañado en los primeros momentos de estos años, y a mi pareja, por estar a mi lado en el final de los mismos y el comienzo del resto.

Muchas gracias a todos.

Resumen

El departamento de operaciones de CIC Consulting Informático ofrece sus servicios *Information Technology* (IT) a clientes de la compañía y al resto de departamentos de la propia empresa, los cuales se dedican principalmente a la programación en diversas áreas de negocio. Las necesidades de estos departamentos están en continuo crecimiento y, además, el departamento de operaciones está asumiendo cada vez más carga de trabajo por parte de sus clientes. Estos hechos están provocando que poco a poco el servicio ofrecido a las áreas de desarrollo por el departamento de operaciones se esté volviendo menos ágil y genere retrasos a estos departamentos.

Junto con el responsable de producción de la compañía, el coordinador del área de operaciones ha llegado a la conclusión de que una posible forma de abordar este problema es la implantación de un sistema que permita a los programadores el auto-aprovisionamiento de la infraestructura necesaria para realizar sus proyectos. Con esto se lograría la flexibilidad y ligereza que se busca para el servicio a prestar por el departamento de operaciones. Además, el sistema debe permitir mantener el control de ciertos aspectos como el uso de los recursos, el dimensionado de las máquinas virtuales, la seguridad de la plataforma y la estandarización tecnológica, entre otros.

Por todos estos motivos, este Proyecto Fin de Carrera tiene como objetivo el despliegue e implantación de un entorno *Infrastructure As A Service* (IAAS) de Cloud Computing privado para las áreas de desarrollo de CIC. Tras un estudio inicial de las herramientas disponibles, se optó por elegir como base el proyecto OpenStack. Los motivos de esta elección se basan en que dicha herramienta cumple los requisitos especificados a alto nivel, permite trabajar con tecnologías abiertas y sobre la implementación de lo que pretende ser el primer estándar de IAAS. Durante la realización del proyecto se ha trabajado sobre distintos campos, como pueden ser la virtualización, los sistemas de almacenamiento y las redes de comunicaciones. En ellos se han utilizado diferentes tecnologías y estándares como paravirtualización, `iSCSI`, volúmenes lógicos, `VLANs`, etc. Además, aunque el equipamiento físico sobre el que se llevó a cabo el proyecto no es el habitual con el que trabaja la compañía, fue necesario plantear y adaptar el despliegue a las tecnologías usadas habitualmente en esta empresa.

Palabras clave: Computación en la nube, `OpenStack`, IAAS.

Abstract

CIC Consulting Informático Operations Department offers its Information Technology (IT) services to company customers and to the rest of its own departments, which are dedicated to programming in different business areas. The necessities of these departments are continuously growing and, moreover, the Operations Department is assuming a greater amount of work. These facts are causing that the service offered to the development areas by the Operations Department is becoming less flexible and generating delays to these departments.

Together with the Production Responsible of the company, the Operations Area coordinator came to the conclusion that a possible way to deal with this problem is to introduce a system which allows programmers to supply themselves with the necessary infrastructure for their projects. With this, the flexibility and agility needed by the Operations Department would be achieved. Furthermore, the system should allow users to keep the control of some aspects, such as the use of the resources, the measurement of virtual machines, the security of the platform and the technological standardization.

Therefore, the aim of this Degree Project is to deploy and implement an Infrastructure As A Service (IAAS) private Cloud Computing environment for CIC Development Areas. The OpenStack project was chosen as base after a first study of available tools. The reasons of this choice were that this tool achieves the high level requirements, allows users to work with open technologies and works over what is expected to be the first IAAS standard. During the development of this project, different fields have been tackled, such as virtualization, storage systems and communication networks. In each one, different technologies and standards have been applied, such as paravirtualization, `iSCSI`, logical volumes, `VLANs`, etc. In addition to this, despite the fact that the physical equipment used for implementing the project is not usual in the company, it was necessary to consider and adapt the deployment to the technologies most commonly used in this company.

Keywords: Cloud Computing, `OpenStack`, IAAS.

Índice general

Página

Índice de ilustraciones

Página

Índice de tablas

Página

1 Introducción

La empresa en la que se ha desarrollado el presente Proyecto Fin de Carrera está en constante evolución. Asimismo las necesidades IT de muchas de las áreas de la misma están en continuo crecimiento. Es objeto de este proyecto desplegar un sistema de computación en la nube privado de tipo infraestructura como servicio que flexibilice y agilice el aprovisionamiento de las mismas.

1.1. Antecedentes y motivación

Durante la última década, y con más intensidad recientemente, la cantidad de servicios que se prestan en internet ha sufrido un crecimiento vertiginoso. El ecosistema de herramientas desplegado es tan grande que algunas empresas han propuesto que los ordenadores ya no deben tener aplicaciones locales sino que un simple navegador y una conexión a internet es material suficiente para el uso diario que se hace de los mismos. Esta explosión de servicios abrió las puertas al hecho de que si las aplicaciones se pueden ofrecer de esta forma, la infraestructura también.

Amazon Web Services es un claro ejemplo de esta idea. Las principales motivaciones que pueden llevar a una empresa a usar este tipo de servicios se presentan a continuación:

- Sin inversión inicial: permiten sustituir una gran inversión inicial en infraestructuras por mensualidades en las que se paga en base a los recursos utilizados.
- Reducir los costes operativos: ofrecen el acceso a una plataforma altamente distribuida y con multitud de características a cambio de un coste muy inferior al supuesto por una tradicional.
- Capacidad flexible: permiten aprovisionar la cantidad de recursos necesarios y, tanto escalar como desactivarlos si ya no son necesarios.
- Velocidad y agilidad: ofrecen la posibilidad de disponer de la infraestructura solicitada de manera prácticamente instantánea.
- Alcance global: permiten ofrecer una menor latencia y una mejor experiencia de usuario al permitir desplegar la infraestructura en *DataCenters* distribuidos por todo el mundo.
- Pago por uso: ofrecen un modelo en el que sólo se paga por el consumo realizado.

Para poder visualizar algunas de estas ventajas, en la ilustración 1.1 se expone una representación de las fluctuaciones de la demanda a lo largo del tiempo que puede sufrir un servicio ofrecido por una empresa, junto con el coste que supondría un despliegue tradicional y el coste que supone el mismo si se adapta a las necesidades.

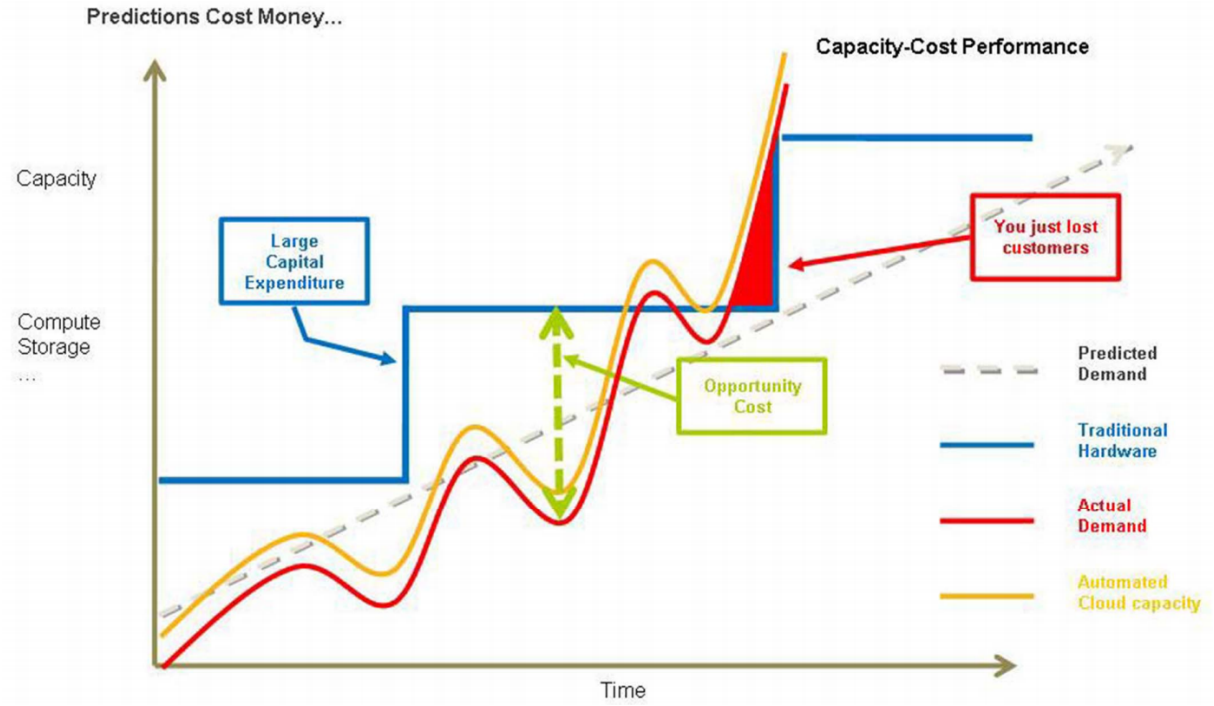


Ilustración 1.1: Fluctuación de la demanda y ventajas del *cloud*. Fuente: Amazon Web Services.

El carácter privado de estos despliegues no se centra tanto en el hecho de que sea fácil adaptar las necesidades *Information Technology* (IT) al negocio y con ello ahorrar costes, sino que presta más atención a la flexibilidad que aportan los mismos. Esta agilidad a la hora de aprovisionar infraestructuras es la que actualmente busca la compañía en la que ha sido desarrollado este proyecto.

Actualmente, cuando un equipo de desarrollo recibe un nuevo proyecto y requiere de cierta infraestructura computacional para realizar su trabajo, solicita esa infraestructura al área de operaciones. Tras analizar los requisitos, este departamento realiza un diseño, crea las máquinas virtuales necesarias y despliega el software pertinente para satisfacer los requisitos del departamento de desarrollo al cual se le cede posteriormente el control de los sistemas.

A modo de ejemplo, se expone el caso particular de un desarrollador de la compañía. Una de sus funciones es la de crear un paquete de instalación de uno de los productos de la empresa. Para poder probar la funcionalidad del mismo le solicita al departamento de operaciones un servidor con los siguientes paquetes *software* instalados y totalmente actualizados: `Windows Server 2008 R2`, `MS SQL Server 2012`, `IIS 7.5` y `.Net Framework 4`. El despliegue de una máquina virtual que satisface sus requisitos puede llevar aproximadamente dos días, aunque si bien es cierto que la virtualización y su capacidad de tomar instantáneas (*snapshots*) han permitido reducir esos tiempos, sigue siendo necesaria la intervención del departamento de operaciones para la gestión del entorno de virtualización, puesto que las áreas de desarrollo no tienen acceso al mismo.

Este proyecto también ha sido motivado por relaciones mantenidas con el Instituto de Física de Cantabria (IFCA). Este centro pretende acercar la computación de alto rendimiento, mediante las

bondades ofrecidas por el modelo de *cloud computing*, a la pequeña y mediana empresa. Puesto que esta idea comparte la fase de estudio inicial con el presente proyecto, este ha sido desplegado sobre equipamiento cedido por dicho centro de investigación.

1.2. Objetivo

Para resolver los problemas de agilidad expuestos previamente, se propone el despliegue de un sistema de computación en la nube privado de tipo *Infrastructure as a service* (IAAS) con el que las áreas de desarrollo de la compañía sean capaces de aprovisionarse de la infraestructura necesaria para la realización de sus proyectos.

Tras unas tareas iniciales a realizar por el departamento de operaciones en las que se crearán y publicarán en el sistema ciertas imágenes de las infraestructuras estándar usadas en los proyectos de desarrollo, los programadores e integradores contarán con la posibilidad de crear y gestionar máquinas virtuales en base a estas plantillas cuando así lo necesiten.

El sistema propuesto tiene que satisfacer bajo demanda las necesidades computacionales de las áreas de desarrollo de la empresa y, teniendo en cuenta que estas no son sólo máquinas virtuales, el sistema a implantar también tiene que ofrecer almacenamiento para las máquinas virtuales cuando así se requiera.

1.3. Fases del proyecto

La realización del proyecto se organizó en las fases que se mencionan a continuación, las cuales fueron guiadas por la metodología impuesta por la empresa en la que se ha desarrollado y que, a pesar de estar muy orientada al desarrollo de *software*, ha sido válida para el proyecto:

1. Estudio y análisis del problema: durante esta fase se analizó la problemática de la empresa y la forma de trabajo actual para obtener el documento de requisitos funcionales a entregar.

2. Diseño del sistema a desplegar: esta fase tuvo como objetivo la realización del documento de diseño del sistema y del plan de pruebas e implantación.

3. Construcción e implantación del sistema: los hitos marcados por esta fase fueron la creación del documento de construcción del sistema y el posterior despliegue del mismo.

4. Pruebas y validación: en esta última fase se procedió a ejecutar el plan de pruebas creado en la segunda fase del proyecto y a supervisar la validación del sistema.

1.4. Estructura del documento

El presente documento se ha estructurado en 7 secciones, las cuales se resumen a continuación.

1. **Introducción**: pretende que el lector entre en materia y capte las motivaciones del proyecto.
2. **Conceptos y herramientas**: breve introducción y nociones básicas sobre las tecnologías empleadas en este proyecto.
3. **Requisitos**: muestra los requisitos capturados para el sistema a desplegar.
4. **Infraestructura**: primera parte del diseño del sistema, en la cual se proporciona una sólida capa de conectividad, almacenamiento y virtualización.
5. **IAAS privado**: segunda parte del diseño del sistema, en la que se despliega un sistema de infraestructura como servicio de tipo privado.
6. **Verificación y validación**: muestra las pruebas de verificación y el proceso de validación del sistema desplegado.
7. **Conclusiones y trabajo futuro**: describe los objetivos alcanzados y las posibilidades de futuro.

El proceso de construcción e implantación del sistema se ha documentado en un *blog*[1] personal puesto que se considera interesante exponer este proceso al público que puede estar interesado en el mismo. Aunque es muy difícil que alguien realice un despliegue con las mismas características, algunas de ellas, y sobre todo, las ideas generales sí que pueden ser comunes.

[1] Durante las fases de diseño se comentarán las entradas que el lector puede consultar si le parece interesante el diseño.

2 Conceptos y herramientas

El objetivo principal del capítulo es realizar una breve introducción a las tecnologías empleadas en este proyecto, consiguiendo de esta forma que los conceptos aquí descritos queden claros en los siguientes capítulos del documento. Asimismo, a lo largo del capítulo se exponen referencias sobre las tecnologías expuestas por si al lector le interesa indagar más en un tema concreto.

2.1. Cloud Computing

"*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*" [NIS12]

Tras la definición formal mostrada previamente, se puede definir *Cloud Computing* o "computación en la nube" como un modelo de negocio que permite ofrecer servicios de computación de forma ilimitada, remota, ubicua, dinámica y basada en pago por uso. En este modelo, todo lo que puede ofrecer un sistema informático se ofrece como servicio, consiguiendo así que los usuarios puedan acceder a sus características sin ser expertos en la gestión de los recursos necesarios para la funcionalidad del servicio que están usando.

La computación en la nube de carácter público se compone de servidores en internet encargados de atender las peticiones en cualquier momento. Se puede tener acceso a su información o servicio, mediante una conexión a internet desde cualquier dispositivo móvil o fijo ubicado en cualquier lugar. Es habitual que los grandes proveedores de estos servicios los proporcionen desde varios *DataCenters* repartidos por todo el mundo, lo que garantiza un mejor tiempo de actividad, una mejor respuesta al estar más cerca del usuario y ayuda a que los servicios ofrecidos sean menos vulnerables.

Este modelo de prestación de servicios de negocio y tecnología permite al usuario acceder a un catálogo de servicios, estando algunos estandarizados y responder con ellos a las necesidades de su negocio de forma flexible y adaptativa en caso de demandas no previsibles o de picos de trabajo, pagando únicamente por el consumo efectuado. O incluso gratuitamente, en caso de proveedores que se financian mediante publicidad o de organizaciones sin ánimo de lucro.

El cambio que ofrece la computación desde la nube es que permite aumentar el número de servicios basados en la red. Esto genera beneficios tanto para los proveedores, que pueden ofrecer, de forma más rápida y eficiente, un mayor número de servicios, como para los usuarios que tienen

la posibilidad de acceder a ellos, disfrutando de la "transparencia" e inmediatez del sistema y de un modelo de pago por uso.

Para aportar estas ventajas, la computación en la nube se apoya sobre una infraestructura tecnológica dinámica que se caracteriza, entre otros factores, por un alto grado de automatización, una rápida movilización de los recursos, una elevada capacidad de adaptación para atender a una demanda variable, así como virtualización avanzada y un precio flexible en función del consumo realizado, evitando además el uso fraudulento del software y la piratería.

2.1.1. Clasificación

La denominación de *cloud computing* abarca muchos aspectos diferentes, por lo que pueden realizarse distintas clasificaciones dependiendo de la característica que se considere. Tradicionalmente se pueden señalar los siguientes tipos de nubes en base al servicio ofrecido:

- *Software as a Service* (SaaS): es la forma más conocida de *Cloud Computing*. En ella, todas las aplicaciones de software se encuentran en la nube y el usuario suele acceder a ellas mediante un simple navegador web. En la actualidad, hay un gran número de aplicaciones en la nube. Muchas de éstas son aplicaciones web generales muy populares y utilizadas a diario, como redes sociales, correo web o aplicaciones ofimáticas online. Asimismo, también existen aplicaciones específicas de uso empresarial, como `ERP` o `CRM`.

- *Platform as a Service* (PaaS): este tipo de nube constituye un nuevo enfoque para el desarrollo de software ofreciendo la posibilidad de acceder a todas las herramientas de desarrollo de aplicaciones sin instalar nada en el equipo propio. Las principales compañías de software han desarrollado sus propios PaaS, entre las que cabe mencionar Google App Engine, Microsoft Windows Azure y Oracle Cloud.

- *Infraestructure as a Service* (IaaS): este tipo representa la evolución de la infraestructura clásica de servidores físicos en las empresas al sustituirlos por servidores virtuales con ubicación en la propia empresa o Internet. Destaca en este ámbito la implementación comercial Amazon EC2 (Elastic Compute Cloud) y las implementaciones de *software* libre Opennebula y Eucalyptus, que son compatibles con el API de Amazon EC2, pero que permiten un control total sobre la tecnología.

Además de esta primera clasificación, los tipos de nubes se pueden clasificar en función de los usuarios a los que atienden. En base a esto, los tipos son los siguientes:

- Públicos: cuando los servicios ofrecidos se ponen a disposición del público general.

- Privados: cuando se presta el servicio exclusivamente a una única organización, la cual controla las aplicaciones que se ejecutan.

- Híbridos: solución que combina las dos anteriores.

En algunos casos la utilización de IaaS en el mundo empresarial se plantea como una paulatina eliminación de los servidores físicos propios y su sustitución por servidores virtuales ubicados en *DataCenters* remotos. Esta solución redunda de forma inmediata en importantes ahorros de costes, pero no se puede plantear para determinados servicios ya que se asumirían importantes riesgos al no controlar directamente sus equipos y se adquiere con ello una gran dependencia de un proveedor. Actualmente la migración de los recursos entre unas plataformas y otras no suele ser fácil y, a pesar de que esta característica se muestra muy interesante para los usuarios, actualmente no se encuentra muy extendida.

2.1.2. Marco histórico

En el año 1961, John McCarthy, inventor del lenguaje de programación LISP intuyó que un día la computación estará organizada como un servicio publico. Posteriormente, en el año 1969, Leonard Lkeinrock, uno de los científicos a cargo del proyecto Arpanet (Advanced Research Project Agency Netowrk) dijo que "*actualmente las redes de computadoras están en su infancia, pero en la medida en que crezcan y se vuelvan sofisticadas, probablemente veremos el nacimiento de servicio de computación, los cuales, al igual que los servicios de electricidad y teléfono, llegarán a cada casa y oficinas alrededor de todo el país*". Estas visiones se anticipaban a la aparición de nuevos paradigmas de computación fortalecidos por el desarrollo de tecnologías de vanguardia capaces de proveer medidas de desempeño, eficiencia, escalabilidad, distribución, autonomía y ubicuidad, nunca antes vistas. Estos novedosos paradigmas de la computación incluyen: *cluster computing*, *grid computing*, *global computing*, *Internet computing*, *peer-to-peer computing*, *ubiquitous computing*, *utility computing* y mas recientemente *cloud computing*, derivada del termino *cloud*, usado como metáfora de infraestructuras complejas y cuyo origen se remite a la década de los 90, en referencia a las ya enormes redes Asynchronous Transfer Mode (ATM).

En el año 1999, Marc Benioff, Parker Harris y otros socios, fundaron la compañía `salesforce`, aplicando tecnología desarrolladas por compañías como Goole y Yahoo a diversas aplicaciones de negocio. Ellos fortalecieron la entrada de servicio bajo demanda, particularmente SaaS, viéndose respaldado por miles de clientes y negocios exitosos. Al inicio del año 2000, Yahoo, y Google anunciaron la prestación de servicio cloud a cuatro de las más grandes universidades de Estados Unidos: la Universidad de Carnegie Mello, la Universidad de Washington, la Universidad de Stanford y el Massachusetts Institute of Technology (MIT). Poco tiempo después, IBM anunció el ofrecimiento de servicios *cloud*, seguido por gigantes informáticos como Microsoft, Oracle, Intel, SUN, SAS y Adobe, cuyos enfoques abarcaron la provisión de modelos IaaS, PaaS, SaaS. Sin embargo, se considera que el inicio de cloud computing puede ser atribuido a la aparición de los servicios Web de Amazon (Amazon Web Services), que iniciaron su producción en el año 2006 ofreciendo el modelo IaaS con capacidades básicas de procesamiento y almacenamiento a traves de Internet.

Amazon web services popularizó el modelo IaaS, convirtiéndolo en una de las nociones principales de cloud computing. Su novedosa estrategia permitió la ejecución personalizada y bajo demanda de maquinas virtuales `Linux` en infraestructuras computacionales con una complejidad totalmente oculta a los usuarios finales. Esta estrategia minimizó e incluso eliminó los costes para los consumidores de servicios *cloud*, otorgándoles la posibilidad de aumentar o disminuir las capacidades

de su infraestructura computacional para satisfacer los picos o las fluctuaciones en la demanda de servicio IT, pagando únicamente por la capacidad consumida bajo un modelo de facturación basado en tarifas horarias.

2.1.3. Estado del arte

A continuación se describen algunos de los proyectos más destacados en la actualidad para proporcionar *Cloud Computing*:

- OpenStack [Opeg]: proyecto de computación en la nube para proporcionar IAAS. Es un software libre y de código abierto distribuido bajo los términos de la licencia Apache.
- Eucalyptus [Euc]: plataforma *open source* para la implementación de computación en la nube privada. Eucalyptus implementa nubes de tipo privado e híbrido de estilo IAAS.
- CloudStack [apa]: *software open source* de *Cloud Computing* para crear, controlar y desplegar infraestructuras cloud. Originalmente fue desarrollado por `cloud.com` y posteriormente por `Citrix`.
- Nimbus [por]: conjunto de herramientas que permiten adaptar la versatilidad de los entornos de infraestructura como servicio para usuarios científicos.
- OpenNebula [opeb]: conjunto de herramientas *open source* que permiten manejar la infraestructura de virtualización de un centro de datos para construir nubes IAAS privadas, públicas e híbridas.

2.2. OpenStack

A grandes rasgos, `OpenStack` [Opeg] es un *software open source* usado para la construcción de nubes públicas y privadas. `OpenStack` representa tanto a una comunidad y un proyecto de Software Libre, como un software para ayudar a las organizaciónes a ejecutar sus propios entornos de nube para computación o almacenamiento virtual.

Desde el punto de vista *software*, `OpenStack` es una colección de proyectos de *software* libre que incluyen varios componentes, siendo los más importantes los siguientes:

- `OpenStack Compute`, con nombre en clave `Nova`.
- `OpenStack Object Storage`, con nombre en clave `Swift`.
- `OpenStack Network`, con nombre en clave `Neutron`.
- `OpenStack Dashboard`, con nombre en clave `Horizon`.

A través de estos servicios, OpenStack proporciona una completa plataforma operativa para la administración y gestión de entornos de computación en la nube.

Su misión principal es proporcionar un software que cubra el ciclo completo de este tipo de despliegues y que proporcione la posibilidad de desplegar de forma sencilla, escalable, elástica y de cualquier tamaño, tanto nubes públicas como privadas.

Para alguien que se acerca por primera vez a OpenStack, esta aproximación puede resultar difícil. Puesto que OpenStack es un *software* joven y complejo y se trata de un proyecto abierto, existe una gran cantidad de documentación, guías, foros, etc. cuyo objetivo es ayudar en la resolución de los problemas que puedan ir surgiendo durante su utilización. Asimismo, y debido a su juventud, se ha de tener presente que tanto estas fuentes de ayuda como el propio *software* están en constante revisión

2.2.1. Usuarios y proyectos (Tenants)

OpenStack Compute está diseñado para que lo utilicen usuarios muy diversos a los que se les pueden asignar diferentes roles que servirán para controlar las acciones que cada uno de estos usuarios puede realizar. En la configuración por defecto, la mayoría de las acciones no llevan asociadas ningún rol, pero en el administrador del cloud recae la responsabilidad de configurar apropiadamente estas reglas a través de un fichero llamado policy.json. Por ejemplo, se puede definir una regla en este fichero que limite al rol de administrador la solicitud de una dirección IP pública.

2.2.2. Imágenes e instancias

En OpenStack, el concepto de "imagen" hace referencia a las imágenes de los discos, las cuales son plantillas para las máquinas virtuales que se van a crear. El servicios que proporciona las imágenes, Glance, es el responsable de almacenar y gestionar las imágenes en OpenStack.

Por su parte, las instancias son las máquinas virtuales que se ejecutan en los nodos de computación. El encargado de gestionar estas instancias es Nova y, a partir de una determinada imagen, puede lanzarse un número cualquiera de instancias. Cada instancia se ejecuta a partir de una copia de una imagen base, por lo que las modificaciones que se realicen en la instancia no alteran la imagen en la que se basa. Mediante el uso de instantáneas de las instancias, se pueden crear nuevas imágenes que guardan todas las modificaciones realizadas hasta ese momento en la instancia.

Cuando se lanza una instancia, se debe seleccionar un conjunto de recursos virtuales, conocido como sabor (*flavor*). Un sabor define para una instancia el número de CPUs virtuales, la cantidad de memoria, si dispone o no de discos efímeros, etc. OpenStack preinstala una serie de tamaños, que el administrador puede modificar. Asimismo, es importante tener en cuenta que aunque se especifique un tamaño de disco, si la imagen es más grande, el disco se creará del tamaño establecido por la imagen y, por disco efímero se conoce un segundo disco que opcionalmente se puede asociar a la instancia, el cual es de tipo no persistente, lo que implica que se destruirá con la instancia.

2.3. Virtualización

La virtualización es el proceso de creación de máquinas virtuales, donde una máquina virtual es una instancia de sistema operativo y dispositivos que opera sobre software y no sobre *hardware*. Esta tecnología proporciona la capacidad de ejecutar varias instancias de sistema operativo en una misma máquina física, particionando y compartiendo los recursos de esta de forma dinámica. El concepto no es nuevo: proviene de los años 60, cuando `IBM` propuso el concepto de virtualización como la capacidad de compartir el acceso a recursos *hardware* costosos y posteriormente explotado en los sistemas VM/370 [Cre81].

Virtual Machine Monitor (VMM), conocido también como hypervisor es la capa software de abstracción en la que las máquinas virtuales operan. Se puede encontrar en dos versiones, una en la que esta capa trabaja en conjunción con el sistema operativo del equipo y otra en la que trabaja directamente sobre el *hardware* del equipo. Las dos opciones anteriores se conocen como hypervisor de tipo 2 y de tipo 1 respectivamente y se representan en la ilustración 2.1.

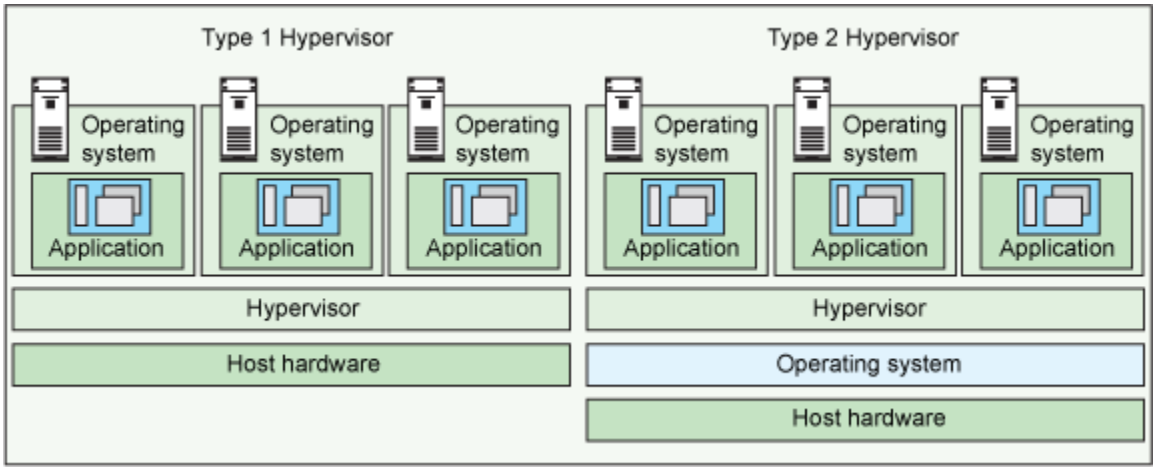


Ilustración 2.1: Representación de los tipos de hypervisores. Fuente: IBM.

La clasificación anterior se basa en quién es capaz de detectar la ejecución de instrucciones ISA conocidas como sensitive instructions. A continuación se exponen dos conceptos previos para poder entender estas instrucciones:

- *Current Privilege Level* (CPL) [insb]: En general, el valor 0 indica *modo privilegiado* mientras que cualquier otro valor indica *modo no-privilegiado*. Esta diferencia se realiza porque ciertas instrucciones (conocidas como CPL sensitive) sólo pueden ser ejecutadas en modo privilegiado, como por ejemplo `HLT`, `LIDT`, `LGTD` y `LMSW`.
- *I/O Privilege Level* (IOPL) [insa]: Este valor indica el valor mínimo de `CPL` requerido para ejecutar cierta instrucción privilegiada de entrada/salida. Estas instrucciones privilegiadas se conocen como *IOPL sensitive* y son, por ejemplo, `IN`, `INS`, `OUT`, `OUTS`, `CLI` y `STI`.

Tras la breve descripción de estos conceptos, y volviendo a las *sensitive instructions*, éstas son instrucciones protegidas porque proveen control sobre los recursos *hardware*, por lo que no pueden

ser ejecutadas directamente por las máquinas virtuales. En base a lo mencionado anteriormente, un hypervisor necesita el control y administra la ejecución de todas estas instrucciones sensibles. Un hypervisor del tipo 1 como XEN detecta él mismo las instrucciones sensibles, pero un hypervisor del tipo 2 depende de otro software (un S.O tradicional) para detectarlas y notificar al hypervisor.

Respecto a los tipos de virtualización también son dos; paravirtualización y virtualización completa. La primera de ellas requiere modificaciones en el sistema a correr en las máquinas virtuales mientras que en la segunda no. Además, la virtualización completa se puede implementar de dos formas conocidas como traducción binaria o asistida por *hardware*, las cuales se describen a continuación:

- La virtualización completa mediante traducción binaria implica que las instrucciones virtuales (en ocasiones sólo las no virtualizables) se traducen en una secuencia de instrucciones a ejecutar en el procesador real que consigue el efecto buscado en la máquina virtual. Como contrapunto, este proceso de traducción penaliza el rendimiento.
- La virtualización completa asistida por hardware elimina la necesidad de traducir y de modificar el sistema de las máquinas virtuales, lo que permite que las instrucciones de las máquinas virtuales se ejecuten directamente en el procesador real. La limitación de esta tecnología es que las instrucciones de las máquinas virtuales tienen que ser compatibles con el conjunto de instrucciones del procesador real.

En caso de que se desee conocer información adicional sobre virtualización de sistemas x86, esta se puede encontrar en [Hor07].

2.4. Alta disponibilidad

En el mundo empresarial existen muchas aplicaciones que, dada su naturaleza crítica, deben proporcionar un servicio ininterrumpido de 24 horas al día, 7 días a la semana. Para conseguir estos niveles de disponibilidad se utiliza una configuración avanzada de hardware y/o software denominada en su conjunto *cluster* de alta disponibilidad.

Si un servicio presenta de alta disponibilidad y, por cualquier motivo falla, las consecuencias serán mínimas ya que el servicio seguirá funcionando de forma transparente para los usuarios como si el fallo nunca hubiera ocurrido aunque puede haber una penalización en el rendimiento.

Esta característica de los sistemas se puede implementar mediante una configuración basada en *hardware* o en *software*. La configuración *hardware* trata de asegurar que el servidor funcione de forma interrumpida haciendo uso de un sistema redundante de fuentes de alimentación, de discos duros (RAID), tarjetas de red, etc. De esta forma, si falla cualquiera de esos elementos *hardware* el sistema funcionará correctamente y lo único que será necesario hacer es reemplazar el dispositivo defectuoso, en “caliente” si es posible, o programar una parada de mantenimiento. Por su parte, una configuración basada en *software* consiste en una serie de ordenadores, denominados nodos, que se conectan entre sí de tal manera que ante un fallo *hardware*/*software* el servicio o servicios ofrecidos por el nodo fallido son relanzados por otro nodo del *cluster*. De esta manera, el servicio que se ofrece sigue en funcionamiento de manera casi ininterrumpida.

A modo de resumen, en un entorno empresarial las soluciones de alta disponibilidad ofrecen tolerancia a fallos, flexibilidad y tranquilidad a la empresa.

2.4.1. Proyectos de alta disponibilidad

En entornos Linux existen una gran variedad de proyectos que aportan características de alta disponibilidad, algunos de los cuales se presentan a continuación:

- The High Availability Linux Project [linb]: proporciona una solución de alta disponibilidad ofreciendo fiabilidad y disponibilidad. Es una de las soluciones más ampliamente extendidas. Se integra perfectamente con multitud de servicios y sigue aún mejorando dicha integración gracias a la aportación de su activa comunidad.
- Red Hat Cluster Suite [RHECS13]: diseñado específicamente para Red Hat Enterprise Linux. Dispone de soluciones listas para implantar para la mayoría de los servicios, pudiendo además proporcionar soporte cuando sea necesario por parte de RedHat.
- KeepAlived [kee]: su objetivo principal es proporcionar alta disponibilidad al proyecto Linux Virtual Server (LVS). KeepAlived es un servicio que monitoriza el estado del *cluster* LVS para que en caso de fallo el *cluster* siga en funcionamiento.

De estos tres proyectos descritos, se optó por Red Hat Cluster Suite, el cual se describirá en las líneas siguientes. Este software ofrece diferentes opciones para gestionar la preferencia en dominios de fallo, las cuales se presentan a continuación:

- Unrestricted: permite establecer miembros predefinidos para los servicios, pudiendo estos activarse en cualquier equipo disponible.
- Restricted: permite establecer los miembros que pueden activar un servicio. En el caso de *clusters* con bastantes nodos esta opción es adecuada para dar menos trabajo al software.
- Unordered: en este caso, los equipos no tienen orden a la hora de levantar un servicio.
- Ordered: permite especificar la preferencia, pudiendo variar la prioridad desde 1 (más prioritario) a 100 (la menor).
- Failback: establece que un servicio vuelve a su servidor de origen cuando este se recupere de una caída. Esta opción sólo es aplicable si se configura como ordered.

Cuando se define un servicio, las posibles politicas de recuperacion, entendiendo esta como la forma en la que el servicio se vuelve a poner en funcionamiento tras una caída, son:

- Restart: intenta reiniciar los recursos antes de recolocarlo tantas veces como se especifique.
- Relocate: migra el servicio directamente.
- Disable: deshabilita el servicio si algo falla.
- Restart-disable: intenta reiniciar el servicio y, si no es posible, lo deshabilita.

2.4.2. Cerebro dividido

Uno de los problemas con los que tienen que tratar este conjunto de técnicas es el conocido como "cerebro dividido". Este problema se presenta cuando un subconjunto de equipos que conforman un *cluster* comienza a operar de manera autónoma. Es decir, ocurre cuando en un *cluster* se generan "particiones" y cada subconjunto no tiene conocimiento del otro.

Cuando se producen particiones, puede ser que los nodos de cada partición estén inactivos, pero también puede suceder que estén incomunicados, por lo que lo único que se puede decir del otro nodo es que su estado es desconocido. A pesar de que está situación no ocurre con frecuencia en la práctica, es vital que un *cluster* disponga de algunas soluciones para esta situación.

Existen diferentes formas de solucionar este problema, las cuales no son incompatibles entre sí. Estas se muestran en las siguientes subsecciones.

Fencing

Se denomina *fencing* al proceso de bloquear los recursos de un nodo cuyo estado es incierto. Hay varias técnicas disponibles para realizar *fencing*, las cuales son presentadas a continuación:

- `NodeFencing`: Permite bloquear o separar un nodo del *cluster* de todos los recursos que maneja de una vez sin la cooperación o aprobación por parte del nodo, esto es, lo hace forzosamente. Para esta técnica de *fencing*, `RedHat Cluster` hace uso de `Stonith`. `Stonith` realiza esto de una manera muy especial que requiere de dispositivos externos adicionales, como los `BMC` de los servidores por `IPMI` o PDUs administrables. Se trata de hacer que el dispositivo adicional reinicie o apague un nodo en caso de fallo.

- `ResourceFencing`: Consiste en permitir o negar la funcionalidad de un recurso. Por ejemplo, si a una unidad SAN se accede a través de un conmutador, se podría indicar al conmutador que rechace la comunicación del nodo caído con el recurso SAN, con la finalidad de que no pueda hacer uso de él. La implementación de `ResourceFencing` varía en función del recurso y de como el acceso a este puede ser permitido o denegado. A grandes rasgos, es la misma idea que `NodeFencing` con la diferencia que el *fencing* en este caso se realiza sobre un recurso en lugar de sobre un nodo.

- `SelfFencingResourcer`: Consiste en que es el propio recurso el que automáticamente garantiza el acceso exclusivo. Es la misma idea que ResourceFencing con la diferencia que en este caso es el propio recurso el que se hace *fencing* a si mismo, sin requerir la intervención de terceras partes.

Aunque suele ser la primera medida a configurar, en algunas situaciones no es suficiente (por ejemplo, si hay dos nodos, las dos partes se intentarían bloquear a sí mismas si se aíslan). Para solventar este problema, existe la opción `Quorum` descrita en la siguiente subsección.

Quorum

La solución que plantea `Quorum` es la de no seleccionar más de una "partición del *cluster*" cuando falla la comunicación en el *cluster*, es decir, que sólo una partición del *cluster* ofrezca los servicios. El método más conocido para hacer `Quorum` en un *cluster* de n-nodos es dar "el quorum" a la partición que tenga más de n/2 nodos en ella. En caso de que el número de nodos sea 2, el cerebro divido causará el apagado del *cluster*.

Para resolver este caso particular, y como se comentará en secciones posteriores, existen técnicas para sumar algún voto más al algoritmo mediante, por ejemplo, un disco de `Quorum` o una heurística. Estas técnicas se comportan como una especie de árbitro que puede ser tanto software como *hardware*.

Alternativas a Fencing

En algunas ocasiones, hacer uso de las técnicas de *fencing* vistas previamente no es posible. Por ejemplo, utilizar `Stonith` requiere el uso de un dispositivo *hardware* o *software* adicional y no siempre existe esa posibilidad. Para estos casos se dispone de alternativas a *fencing*, conocidas como *self-fencing*, las cuales son mostradas a continuación:

- *Node suicide*: si un nodo pierde el `quorum`, este puede y debe apagarse o reiniciarse él mismo (es una especie de auto-stonith). El problema que plantea esta solución es que un nodo en mal funcionamiento, por ejemplo que se encuentre saturado, es posible que no pueda reiniciarse o apagarse o que ni siquiera sea consciente de que ha perdido el `quorum`. A diferencia de otras soluciones vistas anteriormente, como *Node fencing*, en *Node suicide* se requiere la colaboración del propio nodo. Un retraso en este auto-suicidio presentaría problemas para el *cluster*.

- *Self-shutdown*: Esta técnica de auto-fencing es una variante de la técnica que se acaba de comentar, con la diferencia de que los recursos del nodo son previamente parados. Presenta los mismos problemas que la solución anterior, incluso agravados ya que hay que esperar un cierto tiempo para liberar los recursos. Esta técnica puede realizarse con un software *Watchdog* como `Softdog`, de tal manera que cuando el nodo pierde el `quorum` se reinicia.

Estas alternativas son menos fiables ya que no se depende de terceras partes para realizar el fencing, sino que se depende del propio nodo el cual además podría no estar funcionando correctamente por motivos tales como saturación.

2.4.3. Red

En el apartado de red lo habitual es hacer redundantes los enlaces y si además se quiere aprovechar estos recursos adicionales se usan técnicas de agregado como "channel bonding" en las que dos o mas interfaces de red se combinan para ofrecer redundancia y/o incrementar el rendimiento.

A continuación se presenta los modos de *bonding* que soporta `Linux`:

- *balance-rr* o 0: política *round-robin*, es decir, transmite los paquetes en orden secuencial entre las diferentes interfaces disponibles. De esta forma se consigue balanceo de carga y tolerancia a fallos.
- *active-backup* o 1: política activo-pasivo, es decir, sólo una tarjeta está activa en cada momento pero, sin embargo, si la activa falla, una de las pasivas se convierte en activa.
- *balance-xor* o 2: política `XOR`, es decir, transmite los paquetes basándose en un *hash* elegido por defecto: [(MAC origen XOR MAC destino) modulo cantidad de tarjetas]. Políticas alternativas pueden ser configuradas fijando el parámetro `xmit_hash_policy`. Este modo proporciona balanceo de carga y tolerancia a fallos.
- *broadcast* o 3: política de difusión, es decir, el tráfico es transmitido por todas las interfaces. De esta forma se proporciona tolerancia a fallos.
- `802.3ad` o 4: `IEEE 802.3ad Dynamic link aggregation`, se crean grupos de agregación que comparten los mismos parámetros de velocidad y *duplex*. Se utilizan todas las tarjetas disponibles en el agregado de acuerdo a la especificación 802.3ad. La mayoría de los *switchs* necesitan alguna configuración para trabajar con estos agregados, se suele denominar como *Link Aggregation Control Protocol* (LACP) en las configuraciones. Esta opción proporciona balanceo de carga y tolerancia a fallos.
- *balance-tlb* o 5: balanceo de carga adaptativo en la transmisión, es decir, "Channel bonding" que no requiere soporte adicional por parte del *switch* al que se conecta el equipo. El tráfico saliente es distribuido de acuerdo a la carga de cada tarjeta. Esta opción proporciona balanceo de carga y tolerancia a fallos.
- *balance-alb* o 6: balanceo de carga adaptativo, es decir, incluye el modo 5 y además balanceo en la recepción para tráfico IPv4. No requiere de ninguna configuración adicional en el *switch* al que está conectado el equipo. Esta opción soporta balanceo de carga y tolerancia a fallos.

2.4.4. Alta disponibilidad aplicada a OpenStack

`OpenStack` es una conjunción de herramientas que juntas proporcionan un sistema completo. Esta diversidad de herramientas facilita la flexibilidad en los diseños y permite realizar múltiples configuraciones. Para asegurar la disponibilidad del sistema completo, hay que garantizar la de cada servicio de los que se compone. Los servicios de `OpenStack` pueden ser agrupados en base al enfoque de alta disponibilidad que tienen en los siguientes grupos:

- Servicios *Application programming interface* (API): son servicios HTTP/REST, fáciles de replicar con un balanceador de carga y, si además el balanceador cuenta con la posibilidad de realizar *tests* de los servicios, se tiene también alta disponibilidad. Debido a la versión de `OpenStack` con la que se trabaja, sólo la API de `swift` soporta una llamada especifica para comprobar el estado del servicio, por lo que bastará que responda para asumir que cada servicio trabaja correctamente.

- Servicios de cómputo (nova-compute, nova-network y cincer-volume): estos servicios no requieren de una redundancia especifica aunque si escalabilidad, sobre todo el servicio de red, en el que la alta disponibilidad se ofrece mediante la característica `multi-host`.
- Planificadores: los planificadores están construidos pensando en la redundancia. Cuando se ejecuta la primera instancia, esta consume mensajes de la cola del servidor `RabbitMQ` y se crea una cola en base al id de la instancia. Las sucesivas instancias se comportan igual, por lo que pueden trabajar en paralelo.

En cuanto al servidor de colas `RabbitMQ` mencionado en el párrafo anterior es el bus de comunicaciones principal para todos los servicios, por lo que es importante que esté accesible siempre. De forma nativa `RabbitMQ` permite formar un *cluster* replicando las colas de mensajes en otro equipo y, mediante un balanceador de carga se pueden distribuir las conexiones entre los diferentes servidores `RabbitMQ` que trabajan en el *cluster*.

Respecto a la base de datos, la opción más usada es `MySQL`. Para proporcionar características de alta disponibilidad a este gestor de bases de datos hay varias soluciones. De entre todas, la más habitual es `MySQL-mmm` (multi master replication manager), aunque también se usa bastante el motor de *clustering* de Galera.

Una vez que se sabe como balancear o paralelizar la carga de trabajo se necesita un mecanismo que permita añadir nodos y expandir la capacidad de trabajo, lo que se denomina “escalabilidad horizontal”. Para la mayoría de los servicios de `OpenStack` esto se realiza añadiendo una instancia del servicio e introduciendo ésta en el balanceador de la plataforma.

2.4.5. Tolerancia a fallos

Cuando se trabaja con diversos sistemas y herramientas pueden ocurrir multitud de fallos, algunos predecibles y otros impredecibles. Con un gran nivel de abstracción, se pueden clasificar en base a su impacto en los siguientes:

- Falla la instancia de un servicio: se produce cuando falla un demonio pero el resto de servicios del servidor siguen funcionando con normalidad.
- Falla un servidor completo: se produce cuando falla un servidor completo porque se ha producido un fallo de *hardware* o porque ha fallado el sistema operativo.

A la hora de tratar los fallos descritos hay tres niveles de tolerancia que cuentan con diferentes garantías y complejidad de implementación:

1. El fallo de un componente no provoca la interrupción permanente del servicio.
2. El fallo de un componente no provoca el fallo de nuevas peticiones.
3. El fallo de un componente no provoca el fallo de ninguna petición (ni nuevas ni existentes).

A lo largo del presente documento, se hará referencia a los niveles mostrados previamente para mostrar el nivel de disponibilidad ofrecido por los servicios desplegados.

2.5. Servicios web RESTful

La *Representational State Transfer* (REST) ha ganado amplia aceptación en toda la web como una alternativa más simple a SOAP y a los servicios web basados en WSDL. Este tipo de representación es la usada por los servicios de OpenStack[1].

REST define un conjunto de principios arquitectónicos por los cuales se diseñan servicios web haciendo énfasis en los recursos del sistema, incluyendo aspectos como, por ejemplo, cómo acceder al estado de dichos recursos y cómo transferirlos por HTTP hacia clientes escritos en diversos lenguajes.

Un servicio web REST sigue cuatro principios de diseño fundamentales:

- Utiliza los métodos HTTP de manera explícita
- No mantiene estado
- Expone URIs con forma de directorios
- Transfiere XML, JavaScript Object Notation (JSON), o ambos

[1] En [Opec] se puede encontrar información de las APIs de los servicios de OpenStack.

3 Requisitos

> La especificación de requisitos es una tarea importante en cualquier proyecto puesto que se trata de una descripción completa del comportamiento del sistema. Incluye un conjunto de requisitos funcionales y no funcionales, los cuales, respectivamente, exponen las necesidades del usuario e imponen restricciones en el diseño o la implantación. Se define requisito como una condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado [Vel96].

A continuación se describen brevemente los requisitos del sistema, clasificándolos en funcionales y no funcionales. Posteriormente a estos, se muestran las necesidades de dimensionamiento que se le han impuesto al sistema. Asimismo, es importante mencionar que tanto los requisitos como el dimensionamiento han sido impuestos por la empresa en la que se ha realizado el proyecto.

3.1. Requisitos funcionales

El sistema debe contemplar las siguientes necesidades funcionales:

1. Se debe permitir la gestión de plantillas, para poder ofrecer máquinas virtuales con *software* y configuraciones específicas o con el sistema operativo recién instalado.
2. El sistema debe contemplar la definición de tallas de las máquinas virtuales a crear, en base al número de CPUs virtuales, a la cantidad de memoria y al tamaño del disco duro.
3. Se debe permitir la creación de las máquinas virtuales que solicite un usuario en base a la plantilla seleccionada y a la talla elegida, así como su posterior pausa o destrucción.
4. El sistema debe permitir la gestión de los recursos disponibles para cada proyecto, es decir, la posibilidad de establecer cuotas al servicio prestado por el sistema. Estas cuotas se tienen que poder particularizar si algún proyecto así lo necesita.
5. Se debe poder definir la definición de la visibilidad de las máquinas virtuales en la red corporativa y los puertos de éstas a los que se puede acceder. Además, con la finalidad de ofrecer demostraciones a clientes, se deberá dar la posibilidad de publicar las máquinas en internet mediante el cortafuegos de la compañía.
6. El sistema debe ofrecer la creación y destrucción bajo demanda de discos duros virtuales y la conexión de éstos a las máquinas virtuales previamente instanciadas.

7. Se debe permitir la gestión de usuarios y proyectos.
8. El sistema debe contemplar la contabilidad del uso de los recursos realizada por los proyectos.

3.2. Requisitos no funcionales

El sistema tiene que contemplar los siguientes requisitos no funcionales:

1. Las redes de diferentes proyectos deben estar aisladas, es decir, dos máquinas de diferentes proyectos no se tienen que poder intercambiar información salvo que así se establezca.
2. El almacenamiento se tiene que gestionar usando *internet Small Computer System Interface* (iSCSI) dado que es la tecnología usada habitualmente en la compañía debido a su relación calidad/precio.
3. El sistema tiene que diseñarse para asumir la carga actual de proyectos de la empresa con necesidades computacionales y el crecimiento esperado a 3 años vista, aunque no sea factible esta carga sobre el equipamiento del que se dispone en la actualidad.
4. El equipamiento disponible permite desplegar un prototipo del sistema pero, igualmente, tiene que diseñarse como un sistema "estable".
5. La gestión de usuarios y proyectos se debe integrar con el directorio activo de la compañía.
6. El sistema debe monitorizarse con las herramientas dedicadas a tales efectos en la empresa, actualmente Zabbix [SIA].
7. Salvo que haya alguna justificación en contra, el sistema operativo que se debe instalar en las máquinas es `CentOS`[1].

3.3. Dimensionamiento

Uno de los aspectos importantes del proyecto es el dimensionamiento del mismo. Para poder abordar este apartado se analizaron las máquinas virtuales desplegadas actualmente en el sistema de virtualización de la compañía y se solicitó una previsión de futuro al departamento de "Gestión Interna" en cuanto a la evolución en el número de proyectos.

El análisis realizado arrojó una cifra aproximada de 70 proyectos con necesidades computacionales[2] y las estimaciones facilitadas por dicho departamento son las mostradas en la tabla 3.1.

Asimismo, el análisis realizado desvela que la mayoría de los proyectos cuentan con una o dos máquinas pero, sin embargo, un pequeño porcentaje de ellos se salen de esta norma. Para adaptar el diseño del sistema a estos datos se proponen diferentes tipos de proyectos (véase tabla 3.2).

[1] Distribución `Linux` de tipo empresarial derivada del código fuente de `Red Hat Enterprise Linux` - `www.centos.org`.
[2] No todos los proyectos de la compañía tienen esa necesidad. Por ejemplo, en algunos proyectos se usa la infraestructura y los equipos proporcionados por el cliente.

Año	Tasa de creación	Tasa de destrucción	Tasa de crecimiento
2014	15 %	10 %	5 %
2015	15 %	10 %	5 %
2016	20 %	10 %	10 %

Tabla 3.1: Estimaciones de la evolución del número de proyectos.

Tipo	Número de IPs privadas	Número de IPs públicas
ServicioA	5	5
ServicioB	20	10
ServicioC	50	35
ServicioD	100	70
ServicioE	200	0

Tabla 3.2: Tipos de proyectos y número de direcciones IP.

Teniendo en cuenta los datos del estudio y los tipos de proyectos, se acuerda con el responsable del área de operaciones que de forma mínima se necesitan los siguientes proyectos: 86 proyectos del "servicioA", 8 del "servicioB", 4 del "servicioC", 2 del "servicioD" y uno creado ad-hoc del "servicioE" para grandes pruebas.

El "servicioE" no necesita direcciones públicas debido a que su función es la de albergar pruebas en las que se puedan levantar una gran cantidad de máquinas que no van a ofrecer servicio alguno.

4 Infraestructura

Si bien es cierto que la parte llamativa del sistema a desplegar en este proyecto es la percibida por el usuario final, la capa de infraestructura es la base del sistema, por lo que también merece atención. A lo largo del presente capítulo se va a exponer su diseño, resaltando su importancia y características como parte del sistema completo.

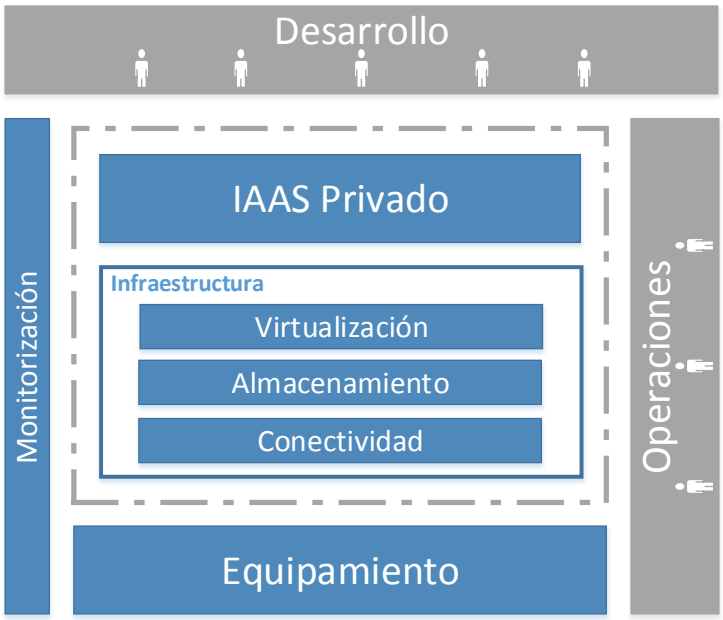


Ilustración 4.1: Diagrama de módulos resaltando y expandiendo la capa de infraestructura.

El sistema IAAS de computación en la nube a implantar en este proyecto se despliega sobre una infraestructura. Dicha infraestructura tiene que proporcionar todos los servicios que necesita la capa superior y, además, ofrecer un cierto grado de disponibilidad para que el servicio se vea afectado lo menos posible por un fallo del equipamiento. Para abordar su diseño y construcción se ha dividido en módulos, tal y como se puede apreciar en la ilustración 4.1.

El módulo de “conectividad” se encarga de gestionar la comunicación entre los diferentes elementos del sistema. Utilizando esa comunicación, el módulo de “almacenamiento” ofrece al resto de componentes los recursos de almacenamiento disponibles y, además permite una gestión dinámica de los mismos. Usando los anteriores, el módulo de “virtualización” flexibiliza el uso del equipamiento dedicado a computación y soporta las operaciones con máquinas virtuales de la capa IAAS.

En el diagrama expuesto previamente se ha incluido un bloque de “monitorización” que se encarga de satisfacer el requisito no funcional 6, afectando también éste a la capa de infraestructura. Por otra parte, los módulos que la componen se han diseñado teniendo en cuenta los requisitos no funcionales 3, 4 y 7 de forma general, cumpliendo también otros requisitos más específicos.

Durante el diseño de los módulos de la capa de infraestructura se ha tenido en cuenta su disponibilidad, orientándose dicho diseño a ofrecer la mayor disponibilidad posible teniendo en cuenta el equipamiento con el que se cuenta.

4.1. Equipamiento

Los equipos sobre los que se van a desplegar los módulos de la capa de infraestructura se representan en la ilustración 4.2 y se exponen a continuación:

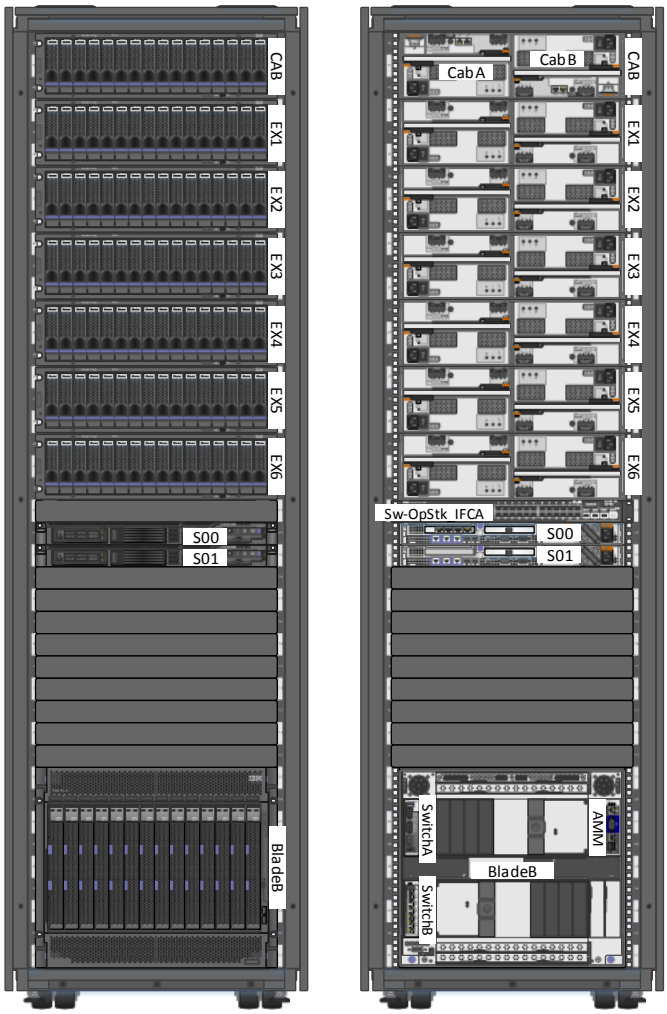
- Un armario IBM de 42U.
- Una cabina IBM DS4700 (“CAB”) con dos controladoras (“CabA” y “CabB”) y 16 discos *Serial Advance Technology Attachment* (SATA) de 750 GB.
- Seis expansores IBM EXP810 (“EX1”..“EX6”) con 16 discos SATA de 750 GB cada uno.
- Un IMB BladeCenter H (“BladeB”) con dos *switchs* (“switchA” y “switchB”) y 14 *blades* IBM HS21XM con 2 Xeon E5420 y 16 GB de RAM cada uno.
- Dos servidores IBM x3550 (“S00” y “S01”).
- Un *switch* Dell 5424 (“Sw-OpStk_IFCA”).

Como se ha mencionado previamente (véase la página 2) parte del equipamiento está compartido con el IFCA. Esto implica que del almacenamiento visible en “S00” y “S01” se puede disponer de 12 Logical Unit Numbers (LUNs) (`/dev/sdl` a `/dev/sdw`) y dos puertos del *switch* “SwitchB” se encuentran ocupados. Además, las configuraciones realizadas sobre estos equipos han mantenido la funcionalidad existente añadiendo las características necesarias para el sistema a implantar en este proyecto. Por otra parte, cuatro de los *blades* se encuentran apagados por rotura.

La cabina junto con los expansores representan los recursos de almacenamiento en bruto. Cada controladora de la misma se conecta mediante Fibre Channel (FC) a uno de los servidores “S00” y “S01” que exportan el almacenamiento al resto del sistema. Por su parte, los 10 *blades* representan los recursos de computación y el *switch* “Sw-OpStk_IFCA” junto con el “switchA” y “switchB” del BladeCenter forman los recursos de red.

4.2. Conectividad

Este módulo se encarga de proveer a los demás componentes de la infraestructura y a la capa superior la conectividad de red necesaria para realizar sus funciones. Se ha prestado especial interés en su diseño debido a la importancia que tiene para el sistema y la dificultad que supondrían posibles futuras modificaciones no planeadas previamente.



(a) Frontal (b) Parte posterior

Ilustración 4.2: Diagrama del armario y el equipamiento que contiene.

Para este módulo se emplea una única red *ethernet* dividida en múltiples redes IP (IPv4). Cada una de estas redes se usa para diferentes funciones del sistema, así como para las redes de los proyectos desplegados en el sistema. Para aislar estas redes se utilizan Virtual LANs (VLANs).

4.2.1. Red de proyectos

Cada una de las instancias creadas por el sistema IAAS se conecta a la red privada asignada al proyecto al que pertenece. OpenStack es flexible en cuanto al diseño de red en general y, en concreto, para esta red ofrece tres opciones de asignación de IPs implementadas en tres implementadas en tres Network Manager [Ope12] descritos a continuación:

- `Flat Network Manager`: Cuando una instancia es creada, el sistema reserva una `IP` del *pool* de direcciones disponibles (fijado como una subred) y la inyecta en la imagen al arrancar. Esta "inyección" funciona en sistemas que guardan la configuración en `/etc/network/interfaces`.
- `Flat DHCP Network Manager`: En este modo se configura un servidor `DHCP` para todos los proyectos que se encarga de dar `IP` (de un rango fijado como una subred) a las instancias.
- `VLAN Network Manager`: Se configura un servidor `DHCP` para cada proyecto que se encarga de facilitar direcciones del rango fijado en la `VLAN` correspondiente. Las subredes a utilizar se definen inicialmente y se asignan a los proyectos dinámicamente. Para instalaciones con más de un equipo, este modo requiere un *switch* que soporte etiquetado `VLAN` (802.1q) [IEE11].

Un gestor de red puede trabajar en dos modos: `single-host` o `multi-host`, lo que influye en gran medida sobre la configuración de red. En el modo `single-host` existe únicamente una instancia de `nova-network` la cual se utiliza como *gateway* por defecto para las máquinas virtuales y se comporta según lo mostrado anteriormente en cuanto a `DHCP`. Por su parte, en el modo *multi-host* todos los nodos de computación ejecutan su propia instancia de `nova-network` y, se configura uno (opción 2 de las anteriores) o varios (opción 3 de las anteriores) servidores `DHCP` en cada equipo.

Debido a la restricción impuesta en los requisitos de que las máquinas de diferentes proyectos deben aislarse salvo que se hagan públicas (requisito no funcional 1), se ha de optar por el manejador de red basado en redes virtuales (`VLAN Network Manager`). Este modo de red impone un límite al número de proyectos debido a que el campo de la etiqueta tiene 12 bits de longitud, pero en este proyecto, tal y como se muestra a continuación, no se llega a ese límite.

Teniendo en cuenta el requisito no funcional 3 y que el diseño de red es un apartado importante, a los datos mostrados en la sección 3.3 sobre dimensionamiento se les aplica adicionalmente un 30 % de crecimiento (habitual en el diseño de infraestructuras de red [DAV13]) y se redondea el número de proyectos a la siguiente potencia de dos para obtener unos direccionamientos de red factibles. Por lo que finalmente se necesita planificar 128 proyectos del "servicioA", 16 del "servicioB", 8 del "servicioC", 4 del "servicioD" y 2 del "servicioE" que se plasman en las siguientes redes.

Nombre RED	Subred	Cantidad	Tamaño	VLAN inicial	VLAN final
OpStk_servA	10.62.16.0/21	128	16	1000	1127
OpStk_servB	10.62.24.0/23	16	32	1200	1215
OpStk_servC	10.62.26.0/23	8	64	1300	1307
OpStk_servD	10.62.28.0/23	4	128	1400	1403
OpStk_servE	10.62.30.0/23	2	256	1500	1501

Tabla 4.1: Divisiones de la red de proyectos 10.62.16.0/20.

En base a los datos de partida la mayoría de proyectos de la compañía harán uso del tipo A, pero en algunos momentos 5 máquinas no serán suficientes, por lo que se hará uso de alguno de los otros tipos de proyecto. Estas necesidades fuera de lo habitual se comunicarán al área de operaciones para que habilite un proyecto del tipo adecuado.

4.2.2. Red pública

Como ya se ha visto, a cada máquina virtual se le asigna de forma automática una dirección IP privada. Opcionalmente se les puede asignar una dirección pública que OpenStack denomina "flotante". La "visibilidad pública" es un concepto relativo puesto que para nubes públicas serán direcciones visibles en internet y para entornos privados serán direcciones visibles desde la red corporativa.

Las direcciones flotantes pueden asignarse de forma dinámica a cualquier máquina virtual y posteriormente liberarse. Para que estas direcciones puedan ser usadas por las MVs OpenStack hace *Network Address Translation* (NAT) sobre ellas (véase ilustración 4.3).

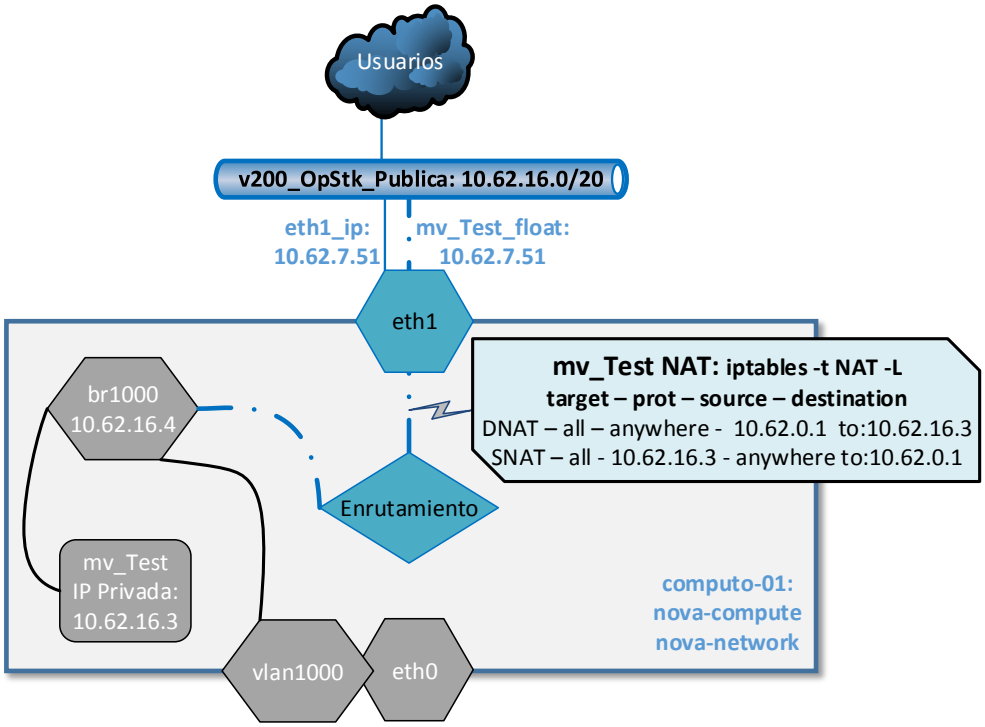


Ilustración 4.3: Ejemplo de MV con IP flotante y su NAT.

Teniendo en cuenta el número de direcciones con las que cuenta cada tipo de proyecto (véase tabla 3.2) y el número de proyectos planificados finalmente se requieren 1360[1] direcciones flotantes. Para obtener un número de IPs representable por una subred con *Variable Lenght Subnet Mask* (VLSM) se redondea a la siguiente potencia de dos, que es 2048. La red pública queda definida por el rango 10.62.0.0/20, con la etiqueta de VLAN 200 y con el nombre "v200_OpStk_Publica".

Las máquinas de la capa IAAS que se exponen a los usuarios y las máquinas de computación con las instancias de nova-network también van a contar con una IP en esta red, para ello se borran del *pool* de direcciones flotantes las direcciones entre la 10.62.7.1 y 10.62.7.254.

[1] $128 * 5 + 16 * 10 + 8 * 35 + 4 * 70 + 2 * 0 = 1360$

4.2.3. Otras redes

Como ya se mencionó en la introducción del módulo de conectividad, se usan varias redes en el sistema para diferentes propósitos. Habiéndose tratado previamente las redes de proyectos y la pública, en esta sección se aborda el resto.

Para tareas de gestión y monitorización del *hardware* se fija la red "v136_gestionOpStk-IFCA" (172.27.5.0/24) a la que se van a conectar todos los equipos del entorno con la capacidad de ser administrados. Para trabajos de control y para mantener la comunicación entre los diferentes componentes de `OpenStack` se establece la red "v208_Control" (10.62.8.0/24). Por su parte, la red "v210_HeartBeat" (10.62.10.0/24) es utilizada por los sistemas en alta disponibilidad para comunicarse. Para la conexión entre los servicios que necesitan almacenamiento y los equipos que lo proporcionan en cada capa se ha creado otra red denominada "v209_Almacenamiento" (10.62.9.0/24). Por último, se ha creado otra para monitorización de los equipos denominada "v211_Monitorizacion" (10.62.11.0/24), siempre y cuando estos no estén gestionados a través de la red de gestión, en cuyo caso se les monitoriza a través de esa red como ya se ha mencionado.

Estas redes se diseñan bajo la estimación de que el entorno contará como máximo con 30 servidores, sin embargo, se opta por reservar redes de clase C por simplicidad. Además, se reservan cuatro rangos de clase C para futuras funciones (10.62.$\{$12,13,14,15$\}$.0/24).

4.2.4. Calidad de servicio

Quality of Service (QoS) es un conjunto de tecnologías que permiten aplicar un tratamiento específico a un determinado tipo de tráfico, como priorizarlo o garantizarle un ancho de banda mínimo. A grandes rasgos, el proceso que se realiza se puede dividir en dos tareas: marcado y clasificación. Por un lado, el marcado de los paquetes consiste en fijar a cada uno de ellos su tipo de tráfico y se puede realizar a nivel *ethernet* con el campo Priority Code Point (PCP), usado en este sistema o a nivel 3 con el campo Type of Service (ToS). Por otro lado, por cada puerto se definen múltiples colas FIFO, la clasificación consiste en examinar cada paquete e introducirlo en la cola que le corresponda para ser transmitido. Este encolado de los paquetes también se puede realizar en los sistemas operativos, gestionándose en sistemas `Linux` mediante el comando `tc` [Hub01].

Para aplicar calidad de servicio en una red lo primero que se define es el tráfico considerado importante. En el caso de este sistema, se ha determinado que el tráfico más importante sea el de la red "v210_HeartBeat" y "v209_Almacenamiento". Esto se ha considerado así porque, respectivamente, los *clusters* se comunican por esa red y un fallo causaría reacciones por parte de éstos; y la gestión de los datos es vital para sistemas de virtualización [Ros10].

A continuación se expone el diseño planteado para este sistema tomando el esquema mostrado en la ilustración 4.4 como referencia.

A nivel general se fija que la red "v210_HeartBeat" se marque con el campo PCP a 7, la VLAN 209 con 6 y la 208 y 211 con el valor 5. En los sistemas se marca el tráfico mediante `iptables` para ser posteriormente encolado mediante las políticas fijadas con `tc`.

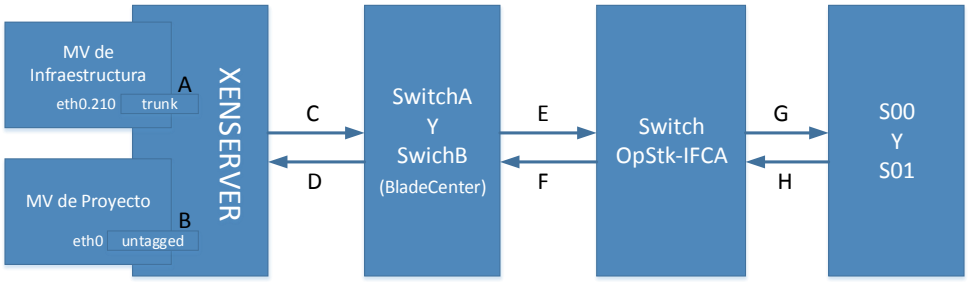


Ilustración 4.4: Esquema de elementos que participan en QoS.

En el caso de las máquinas virtuales de infraestructura (“A”), la interfaz de red asociada a las redes que tienen que ser marcadas, como por ejemplo la “v210_HeartBeat” es configurada en el sistema operativo para que etiquete el tráfico, por lo que se puede fijar el campo PCP mediante el comando de `Linux vconfig` [vB]. Sin embargo, esto no es posible en las máquinas de proyectos puesto que la interfaz que se les presenta ya está asociada a la VLAN que le corresponde al nivel de virtualización, por lo cual, el sistema operativo invitado no puede etiquetar el tráfico. Este detalle es importante, porque no hay que plantear la necesidad de “protegerse” del usuario.

Los conmutadores (“switchA” y “switchB”) de los `BladeCenter` tienen un *bug* reconocido y corregido parcialmente que permite usar solamente dos colas [IBM]. Debido a esto, se ha decidido que el tráfico de la `VLAN` “210” y “209” coexista en la misma cola prioritaria y el resto en otra. En cuando a los pesos, a la cola prioritaria se le asigna el valor 15 puesto que es el valor que indica la máxima prioridad [NT08] y a la otra 5. Además, en estos equipos se ha establecido que el valor del campo PCP para el tráfico sin etiquetar sea de 5, puesto que es el valor elegido para el tráfico de control.

El *switch* “Sw-OpStk-IFCA” permite trabajar con 8 colas, las cuales se configuran como muestra la tabla 4.2, teniendo en cuenta que la suma de todos los pesos debe ser 254 y que la cola 8 es de prioridad estricta[2]. Además, se ha de fijar que confíe en el valor de prioridad en algunos puertos.

En los equipos “S00” y “S01” se marca el tráfico con `vconfig` y se definen las políticas de encolado tal como se describió previamente.

El tráfico transmitido por “C” está marcado para la red “v210_HeartBeat”, el de “E” y ”G” para la `VLAN` 210 y 208, y el de “H”, “F” y “D” para todas las redes según los valores fijados.

4.2.5. Cableado y STP

El cableado de red realizado se muestra en el anexo A (véase página 75). Como se puede apreciar en este esquema, se han redundado todos los enlaces que ha sido posible con el fin de ofrecer disponibilidad y balanceo de carga. Para estandarizar la instalación, en los enlaces en los que era factible, este agregado se ha configurado mediante LACP [FVDH+07].

[2] El tráfico encolado en ella es transmitido antes que el de las demás colas.

Cola	PCP	Peso	Notas
1	0	2	
2	1	3	
3	2	10	
4	3	15	
5	4	20	
6	5	76	30 % para la red de control y monitorización.
7	6	128	50 % para la red de almacenamiento.
8	7	0	Prioridad estricta para la red de *heartbeat*.

Tabla 4.2: Datos de las colas de prioridad del *switch* “Sw-OpStk-IFCA”.

Sin embargo, los enlaces de los equipos “S00” y “S01” contra el conmutador “Sw-OpStk-IFCA” no se han podido gestionar mediante este protocolo debido a que el Baseboard Management Controller (BMC) de los servidores comparte el puerto de red 0 con el sistema operativo instalado en el servidor, por lo que dicho puerto tiene que poder establecer comunicación de forma aislada y, por tanto, el agregado se configurará a nivel *software*. La documentación [RHECS13] del *software* de *cluster* usado en estos servidores especifica que los modos de *bonding* soportados entre los existentes [Clo08] son el 0, el 1 y el 2, en este caso para obtener el mayor rendimiento se ha optado por el modo 2 especificando adicionalmente que se use la dirección a nivel L3+L4 para el cálculo del *hash* en función del cual se balancea el tráfico. Se ha optado por este nivel para que se tenga en cuenta el puerto en el cálculo y así varias comunicaciones entre los mismos equipos aprovechen los enlaces disponibles.

Debido a motivos de espacio y disponibilidad de material, este módulo de conectividad no ha podido diseñarse en alta disponibilidad. En concreto, el `switch` central (“Sw-OpStk-IFCA”) representa un punto de fallo único. Para subsanar este problema sería necesario emplear otro `switch`, realizar un *stack* entre ambos y dividir los enlaces redundantes de los equipos entre los dos. La conexión que se realiza contra la red corporativa desde éste *switch* se redunda añadiendo otro enlace y mediante Spanning Tree Protocol (STP) se bloquea uno de ellos.

Al plasmar el cableado sobre el esquema se aprecia (enlaces entre “SwitchB” del “BladeB” y switch “Sw-OpStk-IFCA”) la necesidad de usar una variante de STP que permite la definición de múltiples árboles, permitiendo la asignación del tráfico de cada VLAN a un árbol, consiguiendo con ello una mejor utilización de la red. Los dos protocolos existentes que implementan esta variación son *Per VLAN Spanning Tree Plus* (PVST+) y *Multiple Spanning Tree Protocol* (MSTP) [IEE11]. Aunque en la compañía se utiliza `PVST+` de Cisco y los conmutadores del `BladeCenter` son compatibles con este protocolo, el `switch` central no lo es, por lo que se utiliza el estándar MSTP.

4.2.6. Configuracón

Como ya se ha comentado (véase página XX), para no cargar la memoria del proyecto con información tipo “*how-to*” se ha publicado esta información en un blog. La configuración relativa a este apartado de la memoria se puede encontrar en las siguientes referencias [Ben13h, Ben13j].

4.3. Almacenamiento

El módulo de almacenamiento ofrece a la capa superior y al módulo de virtualización una gestión dinámica de los recursos de almacenamiento. Con este módulo se simulan las funcionalidades de una cabina de discos, usadas habitualmente en la compañía.

Exportar el almacenamiento por bloques es similar a compartir un disco físico: el equipo que se conecta al almacenamiento exportado debe dotarlo de un sistema de ficheros para poder utilizarlo. Sin embargo, al exportar un sistema de ficheros, el equipo que lo sirve lo ha dotado previamente de formato y el equipo cliente sólo tiene que conectarse para poder utilizarlo.

Durante el diseño del módulo se ha tenido en cuenta que la funcionalidad actual (exportan parte del almacenamiento disponible mediante General Parallel File System (GPFS)) de los equipos sobre los que se trabaja en este apartado ("S00" y "S01") debe seguir funcionando debido a que los equipos están en uso por otros sistemas. Esto contradice con la decisión inicial de usar CentOS en los sistemas puesto que tienen instalado `Scientific Linux Cern`.

En la ilustración 4.5 se muestra un esquema que recoge una visión general del diseño de este módulo, el cual se expone en los apartados siguientes.

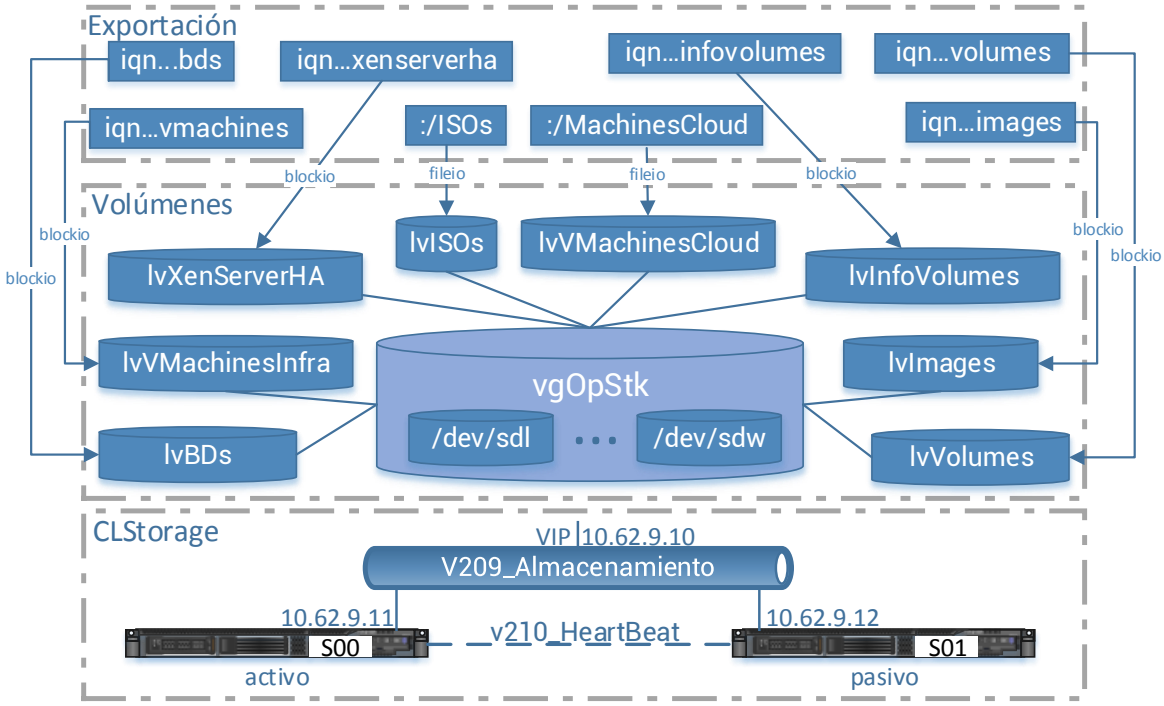


Ilustración 4.5: Esquema del módulo de almacenamiento de infraestructura.

4.3.1. Volúmenes

Como ya se ha mencionado previamente, del almacenamiento exportado por la cabina y sus expansores se pueden usar 12 LUNs. Para ofrecer una gestión del almacenamiento dinámica se hace uso de un *software* de *Logical Volume Management* (LVM). Este tipo de *software* proporciona un punto de vista de los recursos de almacenamiento a mayor nivel que el tradicional basado en discos y particiones. En concreto, en este proyecto se hace uso de `LVM2` [LVM]. A continuación se presentan sus principales componentes:

- Un *Physical Volume* (PV) es un disco duro típico, aunque puede no serlo (p. ej: un `RAID`), siempre y cuando se le presente al sistema operativo como tal (como dispositivo de bloques).
- Un *Logical Volume* (LV) es equivalente a una partición de un disco en un sistema tradicional. Los LVs son mostrados al sistema como un dispositivo de bloques estándar.
- El *Volume Group* (VG) es el mayor nivel de abstracción, representa una colección de LVs y PVs en una unidad de administración.

4.3.2. Exportación (iSCSI y NFS)

La exportación de los volúmenes representados en la ilustración 4.5 al resto de componentes del sistema se ha realizado usando iSCSI como especificaba el requisito no funcional 2 salvo donde ha sido tecnológicamente inviable; el repositorio de imágenes `ISO` y el usado para almacenar las máquinas creadas por el sistema IAAS, consumidos ambos por los `XenServer`.

En el primer caso, `XenServer` impone [Cit13a] que el repositorio de imágenes sea exportado mediante *Network File System* (NFS) o *Common Internet File System* (CIFS). En el segundo, la combinación de `OpenStack` y `XenServer` actualmente no soporta [Gub13] que el almacenamiento de las máquinas virtuales creadas sea exportado mediante iSCSI. En ambos se ha usado NFS.

Aunque no existen muchas opciones para exportar por NFS, no ocurre lo mismo para configurar un *target*, o servidor, iSCSI. En [RS13] se presenta una tabla comparativa de los software más usados en sistemas `Linux` para esta función. En este proyecto se ha optado por `SCST`, siendo una de sus principales ventajas la omisión de copias innecesarias en memoria. Para poder hacer uso de esta funcionalidad el hardware tiene que ofrecer soporte para ella, lo que puede comprobarse mediante el comando `ethtool` [MGH+05]. También es importante verificar si el procesador cuenta con las instrucciones `SSE 4.2` ya que, aunque no ocurra en este caso (Xeon 5160), gracias a ellas se podrían agilizar los cálculos de `SCST` cargando el módulo del `Kernel` `crc32c-intel`.

4.3.3. Alta disponibilidad

Con el fin de asegurar dentro de lo posible la disponibilidad del servicio de almacenamiento, éste se ha implementado mediante un *cluster* activo/pasivo puesto que las tecnologías utilizadas no permiten una configuración en activo/activo, lo que además de mejorar la disponibilidad, mejoraría

el rendimiento. Teniendo en cuenta que los equipos sobre los que se ha desplegado el servicio son heterogéneos, se ha fijado al equipo “S00” como preferente dado que tiene mejores características.

En base a los niveles de tolerancia a fallos descritos en la sección 2.4.5, este diseño ofrece el nivel 1, debido a que tras el fallo del servidor activo, los servicios tienen que arrancarse en el otro.

Un aspecto delicado en el diseño de este módulo es la gestión de la caché de escritura. En sistemas orientados específicamente a almacenamiento, los equipos cuentan con baterías para proteger la memoria caché [HP04]. Sin embargo, los equipos usados para la construcción de este módulo son servidores de propósito general que ni siquiera cuentan con una segunda fuente de alimentación. Debido a esto, aunque habilitarla mejoraría el rendimiento, no se ha realizado puesto que prevalece la integridad de los datos frente al mismo.

El *cluster* diseñado en este módulo se compone de dos servidores, lo que inicialmente inhabilita el uso del algoritmo tradicional de votos, por ello, para solventar el problema del “cerebro dividido” (véase página 13) se debe introducir un disco de *quorum* y además una heurística que permita detectar una situación de aislamiento. Para actuar sobre los equipos, el *software* de gestión del *cluster* va a disponer de un elemento de *fence* basado en el BMC de los servidores.

El *software* usado para la creación del *cluster* es `Red Hat Cluster Ad-On`. Por otro lado, el LVM usado en este diseño permite su clusterización de dos formas diferentes. La primera se basa en un conjunto de extensiones que permiten configuraciones en activo/activo. Y la otra, válida para configuraciones en activo/pasivo y soportada por el *software* de *cluster* usado, se basa en imponer la restricción de que un LV sólo puede ser activado en un equipo mediante “etiquetas” [RHECS13].

4.3.4. Rendimiento y ajustes del sistema de almacenamiento

En este apartado se evalúa el rendimiento y ajusta el sistema de almacenamiento ofrecido en este módulo de la capa de infraestructura. En él entran en juego los siguientes elementos:

- La cabina, los expansores y los discos de dichos equipos.
- Los servidores “S00” y “S01” conectados a las controladoras de la cabina que ofrecen el almacenamiento al resto del sistema.
- Los clientes de éste módulo: los `XenServer` instalados en los *blades* y algunas máquinas virtuales funcionando sobre éstos.
- La conexión entre los servidores y las controladoras de la cabina.
- La conexión entre los clientes y los servidores.

El primer grupo de elementos, que representa los recursos de almacenamiento “en bruto” se encuentra configurado previamente con 22 LUNs formadas cada una por 5 discos en `RAID` 5. La conexión entre los servidores y las controladoras se realiza con tecnología FC a 4 Gbps (máximo 500 MBps). De igual forma, los `XenServer` cuentan previamente con configuraciones específicas para

sistemas de almacenamiento remoto, por lo que no van a ser modificados. Sin embargo, la configuración de los servidores y las máquinas virtuales sí puede ser modificada y, la conexión entre los servidores y los clientes ya ha sido diseñada previamente teniendo en cuenta el rendimiento.

A la hora de ajustar la configuración del sistema de ficheros de un sistema `Linux` se han de revisar principalmente los parámetros descritos a continuación [Nas12, Mon13, Mon12b, Raj12, Mon12a]:

- scheduler: El planificador de entrada/salida usado actualmente por defecto en los sistemas `Linux` se denomina *Complete Fair Queuing* (CFQ) [Jon09] y funciona bien en discos locales porque le permite al `Kernel` optimizar las peticiones de entrada/salida reordenándolas. Sin embargo, se ha de tener presente que este proceso añade latencia al sistema. Por ello, para sistemas con altas tasas de entrada/salida es recomendable cambiar el planificador a `noop`, el cual se basa en no planificar, es decir, usa una cola `FIFO` para las peticiones [Wik13].
- nr_requests: Este valor controla el numero de peticiones que se reservan en el sub-sistema de bloques del `Kernel`. Por defecto es 128.
- Read Ahead Size: El tamaño de las páginas de caché del sistema de ficheros. Cuando el `Kernel` realiza una lectura del sistema de ficheros la información se almacena en caché para que las subsiguientes lecturas no bloqueen el disco. Por defecto es 256 (128 KB).
- max_sector_kb: Número máximo de KB que el sub-sistema de bloques permite para una petición del sistema de ficheros. Debe ser menor o igual que el tamaño máximo admisible por el hardware (max_hw_sectors_kb). Por defecto es 512 (256 KB).

Lo primero que se ha de evaluar es el rendimiento apreciable en los servidores contra los recursos de almacenamiento expuestos por la cabina. Para ello se determinan dos pruebas a realizar desde el servidor “S00”, una contra uno de los discos exportados por la cabina y la otra contra un fichero almacenado en el sistema de ficheros GPFS existente en el servidor. Ambas pruebas leen y escriben 10 GB de información de manera secuencial mediante el comando `dd` [RMK]. Los resultados obtenidos con los valores por defecto para los parámetros expuestos previamente y con los valores usados finalmente tras varias pruebas realizadas se exponen en la tabla 4.3.

	Tasa de lectura	**Tasa de escritura**	**iowait pico**
Parámetros	scheduler=cfq, rn_requests=128, read ahead=256, max_sectors_kb=512		
Prueba1	70 MBps	61 MBps	
Prueba2	164 MBps	156 MBps	70 %
Parámetros	scheduler=noop, rn_requests=1024, read ahead=16384, max_sectors_kb=8192		
Prueba1	180 MBps	130 MBps	
Prueba2	359 MBps	331 MBps	30 %

Tabla 4.3: Resultados de las pruebas propuestas para evaluar la cabina.

Posteriormente se prueba con el mismo comando contra un LV de 2 TB gestionado por el LVM y se obtiene una tasa de lectura de 262 MBps y 255 MBps para escritura. La única optimización que se puede introducir en este punto es dividir los LVs creados en varios PVs. Creando un LV del mismo tamaño divido entre 6 PVs y realizando la misma prueba se obtienen unos resultados de 322 MBps para lectura y 306 MBps para escritura.

La siguiente prueba realizada añade una capa más de *software* y usa el mismo comando contra uno de los destinos iSCSI ("S00" como destino e iniciador) obteniendo unos resultados de 144 MBps para lectura y 165 MBps para escritura.

4.3.5. Configuración

En las secciones anteriores se ha mostrado el diseño y algunas consideraciones para la construcción del módulo de almacenamiento de la capa de infraestructura, sin embargo y, al igual que otros módulos, la construcción se encuentra detallada en el blog [Ben13c, Ben13p, Ben13o].

4.4. Virtualización

El sistema de IAAS a desplegar va a ofrecer computación basada en la creación de máquinas virtuales por lo que, para ello, la infraestructura tiene que ofrecer una capa de virtualización.

En base a la matriz de características [Oped] que presentan varios hypervisores al trabajar con `OpenStack`, y a la búsqueda de otras que se consideran de interés para el proyecto, resumidas ambas en la tabla 4.4, se ha decidido que el módulo de virtualización sea gestionado con `XenServer`.

Hypervisor / Característica	Xen	VMWare vSphere	Hyper-V	XCP	XenServer
Interfaz de gestión "amigable"	✘	✔	✔	✔	✔
Consola VNC	✔	✔	✘	✔	✔
VLANNetworkManager	✔	✔	✘	✔	✔
Gratuito	✔	✘	✔	✔	✔
HA para MVs	✘	✔	✔	✘	✔
Soporte de fabricante	✘	✔	✔	✘	✔

Tabla 4.4: Disponibilidad de características en los diferentes hypervisores.

4.4.1. XenServer

`XenServer` es una solución de virtualización de código abierto que incluye el hypervisor `Xen` y el *stack* de herramientas `XenAPI` [Xen] entre otras características. Su administración y configuración se realiza mediante una aplicación cliente [CT12, opea] con una interfaz gráfica similar a la usada para el actual entorno de virtualización que posee la compañía, lo que permitirá al departamento de operaciones adaptarse fácilmente a la plataforma. Asimismo, las limitaciones de configuración de `XenServer` como plataforma de virtualización pueden encontrarse en [Cit13b].

Xen surgió como un proyecto de investigación de la universidad de Cambridge en 2001 como continuación del proyecto XenoServers liderado por Ian Pratt. Posteriormente, en 2003 se presentó oficialmente su versión 1.0 junto con un documento llamado "Xen and the Art of Virtualization"[BDF+03]. En 2007 Citrix compra XenSource (fundada en 2005) y un tiempo después comienza a liberar el código de Xen paulatinamente hasta liberarlo completamente el 24 de Junio de 2013.

Xen es un hypervisor de tipo 1 (véase página 10) y, en cuanto a los tipos de virtualización, soporta paravirtualización como opción preferida para ofrecer un mejor rendimiento y virtualización completa asistida por *hardware* en el caso de que no sea posible modificar las máquinas virtuales. Esta segunda opción se implementa [Mat09] mediante la emulación de dispositivos ofrecida por QEMU.

4.4.2. Backend de red

La elección de la tecnología de red usada en XenServer es una decisión clave para el diseño del sistema. Como se detalla en posteriores secciones, el diseño realizado impone la necesidad de gestionar las redes de proyectos en cada nodo de computación. De los *backend* de red soportados por el hypervisor, los cuales se presentan a continuación, sólo el segundo admite este diseño.

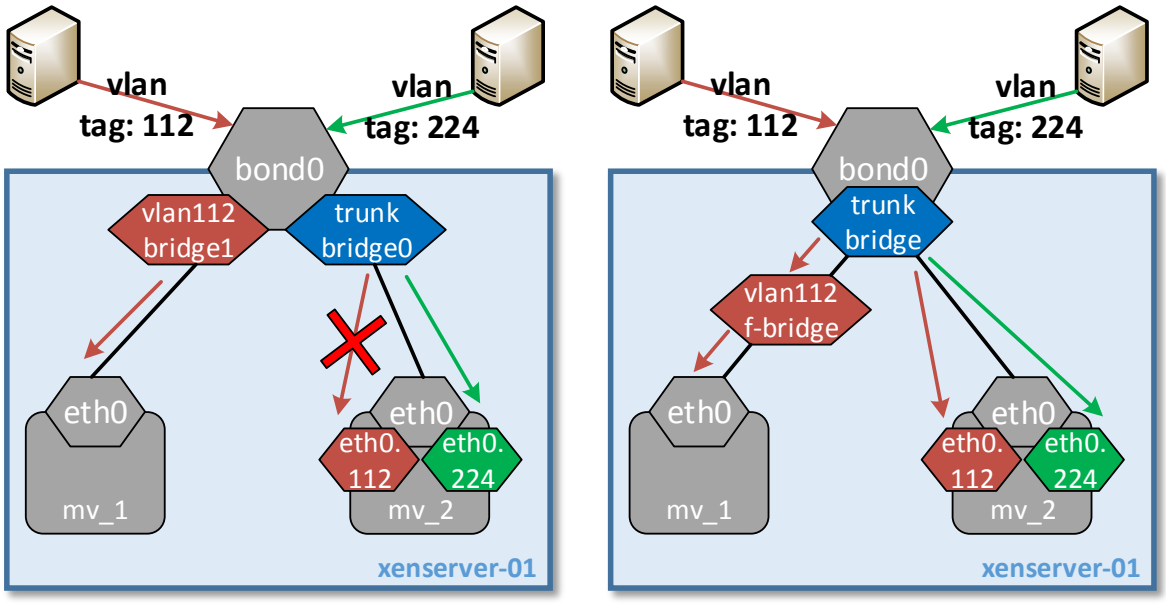
Desde la versión 5.6 FP1, XenServer permite seleccionar entre el clásico Linux Bridge [Fou09] y Open vSwitch. Esta segunda opción se convierte en la preferida a partir de la versión 6.0.

Open vSwitch cuenta con varias características interesantes [opei] pero, sin embargo, la que permite que se adapte al diseño propuesto para este sistema es su modelo de implementación de un *bridge* [Per00, IEE04] y lo que denominan “fake bridge” [opej]. En su modelo, cada puerto se puede comportar como un puerto “trunk” y con ello permitir el paso de paquetes etiquetados ó asignarse a una VLAN. Esta segunda opción se usa en la implementación actual cuando, por compatibilidad con *software* pensado para trabajar con Linux Bridge, se crea un “fake bridge”. Cuando algún puerto asociado a un *bridge* recibe tráfico etiquetado, este tráfico puede acabar tanto en los puertos “trunk” como en los puertos asociados a la VLAN de la etiqueta, no estando este comportamiento contemplado en Linux Bridge. Esta diferencia de comportamiento se ha intentado representar en los dos esquemas de la ilustración 4.6 y se ha probado la configuración mostrada.

4.4.3. Alta disponibilidad en XenServer

Como se ha mencionó previamente, en un entorno de IAAS habitualmente no se ofrece la disponibilidad de las máquinas como característica. Sin embargo, para ofrecer un buen servicio, la infraestructura que sustenta y gestiona el sistema sí tiene que estar disponible. Para garantizar esa disponibilidad se puede hacer uso, en primera instancia, de las características de XenServer, las cuales permiten lo siguiente [Cit13a]:

- Reiniciar las máquinas virtuales que estaban en ejecución en un hypervisor que ha tenido un problema en otro servidor disponible.
- Recuperar el control administrativo del *pool* si el maestro no se encuentra disponible.



(a) Linux Bridge

(b) Open vSwitch

Ilustración 4.6: Comparación entre los dos posibles backend de red.

- Permite definir políticas de reinicio (con prioridades) de máquinas que están en equipos sin problemas si éstas pertenecen a un servicio en el que alguna máquina ha tenido un problema. Para ello se debe haber definido previamente el orden de arranque de las diferentes máquinas virtuales que pertenecen al servicio.

En alta disponibilidad para una infraestructura de nodos de virtualización hay que tratar el término *overcommitting*. Esto ocurre cuando en una agrupación de hypervisores se instancian más máquinas de las que pueden ser reiniciadas si ocurren tantos fallos de equipos como ha fijado el administrador. Para evitar este problema, el centro de control de la infraestructura avisa si detecta esta situación al calcular el plan de alta disponibilidad. Es importante destacar que el número de equipos que pueden fallar tiene en cuenta cualquier equipo, hecho que hay que tener en cuenta si los servidores son heterogéneos en cuanto a recursos.

Si un hypervisor detecta una situación de aislamiento, que se produce cuando no puede contactar con el maestro del *pool*, o tiene problemas con el *software* de control, automáticamente se reinicia para liberar los ficheros de las máquinas virtuales y que el resto de equipos cumplan las políticas de alta disponibilidad. Esto es lo mismo que se ha configurado en el *cluster* de almacenamiento pero a nivel de máquinas virtuales.

Los requisitos para habilitar esta característica son [Cit13a]:

- Un almacenamiento compartido de 356 MB o superior, en el que se crearán dos volúmenes, uno de 4 MB para *heartbeat* y otro de 256 MB para metadatos.

- Un *pool* de `XenServers`.
- La asignación de IP estática en todos los nodos del *pool*.

El sistema permite fijar si una máquina debe ser protegida o no, su prioridad respecto a las demás que varía entre 0 (la mayor) y *best-effort* (la menor) y un tiempo de espera antes de reiniciarla que empieza a contar con el arranque de la última máquina de prioridad inferior.

Para que una máquina virtual esté protegida por esta característica tiene que cumplir los siguientes requisitos [Cit13a]:

- Tener los ficheros de sus discos virtuales en un almacenamiento compartido.
- No tener una conexión con un DVD local de un servidor.
- Sus interfaces de red virtuales tienen que estar asociadas a redes definidas a nivel de *pool*.

Como se ha introducido al comienzo de este apartado, esta característica se va a usar para asegurar la disponibilidad de las máquinas que conforman la infraestructura del sistema IAAS. Sin embargo, es importante destacar que las máquinas que controlan la computación no se van a proteger, debido a que están ligadas al hypervisor sobre el que se ejecutan, es decir, si un hypervisor no es funcional, la máquina que controla la computación de ese hypervisor no puede funcionar.

4.4.4. Construcción

A lo largo de las secciones anteriores se ha mostrado el diseño y las decisiones tomadas para la posterior construcción del modulo de virtualización. Dicha construcción y las partes en las que se ha dividido se pueden encontrar en el blog [Ben13k, Ben13b, Ben13q, Ben13n, Ben13g].

4.5. Monitorización

La monitorización de los entornos es un tema necesario para cualquier departamento de operaciones [CW07] y, en concreto para este proyecto, esa necesidad se pone de manifiesto en el requisito no funcional 6. Este requisito también fijaba a `Zabbix` [SIA] como la herramienta a usar. Además, se impone por herencia de la compañía que la monitorización sea de tipo activo, es decir, no se procesan *traps* *Simple Network Management Protocol* (SNMP).

Como ya se adelantó en el apartado de red, se van a realizar tareas de monitorización a través de dos redes, la de gestión y otra dedicada exclusivamente a monitorización. Esto es así porque la monitorización de la capa de infraestructura se ha enfocado desde dos puntos, uno mas *hardware* o físico, que se encarga de monitorizar los equipos; y otro mas *software* o lógico que se encarga de los sistemas operativos y los servicios.

En la cabina de almacenamiento surge el primer problema debido a la política de evitar el uso de *traps* SNMP: puesto que las controladores sólo permiten esta opción de monitorización, lo único que se va a poder monitorizar es su estado eléctrico realizándolas un `ping`.

Continuando con el análisis de los servidores “s00” y “s01”, a nivel físico se tiene la posibilidad de preguntar por *Intelligent Platform Management Interface* (IPMI) al BMC. Sin embargo, en la compañía se cuenta con la versión 2.0 de la herramienta, que presenta la limitación de no poder consultar sensores discretos mediante este protocolo [Car10], lo que permite solamente consultar a este nivel la temperatura ambiente y su estado mediante un `ping`. A nivel lógico la monitorización se basa en un agente que se instala en los dos equipos que permite obtener parámetros del sistema operativo.

El módulo de administración del `BladeCenter` permite ser consultado mediante SNMP, por lo que a nivel físico se usa este protocolo para obtener la información requerida. Por el tipo de equipos, la posibilidad de trabajar a nivel lógico es nula.

Por último, los `XenServer` permiten lo contrario, por lo que se les pregunta directamente a nivel lógico mediante el agente, el cuál reside en el `dom0`.

La configuración de este apartado no se muestra ni se cita porque no se considera relevante para el proyecto. Se cumple el requisito puesto que es importante pero no le aporta valor al proyecto.

4.6. Relación entre módulos

En los apartados anteriores se ha descrito la funcionalidad, características y diseño de cada módulo en los que se ha dividido conceptualmente la capa de infraestructura, la relación entre los módulos y con la capa superior se representa en la ilustración 4.7.

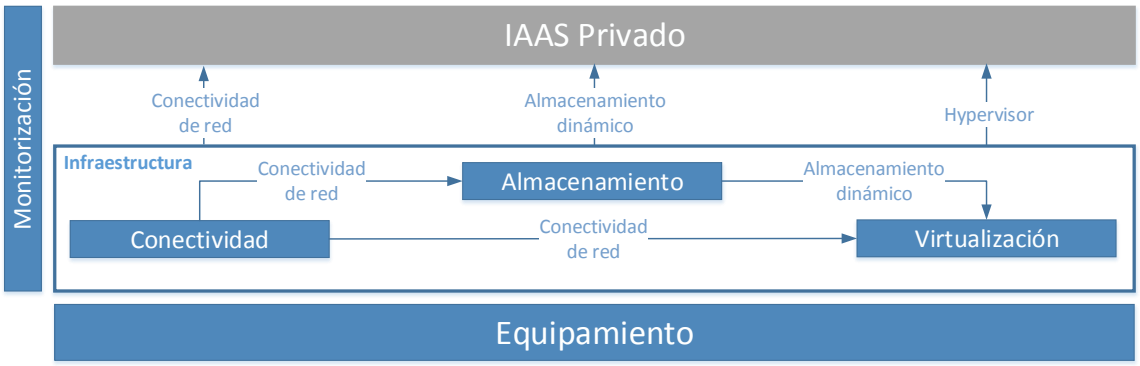


Ilustración 4.7: Representación de las relaciones entre módulos resaltando infraestructura.

5 IAAS privado

La parte más interesante del proyecto que se presenta con este documento se plasma en este capítulo. En el cual, se diseña una plataforma de *Cloud Computing* privada de tipo *Infrastructure as a service* para satisfacer las necesidades de la empresa en la que se ha llevado a cabo.

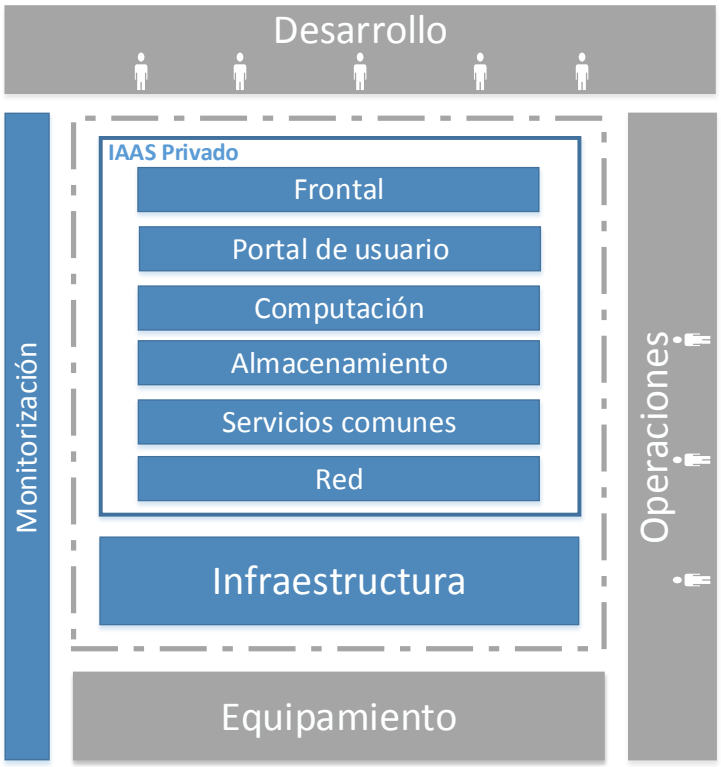


Ilustración 5.1: Diagrama de módulos resaltando y expandiendo la capa de infraestructura.

El sistema IAAS privado a implantar en este proyecto va a permitir a los equipos de desarrollo auto aprovisionarse de la infraestructura necesaria para sus proyectos. Se basa en la plataforma `OpenStack` y otras tecnologías *open source*. Para abordar su diseño y construcción se ha dividido en módulos, tal y como puede verse en la ilustración 5.1.

A grandes rasgos, el módulo de “red” se encarga de la gestión de las comunicaciones de las máquinas virtuales desplegadas. Por su parte, en el módulo de “servicios comunes” recaen varios elementos, tanto de `OpenStack` como externos a la plataforma que ofrecen algún servicio al resto

de módulos. Siendo más específico, el módulo de “almacenamiento” se encarga de la gestión del almacenamiento de bloques. Para soportar la principal funcionalidad del sistema está el módulo de “computación” que, apoyado por la interfaz ofrecida en el módulo de “portal de usuario” ofrece la gestión de máquinas virtuales y, por último, el módulo “frontal” se encarga de recibir y balancear todas las peticiones hacia los servicios desplegados en los anteriores módulos. Al igual que la capa de infraestructura, los sistemas y servicios de ésta también deben monitorizarse.

A la hora de plantear el diseño de los módulos se ha tenido muy en cuenta su disponibilidad, exponiendose tanto ésta como otras características en las siguientes secciones. Para facilitar su comprensión se representan en la ilustración 5.2 las MVs que soportan el sistema y sus servicios.

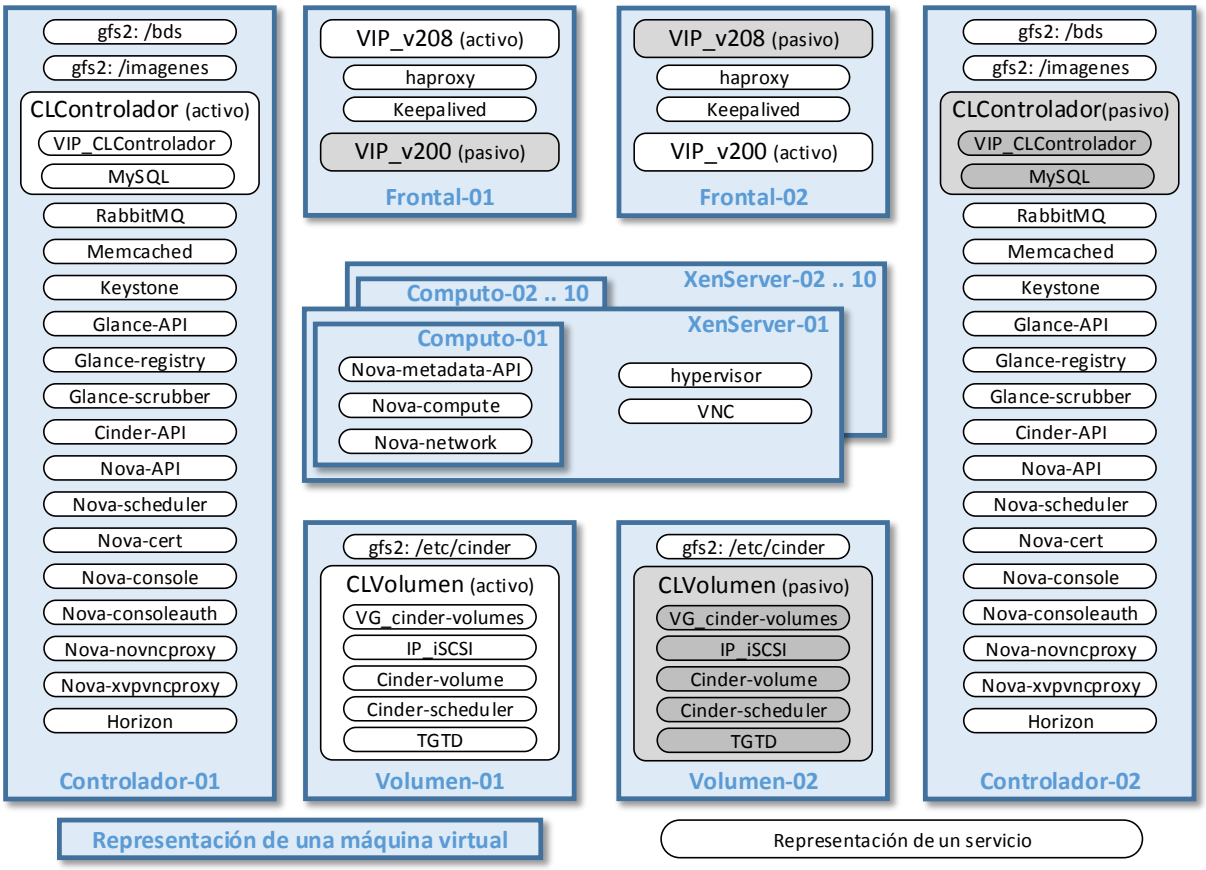


Ilustración 5.2: Diagrama en el que se presentan las MV usadas y sus servicios.

Los componentes resaltados en un color más oscuro en la ilustración anterior son componentes que están detenidos esperando a ser activados, desencadenándose su activación sólo si su servicio gemelo alojado en otro nodo deja de funcionar. Salvo el *cluster* de los equipos “frontales” y, al igual que el *cluster* de almacenamiento, los mostrados en esta ilustración son gestionados mediante el *software* de *clustering* [RHECS13] ofrecido en el sistema operativo, que, cumpliendo con el requisito no funcional número 7 es `CentOS 6.4`. Se ha aprovechado la flexibilidad que ofrece trabajar con máquinas virtuales para gestionar las tareas de *fencing* mediante la API del hypervisor. Y, puesto que las MVs que conforman los *clusters* tienen los mismos recursos, no es importante especificar en qué MV se ejecutan los servicios, aunque se les asigna una prioridad inicial por claridad.

5.1. Red

Este módulo se encarga de la comunicación de las máquinas virtuales trabajando conjuntamente con la capa de infraestructura. La versión de `OpenStack` desplegada permite gestionar la red con `nova-network` o con `quantum` [Ope12] (actualmente denominado `neutron`), por estabilidad se ha optado por la opción clásica (`nova-network`) puesto que además satisface todos los requisitos.

Los manejadores de red presentados en la sección 4.2.1 por defecto hacen uso del *driver* de nivel 3 para `Linux` [l3p, lina], el cual realiza todas las operaciones necesarias para controlar la red mediante comandos estándar como `iptables` o `route`.

5.1.1. Redes de proyecto

Como impone el requisito no funcional 1, las redes de diferentes proyectos deben ser aisladas, para lo que se ha realizado un diseño basado en VLANs. En [Siw12] se puede encontrar un análisis detallado del funcionamiento del manejador de red `VLANNetworkManager` en varios escenarios.

La creación de las redes diseñadas en el capítulo previo se resuelve fácilmente en el lado de `OpenStack` debido a las facilidades que ofrece el comando `nova-manage` [Opee]. Sin embargo, se vuelve más tediosa en el lado de los *switchs*, es por ello que para obtener los comandos a ejecutar sobre éstos se ha realizado un diseño `Java`, el cual se presenta en la ilustración 5.3.

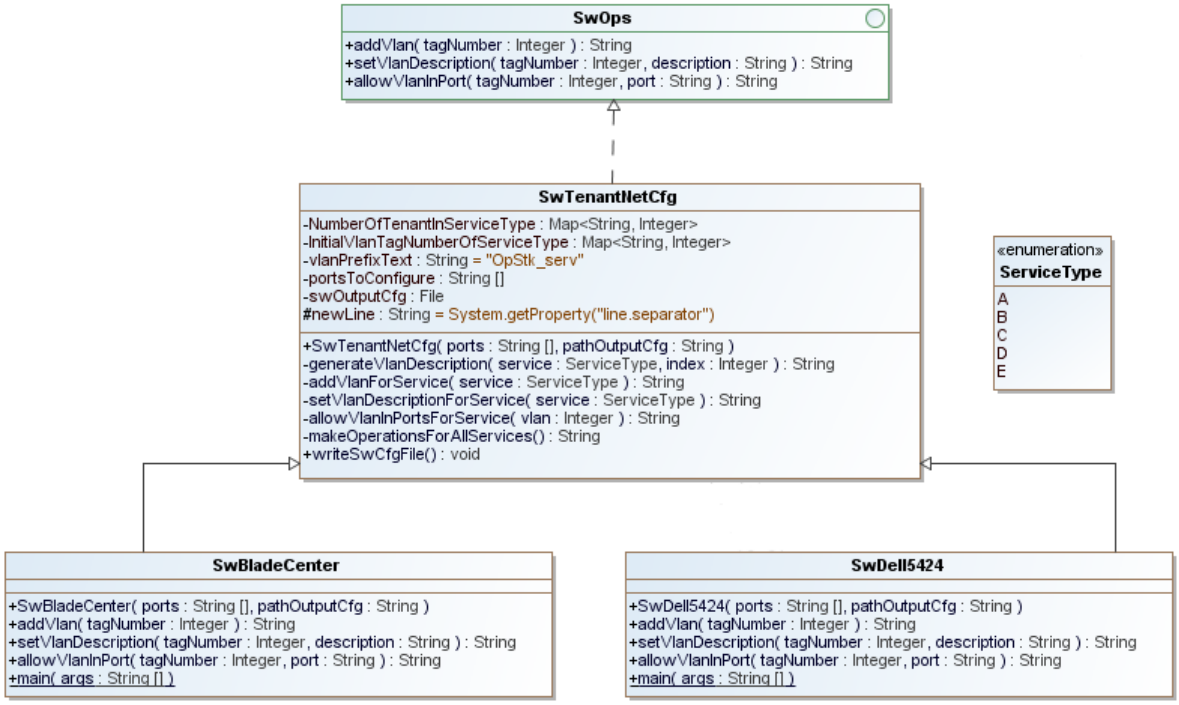


Ilustración 5.3: Diagrama de clases UML.

5.1.2. Diseño multi-host

Como se avanzó en el capítulo previo, el diseño del módulo de "Red" se ha realizado teniendo en cuenta la escalabilidad del sistema. Esta característica se puede obtener desplegando una instancia del servicio `nova-network` en cada nodo de computación ("Computo-01" a "Computo-10") [Ope12].

Este servicio gestiona la configuración necesaria para que cada máquina de cómputo actúe como `gateway` de red para las máquinas virtuales que controla. Las implicaciones de esta decisión de diseño sobre otros apartados del proyecto fue comentada cuando se expuso el módulo de virtualización de la capa de infraestructura (véase la página 36).

5.1.3. OpenStack, VLANNetworkManager y XenServer

Las máquinas virtuales son ejecutadas por `XenServer` bajo la gestión que realizan los servicios de `OpenStack` alojados en una máquina virtual en cada equipo de computación. El proceso que se ejecuta al instanciar una máquina virtual perteneciente a un proyecto cuando se usa el gestor de red basado en VLAN y `XenServer` se divide en los siguientes pasos:

1. En base al parámetro de configuración `vlan_interface` se identifica y se crea si no existe la Physical Interface (PIF) [Wit08] que ha de ser usada para la red del proyecto en `XenServer`. La etiqueta a usar se obtiene de la columna `vlan` de la tabla de red de la base de datos.

2. Conectar la Virtual Interface (VIF) [Wit08] para la máquina virtual a la red creada previamente. En caso de querer realizar un seguimiento del proceso, hay que tener en cuenta que el *bridge* asociado no se muestra en el sistema hasta que se conecta una VIF.

3. Configurar en la máquina virtual que ejecuta los servicios de `OpenStack` la VLAN necesaria en la interfaz especificada por el parámetro de configuración `vlan_interface`.

La maquina virtual donde se ejecutan los servicios de `OpenStack` no se conecta a la red creada en el primer paso, sino que está conectada mediante un puerto "trunk" al bridge padre de los "fake bridge" creados para cada una de las redes de proyecto a las que se conectan las VIF.

Como se puede ver en los pasos mostrados previamente, el parámetro de configuración que especifica la interfaz de red en "trunk" de la máquina virtual que tiene los servicios `OpenStack` y la interfaz a la que se conectan las PIF creadas para las redes de proyectos es el mismo. Esto supone una limitación que fue reportada como *bug* [GB13] y es un problema en este diseño, aunque para poder continuar con el diseño se ha aplicado un parche temporal (cambiar "FLAGS.vlan_interface" por "bond0") en el código [WEM+].

5.2. Servicios comunes

En esta sección se incluyen tanto servicios externos como propios de `OpenStack`, compartiendo todos ellos la característica común de que sirven de apoyo a otros componentes del sistema.

5.2.1. Base de datos

`OpenStack` gestiona la persistencia de la información del estado mediante una base de datos compatible con `SQLAlchemy` [sql]. En este diseño se ha optado por `MySQL` por ser la "elección más popular" [Opef] y mejor documentada para la implantación de este tipo de entornos.

Este componente se ha desplegado sobre los equipos "Controlador-01" y "Controlador-02" formando un *cluster* activo/pasivo. Se ha optado por esta configuración puesto que satisface las necesidades del proyecto. Los componentes de *OpenStack* acceden a la base de datos para obtener información de estado y actualizarla pero no "sufren" si durante unos segundos pierden el servicio en caso de caída del servidor activo en ese momento. Sin embargo, proyectos con mayores necesidades de escalabilidad podrían decantarse por sistemas en activo/activo [Cod, Per].

5.2.2. Gestor de mensajes

Para poder desacoplar los componentes unos de otros, `OpenStack` intercambia información entre los diferentes procesos mediante mensajes con el clásico modelo de publicador/suscriptor [Fat12]. Por la misma razón que la base de datos, para esta tarea se ha optado por `RabbitMQ` [Pivc].

Este *software* se ha desplegado en los equipos "Controlador-0{1 y 2}" formando un *cluster* activo/activo [Piva] con colas de mensajes replicadas y balanceado por el módulo "Frontal". La configuración de la replicación se gestiona mediante una política [Pivb] aplicada al servidor, para poder disponer de esta característica se instala en los equipos la versión 3 del servidor de mensajes. Este diseño ha sido condicionado por el hecho de que la versión de `OpenStack` con la que se ha desplegado el sistema no soporta la clusterización de este servicio de forma nativa.

5.2.3. Caché distribuida

`OpenStack` es capaz de trabajar con un sistema de caché de objetos distribuido para poder agilizar algunas consultas de datos y, sobre todo, para poner cierta información en común, por ejemplo, los *tokens* de acceso a la consola *Virtual Network Computing* (VNC) para varias instancias del mismo servicio. Este servicio se ha implementado mediante el *software* `Memcached` [mem] puesto que es el soportado por `OpenStack` para esta tarea.

El servicio ha sido desplegado sobre "Controlador-0{1 y 2}" y su acceso se balancea mediante el "Frontal". Para tolerar el fallo de uno de los servidores, se ha aplicado un parche al código original [Kla] que permite la replicación de los objetos almacenados en la caché de ambos servidores.

Este servicio también puede usarse para almacenar la información de las sesiones contra el portal web de administración (expuesto en secciones posteriores). Sin embargo, por escalabilidad [dja] esta información se almacenará mediante `cookies` en el equipo de los usuarios.

5.2.4. Servicio de identidad

Los conceptos sobre los que se basa el servicio de identidad de `OpenStack` están derivados de sistemas de similar naturaleza. Los usuarios poseen unos credenciales que usan para autenticarse; los sucesivos trámites con la plataforma se realizan mediante *tokens*. Por otra parte, los usuarios pertenecen a uno o varios grupos conocidos como proyectos o *tenants*.

El componente de `OpenStack` que implementa este servicio se denomina `Keystone` y ofrece un directorio central de usuarios y proyectos, el catálogo de servicios ofertados por el sistema, la gestión de políticas de acceso y la generación y validación de *tokens*. Este componente actúa como sistema de autenticación y autorización para todos los servicios de la plataforma de IAAS.

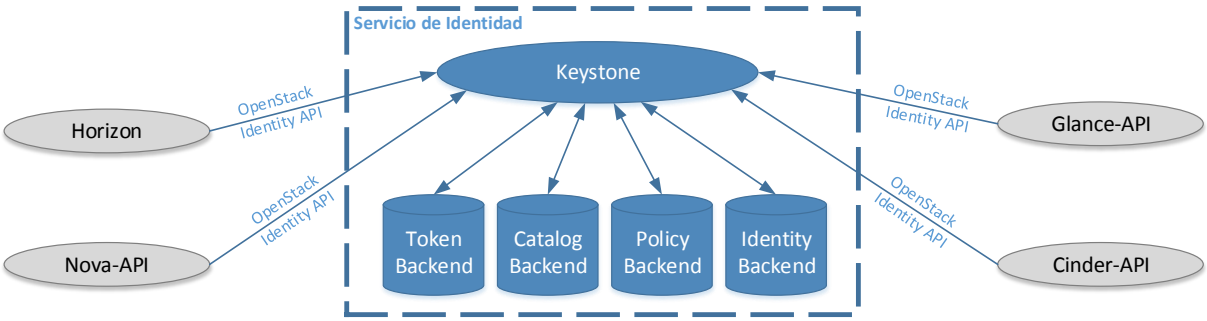


Ilustración 5.4: Diagrama de componentes del servicio de identidad y relaciones con otros servicios.

Este servicio se ha desplegado sobre las MVs "Controlador-01" y "Controlador-02", y las dos APIs que ofrece están balanceadas por el módulo "Frontal".

Debido a la versión de `OpenStack` con la que se trabaja, el requisito no funcional 5 es tecnológicamente inviable, porque cuando se obtienen todos los elementos, por ejemplo los usuarios, lo hace con un nivel de anidamiento (`scope`) [pyt] igual a uno y el esquema de la compañía es más complejo. Esta forma de trabajar con los árboles se puede ver en el código fuente [MAL]. En la siguiente versión estable de `OpenStack` ("`Grizzly`") configurar el `scope` sí es posible [Opeh].

5.2.5. Servicio de imágenes

`OpenStack` utiliza imágenes de sistemas operativos previamente instalados para crear las máquinas virtuales que se ejecutan en los nodos de computación. Estas imágenes pueden ser sistemas operativos recién instalados, sistemas con aplicativos completamente configurados y funcionales o incluso pueden ser instantáneas de otra máquina virtual desplegada en el entorno.

Este servicio es ofrecido por el componente `Glance` de `OpenStack` que se ha desplegado sobre "Controlador-0{1 y 2}" balanceando su API mediante los equipos "Frontal-0{1 y 2}". En la ilustración 5.5 se representan sus componentes, los cuales se exponen a continuación:

- `Glance-API`: acepta las peticiones de los usuarios o de otros servicios que quieran realizar operaciones con imágenes (almacenar, modificar, descargar, borrar, ...).

- `Glance-registry`: procesa la meta-información (tamaño, tipo, ...) de cada imagen.
- `Glance-scrubber`: elimina las imágenes que han sido borradas por los usuarios.
- Base de datos: almacena la información sobre las imágenes.
- Repositorio de imágenes: admite diversos proveedores de imágenes. En este proyecto, por simplicidad, se ha usado un sistema de ficheros `gfs2` montado en los equipos.

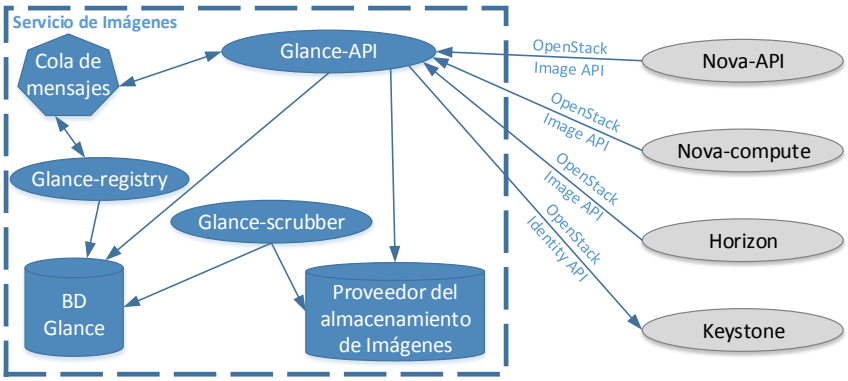


Ilustración 5.5: Diagrama de componentes del servicio de imágenes y relaciones con otros.

Las herramientas de `Glance` permiten trabajar con varios formatos de imagen [for], tanto abiertos como propietarios. Sin embargo, la integración con `XenServer` restringe un poco las opciones y sólo permite trabajar con imágenes de tipo AMI, ISO, RAW y VHD. Otra característica interesante es que trabajando conjuntamente con las herramientas del módulo de computación permite descargar la imagen a los nodos de cómputo consiguiendo así que futuros despliegues sean más ágiles.

5.3. Almacenamiento

Los discos creados con las MVs son destruidos junto con estas, por lo que el usuario pierde los datos contenidos en ellos si se elimina la MV. Por este motivo, se despliega el servicio tratado en este módulo, con el que los usuarios pueden disponer de almacenamiento bajo demanda. Este servicio se implementa mediante el componente `Cinder` de `OpenStack` que proporciona a los usuarios almacenamiento a nivel de bloque. Los usuarios interactúan con este almacenamiento conectando los volúmenes creados a MVs y, a partir de ese momento, cada volumen se comportan como un disco duro más del sistema.

Estos volúmenes ofrecen almacenamiento de tipo persistente: pueden ser desconectados de una MV y conectados a otra permaneciendo intacta la información. Hay que tener presente que un volumen sólo puede conectarse a una MV al mismo tiempo.

Por defecto, `Cinder` gestiona los volúmenes mediante el software `LVM2` [LVM] y por cada volumen solicitado por un usuario se crea un LV en el VG cuyo nombre se ha especificado en la configuración del servicio. Asimismo, cuando un usuario solicita conectar un volumen a una MV,

el primer paso es crear la configuración para exportarlo por iSCSI. Tras esto, el nodo XenServer sobre el que está virtualizada la MV abre una sesión iSCSI contra el volumen y, una vez que el hypervisor dispone del disco localmente, el último paso es presentárselo a la MV como dispositivo de almacenamiento.

El despliegue de los componentes que forman parte de Cinder se ha realizado en varios equipos: mientras que la API se ha implantado en "Controlador-0{1 y 2}" balanceada por el módulo "Frontal", el gestor de volúmenes y el planificador se han desplegado en dos equipos formando un *cluster* activo/pasivo. Esta disposición de los componentes ha sido elegida para redundar la API y, para aislar la gestión de volúmenes del resto de elementos puesto que es un servicio replicable N veces, copiando las mismas veces las MVs "Volumen-0{1 y 2}" y creando tantos volúmenes como copias del servicio en el módulo de almacenamiento de la capa de infraestructura.

5.4. Computación

Este módulo de computación es la pieza principal del sistema a desplegar en este proyecto. Apoyado por el módulo de virtualización de la capa de infraestructura, permite, el aprovisionamiento y control de máquinas virtuales bajo demanda. Su implementación se ha realizado con los componentes del proyecto Nova de OpenStack, su principal aportación es un orquestador de MVs.

5.4.1. Arquitectura

Debido a la importancia de este componente dentro de OpenStack su arquitectura es la más compleja y la que mas relaciones mantiene con el resto de módulos, como se puede ver en la ilustración 5.6. A continuación se describe cada componente de los mostrados en la citada ilustración:

- Nova-API: acepta las peticiones de los usuarios o de otros servicios que quieran realizar alguna operación sobre una máquina virtual o sobre el hypervisor. Implementa la especificación de varias APIs, la nativa de OpenStack (OpenStack Compute API), la del servicio EC2 de Amazon (Amazon's EC2 API) y la de administración.
- Nova-compute: proceso que se encarga de realizar las operaciones sobre las máquinas virtuales recibidas por Nova-API, haciendo uso de la API proporcionada por los hypervisores.
- Nova-scheduler: procesa las peticiones de creación de máquinas virtuales recibidas mediante la cola de mensajes y determina dónde debe ejecutarse cada una.
- Nova-cert: gestiona los certificados x.509 que soportan los procesos de creación de imágenes con las herramientas de gestión de Amazon EC2 y el sistema de VPN cloudpipe.
- Nova-console, Nova-novncproxy y Nova-consoleauth: son servicios que trabajan conjuntamente para permitir a los usuarios acceder a la consola VNC de las MVs.
- Base de datos: base de datos donde se almacena la mayor parte de la información relativa al estado de la infraestructura y a su configuración. Guarda información de los tipos de instancias disponibles, las máquinas virtuales en uso, las redes disponibles, los hypervisores, etc.

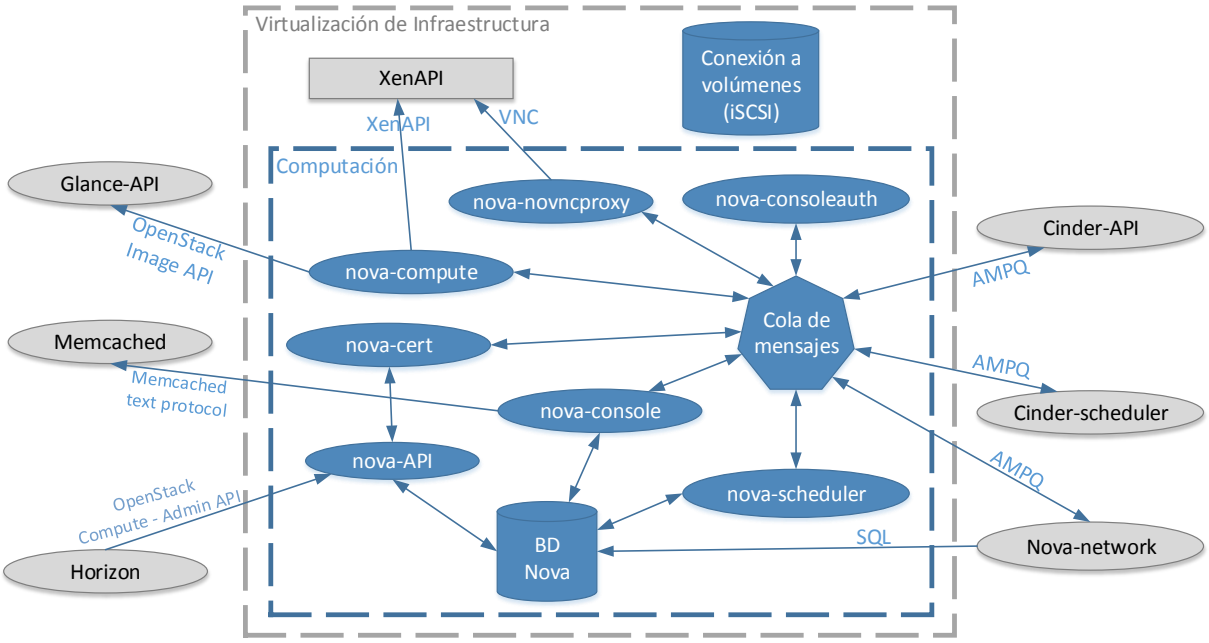


Ilustración 5.6: Diagrama de componentes del módulo de computación y relaciones con otros.

5.4.2. Host-Aggregates, XenServer y migración

`OpenStack` y su módulo de computación soportan varios hypervisores, algunos de los cuales fueron objeto de discusión de la sección 4.4 de este documento así como la decisión de usar `XenServer`.

En la versión de `OpenStack` desplegada se permite segregar un entorno mediante:

- `Availability zones`: permiten agrupar lógicamente los nodos que se encuentran aislados físicamente por argumentos de disponibilidad, como por ejemplo equipos alimentados por diferentes circuitos eléctricos. Los usuarios pueden tener presentes estas agrupaciones a la hora de crear sus infraestructuras y levantar los servicios a lo largo de diferentes zonas para intentar asegurar la disponibilidad de los mismos.

- `Host aggregates`: permiten agrupar lógicamente los nodos de computación para tareas de balanceo de carga y planificación. Por ejemplo, si hay dos clases de nodos de cómputo, se crean dos agregados con los nodos de cada tipo y se especifica que ciertas imágenes sólo pueden ejecutarse en nodos de uno de los agregados. Al contrario que la anterior, esta división no es perceptible por el usuario.

`OpenStack` implementa la gestión de un *pool* de `XenServers` como un "host aggregate". Cuando el *software* es "consciente" mediante la configuración aplicada que los nodos de cómputo pertenecen a un *pool* permite la migración de las MVs entre los mismos. Esta característica se considera necesaria para poder realizar tareas de mantenimiento de los nodos de cómputo "cómodamente".

La intregración con `XenServer` de las herramientas de gestión permiten añadir nodos a un *pool* creado previamente siempre y cuando estos cumplan las siguientes características [Cit13a]:

- No pertenecer a otro *pool*.
- No tener ningún almacenamiento compartido.
- No tener operaciones pendientes (apagar una máquina, borrarla, etc...)

5.4.3. Almacenamiento de las máquinas virtuales

Durante el proceso de creación de una MV se crea el almacenamiento no persistente sobre el que se ejecuta dicha máquina. Para proveer este almacenamiento hay principalmente tres posibilidades, cada una con sus ventajas y sus inconvenientes. Dichas opciones se enumeran a continuación:

1. Nodos de computación sin almacenamiento y sistema de ficheros compartido
2. Nodos de computación con almacenamiento y sistema de ficheros compartido
3. Nodos de computación con almacenamiento y sin sistema de ficheros compartido

En la opción 1 de las presentadas anteriormente el almacenamiento se separa de los nodos de computación, lo cual permite tratarlos como nodos “sin estado”. Esto facilita el mantenimiento puesto, que si no hay maquinas virtuales en ejecución se pueden realizar operaciones sobre los nodos de computo sin afectar al resto del entorno. Además, si un nodo de cómputo falla, las MVs que estaban en funcionamiento en dicho nodo son fácilmente recuperables [nov]. Por el contrario, un uso excesivo del almacenamiento por parte de una instancia puede afectar al rendimiento de todas las máquinas del entorno.

En la opción 2 los nodos de computación cuentan con cierta capacidad de almacenamiento pero mediante un sistema de ficheros distribuido formar un disco común. Además, en caso de ser necesario, la capacidad de almacenamiento puede ser fácilmente incrementada mediante servidores externos. Sin embargo, una operación de mantenimiento puede afectar negativamente al sistema, aunque sólo sea degradando su rendimiento. Además, no es habitual que los servidores orientados a computación cuenten con elevada capacidad de disco debido a sus dimensiones.

En la opción 3 cada nodo de computación cuenta con el almacenamiento necesario para las MV que ejecuta. Como punto fuerte, una tasa de uso alta del almacenamiento no afecta a las MV que se encuentran en otro nodo. Sin embargo, si un nodo falla, las MV que se ejecutaban en él se pierden y, al igual que en la opción anterior, la capacidad de almacenamiento suele ser bastante limitada y en este caso difícil de expandir. Además, la migración de las MV es más complicada y costosa para las máquinas puesto que hay que copiar el almacenamiento.

En este proyecto se ha optado por la primera opción debido al requisito no funcional 2 y por la posibilidad de recuperar las MVs de un nodo de cómputo que ha fallado de forma fácil y rápida. Sin embargo, este requisito no puede cumplirse actualmente con `OpenStack` y `XenServer` [Gub13]. Tras evaluar el problema con el responsable de operaciones se determinó que no era un impedimento puesto que es una característica que está en desarrollo para futuras versiones[1] y este despliegue

[1] Tras tratar el tema con uno de los desarrolladores de Citrix que trabaja con `OpenStack`, se adaptó el código que está en desarrollo para ofrecer esta característica a la versión desplegada en este proyecto y, a pesar de que se consiguió que funcionase, no se implementó puesto que no es una funcionalidad estable.

se puede considerar un prototipo como se especifica en el requisito no funcional 4. Por todo ello, finalmente se ha optado por dicha opción mediante NFS para las máquinas del sistema IAAS.

5.5. Portal de usuario

Este módulo ofrece tanto a las áreas de desarrollo como al departamento de operaciones una interfaz web con la que usar y administrar el servicio. Dicha interfaz web ha sido adaptada a las necesidades de la compañía en la que se ha realizado el proyecto y se ha implementado con el componente `Horizon` de `OpenStack`.

Dicho componente ofrece una interfaz gráfica modular basada en web para la gestión de los servicios de `OpenStack`. Concretamente, ofrece tres entornos de trabajo principalmente: "User Dashboard", "System Dashboard" y "Settings Dashboard", con los que abstrae al usuario del uso de las APIs de los diferentes componentes de un entorno basado en `OpenStack`. Con esta interfaz web se pueden realizar casi la totalidad de las operaciones a ejecutar sobre un entorno de este tipo y la totalidad de las requeridas en este proyecto.

En producción, `Horizon` normalmente proporciona su interfaz sobre una instalación del servidor web `Apache` usando el "*Web Service Gateway Interface*" de `Python` y el *framework* `Django`. El código del componente ha sido dividido en las siguientes dos piezas escritas en `python`:

- Un módulo que contiene la lógica del componente, es decir, el *software* que interactúa con las diversas APIs de los servicios de `OpenStack`.
- Un módulo de presentación que puede ser modificado para adaptarlo a diferentes entornos.

El *software* de `Horizon` ha sido desplegado sobre los equipos "Controlador-01" y "Controlador-02". Las APIs del sistema no se encuentran a disponibilidad de los usuarios finales, por lo que este portal supone el único punto de contacto entre los usuarios y el sistema desplegado. Para poder dotar al mismo de visibilidad en la red pública y de un único punto de acceso altamente disponible, se hizo uso del módulo "Frontal", lo que permite además, no exponer las máquinas sobre las que se ha desplegado a los usuarios en esta red.

5.6. Frontal

Este módulo representa uno de los puntos claves en la arquitectura diseñada para asegurar la disponibilidad del sistema desplegado. Representa el punto de conexión para la mayoría de los servicios, tanto para los usuarios como para los componentes del propio sistema. En él se implementa un balanceador de carga tanto para `HTTP` como para `TCP` en alta disponibilidad. El *software* necesario para dar el servicio ha sido desplegado sobre las MVs "Frontal-01" y "Frontal-02".

Para la implementación de este módulo se ha optado por el *software* `HAProxy` como balanceador de carga. Este *software* representa una solución rápida y fiable que ofrece alta disponibilidad y

balanceo de carga para aplicaciones basadas en TCP y HTTP. Debido a la cantidad de opciones de configuración con las que cuenta es adaptable a muchos entornos y tipos de servicio. Entre estas opciones se encuentra la que fija el algoritmo [Bon] a ser usado para balancear las peticiones, en este trabajo se ha usado por simplicidad roundrobin en todos los servicios menos para RabbitMQ que se ha usado leastconn por ser un algoritmo especialmente orientado a servicios con sesiones largas.

Todos los servicios de OpenStack trabajan sin guardar estado, por lo que es suficiente para asegurar la disponibilidad y escalabilidad de los mismos el uso de un balanceador y replicar N veces cada servicio. Además, el balanceador se ha utilizado también para los servicios Memcached y RabbitMQ, en estos dos casos actúa como balanceador TCP, mientras que en los demás HTTP.

Para asegurar la disponibilidad del balanceador y conseguir que el servicio ofrecido por este módulo se encuentre disponible el mayor tiempo posible se ha optado por el *software* Keepalived frente a otras opciones como las usadas previamente para crear *clusters* debido a la simplicidad de este software. Keepalived es un *software* de enrutado cuyo principal objetivo es proveer de una forma simple y robusta balanceo de carga y alta disponibilidad para infraestructuras basadas en el sistema operativo Linux.

Para asegurar la disponibilidad del servicio Keepalived periódicamente comprueba si el programa haproxy se encuentra en ejecución en el sistema. La otra tarea que realiza es ofrecer dos IPs virtuales (“VIP_v208” y ”VIP_v200” en la ilustración 5.2), una para la red de control y otra para la pública. Para aprovechar los recursos, la IP de cada red estará activa en uno de los servidores. Ante el fallo del servicio haproxy en uno de los equipos, el fallo de KeepAlived o el fallo del servidor, la IP migrará al otro servidor.

Este módulo es el único punto de conexión entre los usuarios y la infraestructura que soporta el sistema. Esto permite aislar las máquinas que realmente ejecutan los servicios de las redes menos controladas, en este despliegue la seguridad es menos importante porque se trata de una nube privada, pero en el caso de una pública, esta característica hay que tenerla presente.

5.7. Cuotas y tamaños de máquina

OpenStack permite imponer límites a los usuarios, los cuales se representan mediante unas cuotas generales que pueden ser personalizadas por proyecto si es necesario. Asimismo, cuando las máquinas virtuales son creadas debe elegirse un tamaño para ellas, es decir, se selecciona un conjunto de recursos virtuales. Estos tamaños se denominan *flavors*” en el lenguaje de OpenStack.

Existe la posibilidad de fijar cuotas para parámetros como CPUs virtuales, memoria, número de MVs, etc. En este caso, se pactaron unas cuotas con el responsable de operaciones las cuales se aplican por defecto a todos los proyectos. Dichas cuotas se exponen a continuación:

- Número de MVs: 5
- Número de CPUs virtuales: 2 por máquina permitida

- Número de IPs privadas: 1 por máquina permitida
- Número de IPs flotantes: 1 por máquina permitida
- Memoria RAM: 4 GB por máquina permitida
- Número de volúmenes: 1 por máquina permitida
- Tamaño por volumen: 60 GB por volumen permitida

Aunque las cuotas anteriores se han establecido en base a la máquina virtual, se pueden combinar de otra forma, es decir, aunque se haya fijado el limite de 2 CPUs virtuales por máquina (lo que impone que un proyecto sólo pueda tener 10 CPUs virtuales) se puede crear una máquina virtual con 8 CPUs.

Cabe destacar que los tamaños de máquina en los despliegues de nubes públicas son importantes puesto que una característica deseable de estos entornos es la compatibidad con otras plataformas existentes, por lo que suelen ser bastante similares. Aunque este despliegue, el cual es de tipo privado, los tamaños se deben adaptabar a las necesidades de la empresa, para ello sólo ha habido que añadir uno más a los predefinidos por `OpenStack`. En la tabla 5.1 se exponen los tamaños disponibles en el sistema.

Nombre	Memoria	Disco	Disco efímero	# CPUs virtuales
m1.xs	512 MB	0	0	1
m1.s	2048 MB	10	10	1
m1.m	4096 MB	10	20	2
m1.l	8192 MB	10	40	4
m1.xl	16894 MB	10	80	8
m1.xxl	32768 MB	10	160	8

Tabla 5.1: Tabla que recoge los tamaños posibles de MVs a crear.

5.8. Resumen del despliegue

Esta arquitectura ha sido configurada basándose en el conjunto actual de características que soporta `OpenStack` en la versión `Folsom`, poniendo énfasis en la estabilidad y la disponibilidad como se ha presentado en las secciones anteriores.

En la tabla 5.2 se presenta un resumen del despliegue realizado.

5.9. Monitorización

Como ya se ha comentado, la monitorización de los entornos es importante. En esta capa toda la monitorización se consideraría de tipo lógico según los dos puntos de vista que fueron presentados en la sección 4.5.

Versión de OpenStack	Folsom
Sistema operativo	Centos 6.4
Repositorio de paquetes	EPEL
Hypervisor	XenServer
Base de datos	MySQL
Gestor de mensajes	RabbitMQ
Servicio de red	Nova-network
Manejador de red	VLANNetworkManager
Nova-network simple o multi-host	multi-host
Repositorio de imágenes	Sistema de ficheros
Back-end para Keystone	SQL
Driver de Cinder	LVM/iSCSI
Almacenamiento de máquinas virtuales	Sistema de ficheros compartido mediante NFS

Tabla 5.2: Tabla que recoge las características del despliegue.

Para monitorizar las MVs de esta capa y los servicios que en ellas se han desplegado fue necesario instalar en todas ellas el agente de la herramienta de monitorización y crear la configuración necesaria en la herramienta.

5.10. Construcción

Al igual que en secciones anteriores a continuación se citan los procesos de construcción llevados a cabo durante el despliegue del proyecto sin describir dichos procesos con el fin de no hacer demasiado tediosa la lectura de este documento. Si los apartados de diseño despiertan el interés del lector sobre la construcción de un módulo en particular, ésta puede ser consultada en la cita correspondiente.

Aunque en ninguna de las secciones en las que se detallan los módulos de la capa “IAAS Privado” se ha hecho referencia a su construcción, se ha optado por presentarlo al final debido a que la construcción no se ha realizado módulo a módulo sino que se ha documentado por equipo en el que se realizaba.

A lo largo del diseño de esta capa se ha trabajado con 4 “tipos” de equipos: los controladores [Ben13e], los volúmenes [Ben13i], los nodos de computación [Ben13d] y los frontales [Ben13f]. Adicionalmente a estas MVs se ha desplegado otra que permite gestionar el entorno [Ben13a].

Lo que si se desea presentar en esta sección es la lista de problemas que han afectado al despliegue del proyecto, si se han podido o no solucionar, y la versión de OpenStack en la que se solucionan definitivamente. Esto queda recogido en la tabla 5.3.

Problema	Bloqueante	Parcheado	Versión soluciona
python-repoze-who-friendlyform from epel overwrites RHEL version [Bau]	✘	✔	-
XenAPI depends on /sys/hypervisor/uuid - not readable across reboot [Bal]	✔	✔	Havana
LVM over ISCSI as default SR not working [Gub13]	✘	✔	Havana
xenapi: checking agent by default is confusing [Bal13]	✘	✘	Havana
Nova API memcached key encoding error [Flo13]	✘	✘	Grizzly
VLAN mode in nova-network does not work with bonded nics [GB13]	✔	✔	-
xenapi: secgroups are not in place after live-migration [Tas13]	✘	✘	-
floating ip does not move with live migration with multi_host [Rho12]	✘	✘	Grizzly
openvswitch-nova in XenServer doesn't work with bonding [Ben13l]	✔	✔	-
openvswith-nova init.d script turn off all networks except management [Ben13m]	✘	✘	-

Tabla 5.3: Lista de los principales problemas encontrados.

6 Verificación y validación

> Uno de los mayores retos a los que se enfrenta un ingeniero sea de la rama que sea es al de verificar y validar algo que ha construido. Hasta cierto punto, si confía plenamente en sí mismo, a los procesos de verificación irá tranquilo porque confiará en que lo construido hace lo que se dijo que debía hacer. Sin embargo, a las pruebas de validación irá nervioso puesto que los usuarios son impredecibles. En esta capítulo se evalúa el sistema desplegado en este proyecto.

La verificación y validación se define como un conjunto de procedimientos, actividades, técnicas y herramientas que se utilizan, tras la construcción, para asegurar que un sistema o un módulo del mismo está acorde a su diseño y cumple con los requisitos planteados por los usuarios finales. Por un lado, la verificación tiene por objetivo comprobar si se ha construido el sistema y sus módulos de acuerdo al diseño especificado. Por otro lado, la validación tiene por objetivo comprobar si el sistema cumple con los objetivos del usuario final, en esta parte entra en juego la correcta captura de los requisitos.

El éxito de lo descrito en este capítulo para detectar alguna anomalía en algún componente del sistema desplegado se basa en la correcta creación de un plan de pruebas, el cual es necesario para señalar el enfoque, los recursos, los elementos a probar y las actividades de prueba. Una vez pasado el plan presentado, el sistema habrá sido verificado y, posteriormente, el proceso de validación se llevará a cabo por personal de la compañía, el cual aunque ha sido positivo no se presenta.

6.1. Plan de pruebas

Debido a la división en dos grandes bloques del sistema a desplegar en este proyecto, y puesto que las diferencias entre ambos bloques son elevadas, el plan de pruebas ha tenido esto en cuenta y se ha enfocado de manera diferente para cada bloque. Se ha de tener presente que el plan de pruebas de la capa de IAAS cuenta con el conocimiento adquirido en el plan de pruebas realizado sobre la capa de infraestructura, por lo que las pruebas parten de cierto nivel hacia arriba.

Respecto al plan de pruebas a ejecutar sobre la capa de infraestructura, su enfoque ha sido más “físico”. Para la realización de estas pruebas se ha necesitado el acceso por consola a los servidores, la gestión web del `BladeCenter` y la manipulación física del equipamiento y, además, se han tenido en cuenta los siguientes tipos de fallos:

- Fallos eléctricos, como desconectar la fuente de alimentación de un equipo.

- Fallos humanos, como desconectar un cable.
- Fallos del *hardware*, como la rotura de una tarjeta de red.
- Fallo del sistema operativo de los equipos simulado mediante comandos de bajo nivel sobre `/proc/sys/kernel/sysrq`.
- Fallos de los procesos y aplicativos simulados matando procesos o mediante situaciones en las que se fuerza su fallo.

Por su parte, el enfoque que se le ha dado al plan de pruebas a realizar sobre la capa IAAS es algo más de "lógico" que el anterior. Los recursos para la realización de las pruebas de esta capa son proporcionados por la capa de infraestructura: se ha hecho uso de la consola de las MVs y de la gestión del estado de las mismas. Para las pruebas se han tenido en cuenta los siguientes tipos de fallos:

- Fallos de comunicaciones simulados desactivando la VIF de las MVs.
- Fallos del sistema operativo de las MVs simulado mediante comandos de bajo nivel sobre `/proc/sys/kernel/sysrq`.
- Fallos de los aplicativos que en ellas se ejecutan simulados matando procesos o mediante situaciones en las que se fuerza su fallo.

Las pruebas realizadas sobre la capa de infraestructura se muestran en el Anexo B (véase página 77) y las realizadas sobre la capa IAAS en el Anexo C (véase página 81).

Como se ha comentado previamente y aunque el documento de validación no se incluye, a continuación se mencionan brevemente las pruebas llevadas a cabo:

1. Crear una máquina virtual en base a la imagen proporcionada en el caso de uso.
2. Crear una segunda máquina virtual con una imagen de tipo Linux.
3. Crear un volumen de 1 GB.
4. Realizar la conexión del volúmen a la máquina Windows y dar formato al disco.
5. Desconectar el volumen y conectarlo a la máquina Linux y comprobar si se visualiza el formato dado previamente.
6. Desconectar y borrar el volumen.
7. Crear otra máquina virtual en un proyecto diferente a las anteriores.
8. Publicar dos de las máquinas en la red pública.
9. Probar la navegación por internet desde una de las máquinas.
10. Probar la conectividad entre las dos máquinas a traves de las direcciones públicas.
11. Probar la conectividad contra una de las máquinas creadas desde el puesto del usuario.

12. Pausar una de las máquinas virtuales.
13. Destruir las máquinas virtuales.
14. Crear una imagen descargando el disco desde un repositorio de internet.
15. Liberar las IPs publicas reservadas y comprobar que si se solicita de nuevo una IP las vuelve a reservar.
16. Cerrar el navegador sin desconectar y volver a acceder.
17. Cerrar la sesión del portal web y volver a acceder sin hacer login.

El personal de la empresa en la que se ha realizado el proyecto dio su conformidad al sistema tras realizar sobre el mismo las pruebas mostradas previamente.

7 Conclusiones y trabajo futuro

Al finalizar un proyecto es conveniente recordar sus comienzos y analizar los objetivos alcanzados. Asimismo, es importante pensar en el futuro puesto que todo se puede mejorar o ampliar para dar soporte a otras necesidades. En este capítulo se muestran estos dos aspectos del presente Proyecto Fin de Carrera junto con una valoración personal de autor del mismo.

7.1. Conclusiones

Este proyecto tenía como objetivo principal el despliegue de un sistema de computación en la nube privado de tipo IAAS con el que las áreas de desarrollo de la compañía fuesen capaces de aprovisionarse de la infraestructura necesaria para la realización de sus proyectos. El objetivo del sistema se ha cumplido y como prueba de ello se recuerda el caso de uso expuesto en la introducción, en el que se ponían como ejemplo las necesidades de un desarrollador que realiza instaladores para un producto. Para permitirle probar sus instaladores de forma ágil se creó y subió al sistema una imagen con el software que solicitó. Tras proporcionarle unos credenciales de acceso, el programador fue capaz de obtener, sin ayuda de nadie, un entorno totalmente funcional en el que probar su instalador en apenas cinco minutos.

La principal utilidad del proyecto en la empresa es el objetivo que dio origen al proyecto. De forma adicional, otros departamentos de la empresa también se ven beneficiados por el sistema desplegado puesto que van a poder crear máquinas virtuales a mayor velocidad, con lo que montar entornos de laboratorio o simular infraestructuras de clientes va a ser mucho más rápido y menos costoso.

Aunque de momento no es una vía que se haya explotado comercialmente en la empresa, es posible adentrarse en un nuevo mercado con este tipo de sistemas, para ello habría que seguir trabajando con la plataforma y adquiriendo más experiencia.

El trabajo realizado durante las distintas fases del proyecto ha sido autónomo, sin embargo, en algunas situaciones ha sido necesaria la interacción con otros miembros de la compañía, sobre todo con el departamento de comunicaciones. Esta interacción me ha obligado a tener que definir claramente las necesidades de red del sistema, lo que me ha permitido adquirir experiencia a la hora de representar este tipo de sistemas y entornos.

Debido al carácter multidisciplinar del proyecto, he podido adquirir competencias técnicas en diversas materias: `iSCSI`, `NFS`, diseño de redes, sistemas de ficheros distribuidos, *clustering*, ... Asimismo, debido a *bugs* detectados durante la realización del proyecto he adquirido competencias

no tan técnicas que me han permitido analizar dichos problemas, exponerlos y comentarlos con desarrolladores de la plataforma.

Durante todo el desarrollo del proyecto se ha prestado especial atención a la disponibilidad de los sistemas. Con ello se ha conseguido ofrecer un sistema de computación en la nube altamente disponible pero sobre todo, se ha obtenido el conocimiento sobre varios sistemas de *clustering* en `Linux` y balanceadores de carga.

Uno de los logros más destacables que no era objeto del sistema a implantar, ha sido el montaje realizado en el módulo de almacenamiento de la capa de infraestructura. El servicio de almacenamiento altamente disponible implementado se podría comparar con el ofrecido por dispositivos comerciales orientados a tales propósitos, aunque, es evidente que no cuenta con muchas de las características de estos, puesto que ha sido desplegado sobre servidores de propósito general.

7.2. Trabajos futuros

Aunque se han cumplido los objetivos fijados para el proyecto hay más características por contemplar y, como se mencionó en la introducción, ya existían otros planes a más largo plazo cuando se comenzó este proyecto. A continuación se exponen una serie de trabajos futuros:

- **Integración de XenServer, OpenvSwitch y Neutron**: abandonar el esquema clásico de las redes y adentrarse en el mundo de las redes definidas por *software*.
- **Consolidación y eficiencia energética**: tener en cuenta la energía en sistemas de este tipo.
- **Planificadores**: relacionado con el punto anterior, aunque también se puede tratar por separado. A este tema OpenStack no le ha prestado mucha atención desde el principio pero una tarea importante es decidir dónde se ejecuta cada instancia y en función de qué parámetros.
- **Escalabilidad y disponibilidad de bases de datos**: en este proyecto se ha implementado mediante un *cluster* activo/pasivo pero sería deseable que fuese un activo/activo. En el documento se ha presentado alguna opción, pero sería interesante implementarlas y probarlas.
- **Integración con tecnologías de red como Infiniband**: uno de los objetivos para usar este tipo de sistemas en computación de alto rendimiento.
- **Ampliar el catalogo de servicio con DBaaS, DNSaaS, LBaaS, ...**: explorar otras opciones a prestar como servicio.
- **Ceilometer**: medición, control y facturación.
- **Automatización de los planes de recuperación ante desastres de** `OpenStack`: ofrecer características de alta disponibilidad también para las máquinas virtuales desplegadas con el sistemas de IAAS.
- **Integrar Zabbix con XenServer**: poder preguntar con la herramienta de monitorización directamente al hypervisor por el estado de una máquina virtual.
- **Grizzly, Havana, ...**: seguir trabajando con las siguientes versiones de `OpenStack`.

Glosario de términos

Baseboard Management Controller Procesador de propósito específico que permite monitorizar y controlar el estado del equipo. 30, 33, 39, 64

Fibre Channel Tecnología de red de alta velocidad usada principalmente para almacenamiento. 24, 33, 64

General Parallel File System Sistema de ficheros distribuido de alto rendimiento desarrollado por IBM. 31, 34, 64

Logical Unit Number Es una dirección para una unidad de disco duro. 24, 32, 33, 64

Physical Interface Representa una interfaz física en XenServer, aunque no lo es, puesto que varias PIF pueden estar conectadas a una sola tarjeta de red física. 44, 64

Priority Code Point Campo de la cabecera 802.1q que especifica un valor de prioridad entre 0 (menor) y 7 (mayor) ambos incluidos.. 28, 29, 64

Spanning Tree Protocol Protocolo de red que asegura una topología sin bucles. 30, 65

Type of Service Campo de la cabecera IP que se compone de 6 bits que indican diferentes servicios y dos bits para notificaciones de congestion explicitas. 28, 65

U Región de 1.75 pulgadas de altura en la que se dividen los “racks” verticalmente. 24

Virtual Interface Interfaz de red virtual conectada a una máquina virtual. 44, 58, 65

Virtual LAN Tecnoloía que permite crear redes lógicas e independientes coexistiendo en una misma red física. 25, 27–30, 36, 43, 44, 65

Acrónimos

API *Application programming interface*. 15, 17, 42, 46, 48, 51

BMC Baseboard Management Controller. 30, 33, 39

CFQ *Complete Fair Queuing*. 34

CIFS *Common Internet File System*. 32

FC Fibre Channel. 24, 33

GPFS General Parallel File System. 31, 34

IAAS *Infrastructure as a service*. 3, 8, 23, 35, 36, 38, 41, 46, 51, 57, 58, 61

IFCA Instituto de Física de Cantabria. 2, 24

IPMI *Intelligent Platform Management Interface*. 39

iSCSI *internet Small Computer System Interface*. 20, 32, 35, 48

IT *Information Technology*. 2

LACP *Link Aggregation Control Protocol*. 15, 29

LUN Logical Unit Number. 24, 32, 33

LV *Logical Volume*. 32–34, 47

LVM *Logical Volume Management*. 32–34

MSTP *Multiple Spanning Tree Protocol*. 30

MV máquina virtual. 27, 35, 42, 46–54, 58

NAT *Network Address Translation*. 27

NFS *Network File System*. 32

PCP Priority Code Point. 28, 29

PIF Physical Interface. 44

PV *Physical Volume*. 32, 34

PVST+ *Per VLAN Spanning Tree Plus*. 30

QoS *Quality of Service*. 28

REST *Representational State Transfer*. 17

SATA *Serial Advance Technology Attachment*. 24

SNMP *Simple Network Management Protocol*. 38, 39

STP Spanning Tree Protocol. 30

ToS Type of Service. 28

VG *Volume Group*. 32, 47

VIF Virtual Interface. 44, 58

VLAN Virtual LAN. 25, 27–30, 36, 43, 44

VLSM *Variable Lenght Subnet Mask*. 27

VNC *Virtual Network Computing*. 45, 48

Referencias

[apa] apache.org. Portal web de Apache CloudStack. http://cloudstack.apache.org. Consultada por última vez el 09 de Septiembre de 2013.

[Bal] Bob Ball. XenAPI depends on /sys/hypervisor/uuid - not readable across reboot. https://bugs.launchpad.net/nova/+bug/1157211.

[Bal13] Bob Ball. xenapi: checking agent by default is confusing. https://bugs.launchpad.net/nova/+bug/1178223, Mayo 2013.

[Bau] Moritzq Baumann. python-repoze-who-friendlyform from epel overwrites RHEL version. https://bugzilla.redhat.com/show_bug.cgi?id=741324.

[BDF+03] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):SIGOPS Oper. Syst. Rev., Oct 2003.

[Ben13a] Maraino Benito. Configuración de gestion. http://blog.mbenito.es/2013/09/configuracion-de-gestion.html, Septiembre 2013.

[Ben13b] Mariano Benito. Adaptar XCP para OpenStack. http://blog.mbenito.es/2013/07/adaptar-xcp-para-openstack.html, Julio 2013.

[Ben13c] Mariano Benito. Cluster para la capa de almacenamiento de infraestructura. http://blog.mbenito.es/2013/06/cluster-para-la-capa-de-almacenamiento.html, Junio 2013.

[Ben13d] Mariano Benito. Configuración de computo-xx. http://blog.mbenito.es/2013/09/configuracion-de-computo-xx.html, Septiembre 2013.

[Ben13e] Mariano Benito. Configuración de controlador-0x. http://blog.mbenito.es/2013/09/configuracion-de-controlador-0x.html, Septiembre 2013.

[Ben13f] Mariano Benito. Configuración de frontal-0x. http://blog.mbenito.es/2013/09/configuracion-de-frontal-0x.html, Septiembre 2013.

[Ben13g] Mariano Benito. Configuración de HA para OpenStack sobre XenServer 6.2. http://blog.mbenito.es/2013/09/configuracion-de-ha-para-openstack.html, Septiembre 2013.

[Ben13h] Mariano Benito. Configuración de red. http://blog.mbenito.es/2013/06/configuracion-de-red.html, Junio 2013.

[Ben13i] Mariano Benito. Configuración de volumen-0x. http://blog.mbenito.es/2013/09/configuracion-de-volumen-0x.html, Septiembre 2013.

[Ben13j] Mariano Benito. Configuración específica para almacenamiento. http://blog.mbenito.es/2013/06/configuracion-especifica-para.html, Junio 2013.

[Ben13k] Mariano Benito. Instalación de XCP. http://blog.mbenito.es/2013/07/instalacion-de-xcp.html, Julio 2013.

[Ben13l] Mariano Benito. openvswitch-nova in XenServer doesn't work with bonding. https://bugs.launchpad.net/nova/+bug/1218528, Agosto 2013.

[Ben13m] Mariano Benito. openvswith-nova init.d script turn off all networks except management. https://bugs.launchpad.net/nova/+bug/1218541, Agosto 2013.

[Ben13n] Mariano Benito. Parchear XenServer 6.2. http://blog.mbenito.es/2013/07/parchear-xenserver-62.html, Julio 2013.

[Ben13o] Mariano Benito. SCST soluciona los problemas de rendimiento iSCSI. http://blog.mbenito.es/2013/07/scst-soluciona-los-problemas-de.html, Julio 2013.

[Ben13p] Mariano Benito. Servicio NFS sobre el cluster de almacenamiento. http://blog.mbenito.es/2013/06/servicio-nfs-sobre-el-cluster-de.html, Junio 2013.

[Ben13q] Mariano Benito. XenServer 6.2. http://blog.mbenito.es/2013/07/xenserver-62.html, Julio 2013.

[Bon] Cyril Bonté. HAProxy Configuration Manual - opción Balance. http://cbonte.github.io/haproxy-dconv/configuration-1.4.html#4.2-balance. Consultada por última vez el 30 de Julio de 2013.

[Car10] Mark Carbonaro. Add support for ipmi discrete sensor values. https://support.zabbix.com/browse/ZBXNEXT-300, Abril 2010. Resuelto en Abril de 2013. Consultado por última vez el 05 de Septiembre de 2013.

[Cit13a] Citrix. *Citrix XenServer 6.2.0 Administrator's Guide*. Citrix, 1.0 edition, Julio 2013.

[Cit13b] Citrix. *XenServer 6.2.0 Configuration Limits*, Jun 2013.

[Clo08] CloudiBee. Network bonding: Part II (modes of bonding). http://linux.cloudibee.com/2008/02/network-bonding-part-ii-modes-of-bonding/, Febrero 2008.

[Cod] Codership. Galera Cluster for MySQL. http://codership.com/content/using-galera-cluster. Consultada por última vez el 07 de Septiembre de 2013.

[Cre81] R. J. Creasy. The origin of the vm/370 time-sharing system. *IBM J. Res. Dev.*, 25(5):483–490, Septiembre 1981.

[CT12] Rich Crusco and Stephen Turner. XenCenter. http://community.citrix.com/xencenter, Octubre 2012. Consultado por última vez el 05 de Septiembre de 2013.

[CW07] David Cannon and David Wheeldon. *Service Operation Itil, Version 3*. The Stationery Office, Mayo 2007.

[DAV13] DAVE777. How to Select a Structured Cabling Contractor for Your Business. http://www.enterprisedojo.com/how-to-select-a-structured-cabling-contractor-for-your-business/, Agosto 2013.

[dja] django. How tu use sessions. https://docs.djangoproject.com/en/dev/topics/http/sessions. Consultada por última vez el 30 de Agosto de 2013.

[Euc] Eucalyptus Systems Inc. Portal web de Eucalyptus. http://www.eucalyptus.com. Consultada por última vez el 09 de Septiembre de 2013.

[Fat12] Ishraq Fatafta. Publish-Subscribe Model Overview. http://www.slideshare.net/ishraqabd/publish-subscribe-model-overview-13368808, Junio 2012. Consultada por última vez el 05 de Septiembre de 2013.

[Flo13] Dan Florea. Nova API memcached key encoding erro. https://bugs.launchpad.net/nova/+bug/1174487, Abril 2013.

[for] Disks and Container Formats. http://docs.openstack.org/developer/glance/formats.html. Consultada por última vez el 06 de Septiembre de 2013.

[Fou09] Linux Foundation. Bridge. http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge, Noviembre 2009. Consultada por última vez el 24 de Junio de 2013.

[FVDH+07] Howard Frazier, Schelto Vna Doorn, Robert Hays, Shimon Muller, Bruce Tolley, Paul Kolesar, Geoff Thompson, and Brad Turner. IEEE 802.3ad Link Aggregation: what it is, and what it is not. http://www.ieee802.org/3/hssg/public/apr07/frazier_01_0407.pdf, Abril 2007.

[GB13] John Garbutt and Mariano Benito. Vlan mode in nova-network does not work with bonded nics. https://bugs.launchpad.net/nova/+bug/1214865, Agosto 2013.

[Gub13] Nikita Gubenko. LVM over ISCSI as default SR not working. https://bugs.launchpad.net/nova/+bug/1162382, Marzo 2013.

[Hor07] Chris Horne. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf, Octubre 2007.

[HP04] HP. Smart Array technology: advantages of battery-backed cache. http://h10032.www1.hp.com/ctg/Manual/c00257513.pdf, Noviembre 2004. Consultado por última vez el 4 de Septiembre de 2013.

[Hub01] Bert Hubert. Man page of tc. http://lartc.org/manpages/tc.txt, Diciembre 2001.

[IBM] IBM. Quality Of Service queue configuration - IBM BladeCenter. http://www-947.ibm.com/support/entry/portal/docdisplay?lndocid=MIGR-5070349.

[IEE04] IEEE. IEEE 802.1D Standard. http://standards.ieee.org/getieee802/download/802.1D-2004.pdf, Junio 2004.

[IEE11] IEEE. IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks. *IEEE Std 802.1Q-2011 (Revision of IEEE Std 802.1Q-2005)*, pages 1–1365, Aug 2011.

[insa] Protection and I/0. http://pdos.csail.mit.edu/6.893/2009/readings/i386/s08_03.htm.

[insb] Segment-Level Protection. http://www.scs.stanford.edu/nyu/04fa/lab/i386/s06_03.htm.

[Jon09] M. Tim Jones. Inside the Linux 2.6 Completely Fair Scheduler. http://www.ibm.com/developerworks/library/l-completely-fair-scheduler/, Diciembre 2009. Consultado por última vez el 24 de Junio de 2013.

[kee] keepalived.org. Portal web de KeepAlived. http://www.keepalived.org. Consultada por última vez el 09 de Septiembre de 2013.

[Kla] Klab Inc. repcached - add data replication feature to memcached 1.2.x. http://repcached.lab.klab.org. Consultado por última vez el 06 de Septiembre de 2013.

[l3p] Código fuente de nova-network l3.py. https://github.com/openstack/nova/blob/stable/folsom/nova/network/l3.py. Consultada por última vez el 06 de Septiembre de 2013.

[lina] Código fuente de nova-network linux_net.py. https://github.com/openstack/nova/blob/stable/folsom/nova/network/linux_net.py. Consultada por última vez el 06 de Septiembre de 2013.

[linb] linux-ha.org. Portal web de Linux-HA. http://www.linux-ha.org/wiki/Main_Page. Consultada por última vez el 09 de Septiembre de 2013.

[LVM] LVM2 Resource Page. http://sourceware.org/lvm2/.

[MAL] Dolph Mathews, Ionut Artarisi, and Zhongyue Luo. Código fuente de keystone, fichero core.py para la versión folsom. https://github.com/openstack/keystone/blob/stable/folsom/keystone/common/ldap/core.py#L204.

[Mat09] Mihai Matei. HVM Xen Architecture. http://www.2virt.com/blog/?p=122, Noviembre 2009. Consultada por última vez el 12 de Julio de 2013.

[mem] memcached.org. Memcached. http://memcached.org. Consultada por última vez el 06 de Septiembre de 2013.

[MGH+05] David Miller, Jess Garzik, Tim Hockin, Jakub Jelinek, Andre Majorel, Eli Kupermann, and Scott Feldman. Man page of ethtool. http://linuxcommand.org/man_pages/ethtool8.html, Enero 2005.

[Mon12a] Martin Monperrus. Performance of read-write throughput with iSCSI. http://www.monperrus.net/martin/performance+of+read-write+throughput+with+iscsi, Octubre 2012. Consultado por última vez el 24 de Junio de 2013.

[Mon12b] Martin Monperrus. Scheduler queue size and resilience to heavy IO. http://www.monperrus.net/martin/scheduler+queue+size+and+resilience+to+heavy+IO, Diciembre 2012. Consultado por última vez el 24 de Junio de 2013.

[Mon13] Martin Monperrus. I/O Scheduling for SAN and Virtualization. http://www.monperrus.net/martin/IO+scheduling+for+san+and+virtualization, Junio 2013. Consultado por última vez el 24 de Junio de 2013.

[Nas12] Daniel Nashed. Linux I/O Performance Tweek. http://blog.nashcom.de/nashcomblog.nsf/dx/linux-io-performance-tweek.htm?opendocument&comments, Abril 2012. Consultado por última vez el 24 de Junio de 2013.

[NIS12] NIST Computer Security Division. NIST SP 800-145: The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, Abril 2012.

[nov] Nova Disaster Recovery Process. http://docs.openstack.org/trunk/openstack-compute/admin/content/nova-disaster-recovery-process.html. Consultada por última vez el 06 de Septiembre de 2013.

[NT08] BLADE Network Technologies. *Alteon OS Command Reference*, Abril 2008.

[opea] OpenXenManager. http://sourceforge.net/projects/openxenmanager/. Consultada por última vez el 08 de Septiembre de 2013.

[opeb] opennebula.org. Potal web de OpenNebula. http://opennebula.org. Consultada por úiltima vez el 09 de Septiembre de 2013.

[Opec] Openstack. API Documentation. http://api.openstack.org/.

[Oped] OpenStack. Hypervisor Support Matrix. https://wiki.openstack.org/wiki/HypervisorSupportMatrix. Consultada por última vez el 13 de Julio de 2013.

[Opee] Openstack. NovaManage. https://wiki.openstack.org/wiki/NovaManage. Consultada por última vez el 04 de Septiembre de 2013.

[Opef] Openstack. OpenStack Basic Installation Guide for Ubuntu and Debian. http://docs.openstack.org/grizzly/basic-install/apt/content/basic-install_controller.html. Consultada por última vez el 06 de Septiembre de 2013.

[Opeg] Openstack. Portal web de OpenStack. http://www.openstack.org. Consultada por última vez el 09 de Septiembre de 2013.

[Opeh] Openstack. Reference for LDAP Configuration Options in keystone.conf for Grizzly. http://docs.openstack.org/grizzly/openstack-compute/admin/content/reference-for-ldap-config-options.html. Consultada por última vez el 03 de Agosto de 2013.

[opei] openvswitch.org. Features. http://openvswitch.org/features. Consultada por última vez el 05 de Septiembre de 2013.

[opej] openvswitch.org. ovs-vsctl. http://openvswitch.org/cgi-bin/ovsman.cgi?page=utilities%2Fovs-vsctl.8. Consultada por última vez el 05 de Septiembre de 2013.

[Ope12] OpenStack. *OpenStack Compute Administration Manual (Folsom)*, Nov 2012.

[Per] Percona. Percona XtraDB Cluster. http://www.percona.com/software/percona-xtradb-cluster. Consultada por última vez el 07 de Septiembre de 2013.

[Per00] Radia Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*. Addison-Wesley professional computing series. Addison Wesley, segunda edition, 2000.

[Piva] Pivotal. RabbitMQ: CCluster Guide. http://www.rabbitmq.com/clustering.html. Consultada por última vez el 30 de Agosto de 2013.

[Pivb] Pivotal. RabbitMQ: High Availability Queues. http://www.rabbitmq.com/ha.html. Consultada por última vez el 30 de Agosto de 2013.

[Pivc] Pivotal. RabbitMQ: Messaging that just works. http://www.rabbitmq.com. Consultada por última vez el 06 de Septiembre de 2013.

[por] Portal web de Nimbus. http://www.nimbusproject.org. Consultada por última vez el 09 de Septiembre de 2013.

[pyt] python-ldap.org. LDAP library interface module. http://www.python-ldap.org/doc/html/ldap.html. Consultada por última vez el 06 de Septiembre de 2013.

[Raj12] Ashok Raj. Improving Read performance of Disks using blockdev. http://ashok-linux-tips.blogspot.com.es/2012/08/improving-read-performance-of-disks.html, Agosto 2012. Consultado por última vez el 24 de Junio de 2013.

[RHECS13] Red Hat Engineering Content Services. *Red Hat Enterprise Linux 6 Cluster Administration: Configuring and Managing the High Availavility Add-On*. Red Hat, Febrero 2013.

[Rho12] Travis Rhoden. floating ip does not move with live migration with multi_host. https://bugs.launchpad.net/nova/+bug/966529, Marzo 2012.

[RMK] Paul Rubin, David MacKenzie, and Stuart Kemp. Man page of dd. http://linux.die.net/man/1/dd.

[Ros10] Josep Ros. La importancia del almacenamiento en un entorno virtualizado. http://www.josepros.com/2010/08/la-importancia-del-almacenamiento-en-un.html, Agosto 2010.

[RS13] Sebastian Riemer and SCST. Features comparison between Linux SCSI targets. http://scst.sourceforge.net/comparison.html, Abril 2013.

[SIA] Zabbix SIA. Portal web de la herramienta de monitorización Zabbix. Última consulta el 2 de Septiembre de 2013.

[Siw12] Piotr Siwczak. Understanding VlanManager Network Flows in OpenStack Cloud: Six Scenarios. http://www.mirantis.com/blog/vlanmanager-network-flow-analysis/, Agosto 2012.

[sql] sqlalchemy.org. SQLAlchemy: The Database ToolKit for Python. http://www.sqlalchemy.org/. Consultada por última vez el 04 de Septiembre de 2013.

[Tas13] Vangelis Tasoulas. xenapi: secgroups are not in place after live-migration. https://bugs.launchpad.net/nova/+bug/1202266, Julio 2013.

[vB] Ard van Breemen. Man page of vconfig. http://linuxcommand.org/man_pages/vconfig8.html.

[Vel96] Mario G. Piattini Velthuis. *Análisis y diseño detallado de aplicaciones informáticas de gestión*. Ra-Ma, Libreria y Editorial Microninfomatica, 1996.

[WEM+] Dan Wendlandt, Johannes Erdfelt, Mark McLoughlin, Ryu Ishimoto, Trey Morris, Jason Kölker, Zhongyue Luo, vishvananda, and Yuriy Taraday. Código fuente de vif.py para la versión Folsom. https://github.com/openstack/nova/blob/stable/folsom/nova/virt/xenapi/vif.py#L79.

[Wik13] Wikipedia. Noop scheduler. http://en.wikipedia.org/wiki/Noop_scheduler, Julio 2013. Consultado por última vez el 22 de Agosto de 2013.

[Wit08] Oliver Withoff. Understanding XenServer Networking - The Linux Perspective. http://support.citrix.com/servlet/KbServlet/download/17424-102-671540/Xen_Networking_in_Linux_final.pdf, Agosto 2008. Consultado por última vez el 05 de Septiembre de 2013.

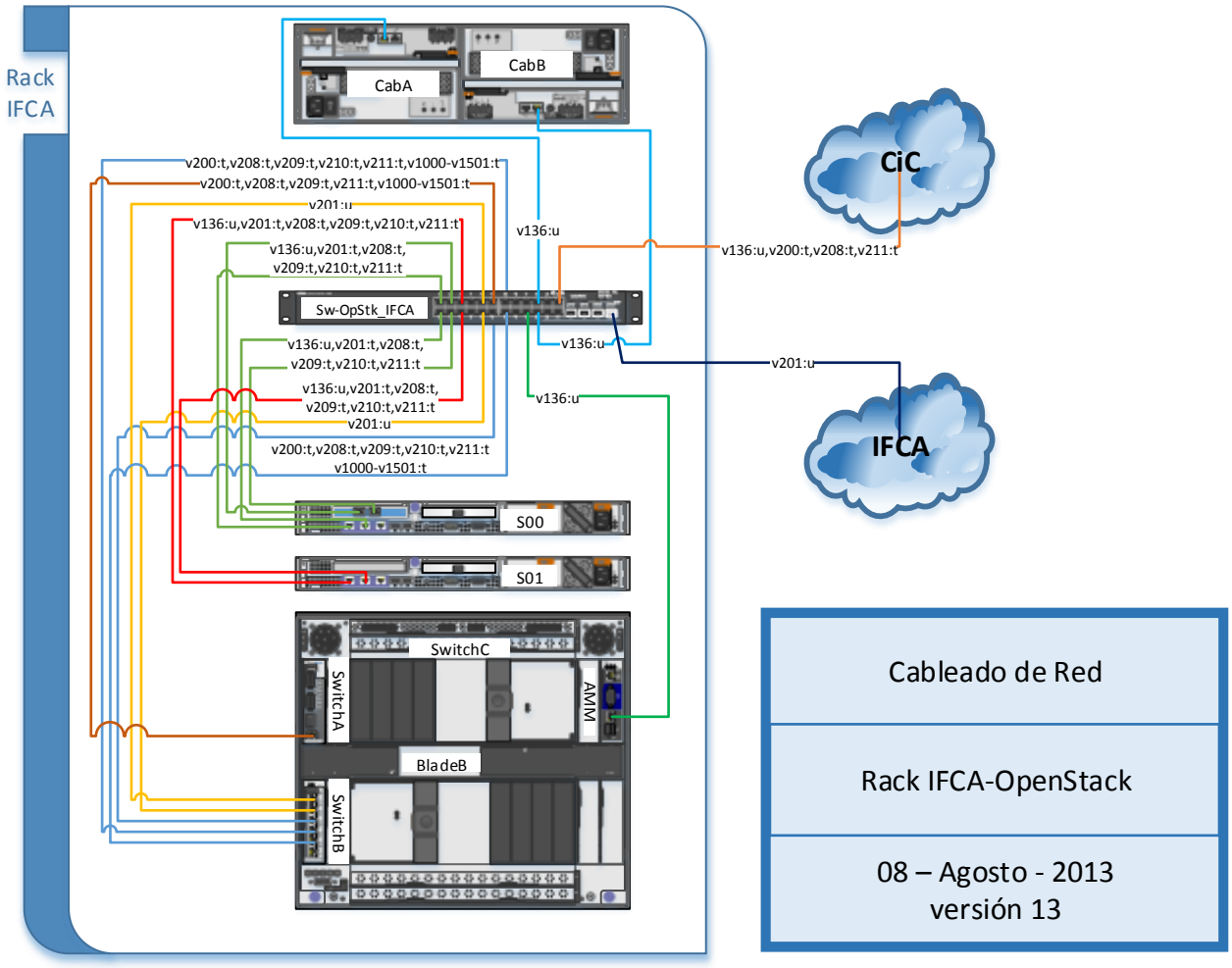
[Xen] XenProject. XAPI. http://xenproject.org/developers/teams/xapi.html. Consultada por última vez el 05 de Septiembre de 2013.

Información sobre Vlans:

v136:v136_gestionIFCA: Red de gestión del entorno OpenStack – IFCA
v200:v200_OpStk_Publica: Red pública de OpenStack
v201:v201_IFCA: Red del Instituto de Física de Cantabria
v208:v208_OpStk_Control: Red de control de OpenStack
v209:v209_Almacenamiento: Red de almacenamiento
v211:v210_HeartBeat: Red de HeartBeat de OpenStack y Almacenamiento
v212:v211_Monitorizacion: Red de monitorización y estadísticas de uso
v1000 – v1501: Redes de proyectos

Información sobre STP:

Protocolo: IEEE MSTP
Nombre: OpStk_Ifca
Instancia 0: vlan 136, vlan 4094, v1000 – v1501
Instancia 1: vlan1 y vlan 201
Instancia 2: vlan 200, vlan 208, vlan 209, vlan 210, vlan 211



Cableado de Red
Rack IFCA-OpenStack
08 – Agosto - 2013 versión 13

Anexo B: Pruebas de infraestructura

Tabla 1: Pruebas eléctricas realizadas sobre el equipamiento.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
1	Desconectar el cable de corriente de uno de los expansores.
	El expansor sigue funcionando, con la alimentación de una fuente es suficiente.
	No afecta.
2	Apagar una de las fuentes de la cabina.
	La cabina sigue funcionando con la alimentación de la otra fuente.
	No afecta.
3	Desconectar el cable de corriente de uno de los servidores (“S00” y “S01”).
	El servidor se apaga.
	Si el apagado es el servidor activo en ese momento, desencadena la migración del servicio.
4	Quitar una de las fuentes de uno de los dominios del “BladeB”..
	Un dominio aguanta funcionando con una fuente.
	No afecta.
5	Quitar las dos fuentes de uno de los dominios del “BladeB”..
	Los siete *blades* de ese dominio se apagan.
	Lo que esté en ejecución en ese *blade* se apaga, si hay alguna máquina en HA se iniciará.

Tabla 2: Pruebas realizadas sobre el módulo de conectividad.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
1	Reiniciar el “switchA” del “bladeB”.
	Los servidores detectan que se les ha apagado una tarjeta de red. Pero las comunicaciones contra servidores que tienen bond siguen activas.
	Se reduce su rendimiento.
2	Reiniciar el “switchB” del “bladeB”.
	Los servidores detectan que se les ha apagado una tarjeta de red. Pero las comunicaciones contra servidores que tienen bond siguen activas.
	Se reduce su rendimiento.
3	Reiniciar el “Sw-OpStk-IFCA”.
	Los servidores “S00” y “S01” se detectan aislados y desactivan los servicios.
	Provoca un problema grave en el sistema.

Tabla 3: Pruebas realizadas sobre el módulo de almacenamiento.

Nº	**Prueba realizada** / **Comportamiento esperado** / **Cómo afecta al servicio**
1	Desconectar los cables de red dejando uno en funcionamiento. (“S00” y “S01”)
	Los servidores detectan el fallo de las interfaces pero el *bond* sigue activo.
	Se reduce su rendimiento.
2	Desconectar el cable de fibra contra la cabina IBM DS4700 del “S00”.
	El servidor “S00” pierde acceso a los discos y se reinicia. El servidor “S01” detecta la caída, asume el control activando los servicios y lanza al dispositivo de fence configurado para “S00” la orden de reiniciarlo. Cuando el servidor “S00” se encuentra de nuevo en situación de prestar el servicio toma el control del mismo.
	El servicio se ve afectado hasta que “S01” asume el control y los iniciadores iSCSI reactivan la conexión.
3	Desconectar el cable de fibra contra la cabina IBM DS4700 del “S01”.
	El servidor “S001” pierde acceso a los discos y se reinicia. El servidor “S00” detecta la caída y lanza al dispositivo de fence configurado para “S01” la orden de reiniciarlo.
	El servicio no se ve afectado.
4	Reiniciar el servidor “S00” de forma correcta: *reboot* o desde el *software* de gestión del cluster.
	Cuando “S00” comienza a desactivar servicios, el servidor “S01” asume el control del *cluster* y levanta el servicio, además, lanza al dispositivo de fence configurado para “S00” la orden de reiniciarlo. Cuando “S00” vuelve a estar activo asume el control de nuevo.
	El servicio se ve afectado hasta que “S01” asume el control y los iniciadores iSCSI reactivan la conexión.
5	Reiniciar el servidor “S00” de forma abrupta: simulando un fallo de `Kernel`.
	El servidor “S01” detecta el fallo de “S00”, asume el control del *cluster* y levanta el servicio, además, lanza al dispositivo de fence configurado para “S00” la orden de reiniciarlo. Cuando “S00” vuelve a estar activo asume el control de nuevo.
	Igual que la prueba 4.
6	Reiniciar el servidor “S01” de forma correcta: *reboot* o desde el *software* de gestión del cluster.
	Cuando el servidor “S00” detecta la caída del “S01” lanza al dispositivo de fence configurado para “S01” la orden de reiniciarlo.
	El servicio no se ve afectado.
7	Reiniciar el servidor “S01” de forma abrupta: simulando un fallo de `Kernel`.
	El servidor “S00” detecta el fallo de “S01” y lanza al dispositivo de fence configurado para “S01” la orden de reiniciarlo.
	Igual que la prueba 6.
8	Desconectar la alimentación de “S00” de forma temporal y volver a conectarla.
	El mismo que la prueba número 5.
	Igual que la prueba número 4.
9	Desconectar la alimentación de ”S00” de forma permanente.
	El servidor “S01” asume el control del *cluster* y levanta el servicio, además, lanza al dispositivo de fence configurado para “S00” la orden de reiniciarlo indefinidamente hasta que obtiene una respuesta correcta[1]. Cuando “S00” vuelve a estar activo asume el control de nuevo.
	Igual que la prueba número 4.

Continúa en la siguiente página.

[1] Si se ha desactivado por tiempo indefinido se puede forzar a que deje de intentarlo mediante `fence_ack_manual`.

Tabla 3 – continuación de la página anterior.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
10	Desconectar la alimentación de “S01” de forma temporal y volver a conectarla.
	El mismo que la prueba número 6.
	Igual que la prueba número 6.
11	Desconectar la alimentación de ”S01” de forma permanente.
	El mismo que la prueba número 6
	Igual que la prueba número 6.
12	Aislar al servidor “S00” desactivando sus cuatro puertos de red.
	El servidor “S00” al perder conectividad con el *switch* deja de escribir en el disco de *quorum* y desactiva los servicios. Por otra parte, el servidor “S01” detecta que el activo se ha caído, por lo que asume el control y levanta el servicio, además, le intenta enviar la orden de reinicio hasta que se activa la red de nuevo y lo consigue. Cuando “S00” vuelve a estar activo asume el control de nuevo.
	Igual que la prueba numero 4
13	Aislar al servidor “S01” desactivando sus dos puertos de red.
	El servidor “S01” al perder conectividad con el *switch* deja de escribir en el disco de *quorum*. Por otra parte, el servidor “S00” detecta que “S01” se ha caído, por lo que le intenta enviar la orden de reinicio hasta que se activa la red de nuevo y lo consigue.
	Igual que la prueba numero 6.

Tabla 4: Pruebas realizadas sobre el módulo de virtualización.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
1	Apagar desde el sistema operativo invitado una máquina protegida por HA que tiene marcado su servidor preferido.
	Cuando el servidor preferido detecta que la máquina está apagada la reinicia.
	Se desactiva la funcionalidad implementada por la máquina temporalmente.
2	Apagar desde `XenCenter` una máquina protegida por HA.
	XenCenter informa de que la máquina tiene habilitada la característica de alta disponibilidad y permite continuar o cancelar, si se acepta la máquina se apaga y no es reiniciada.
	Se desactiva la funcionalidad implementada por la máquina de forma permanente.
3	Apagar un nodo en el que están virtualizadas máquinas protegidas con HA y con diferentes prioridades de arranque.
	Las máquinas se arrancan en otro servidor en el orden establecido.
	Depende de la máquina que sea apagada.

Continúa en la siguiente página.

Tabla 4 – continuación de la página anterior.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
4	Apagar un servidor en el que hay una máquina protegida y otra no.
	La máquina protegida es reiniciada en otro servidor y la no protegida permanece apagada.
	Depende de la máquina que sea apagada.
5	Apagar un servidor en el que hay una máquina protegida y otra protegida si es posible.
	La máquina protegida se reinicia en otro servidor y después se arranca la otra.
	Depende de la máquina que sea apagada.
6	Apagar el nodo maestro del pool.
	Otro servidor de los disponibles asume esta condición. Cuando el servidor se activa de nuevo, al conectarse al pool asume la posición de esclavo.
	Depende de la máquina que sea apagada.
7	Quitar un cable de red desactivando su boca en el switch correspondiente.
	El servidor detecta la perdida de una interfaz pero el agregado sigue funcionando.
	Baja el rendimiento.
8	Desactivar los dos puertos de los dos conmutadores de red (con lo que se consigue aislarlo) correspondientes a un nodo con máquinas protegidas en funcionamiento.
	El nodo detecta que se encuentra aislado y se reinicia. Otros servidores detectan que las máquinas protegidas ya no se encuentran en funcionamiento y éstas son reiniciadas en cualquier servidor disponible.
	Depende de la máquina que sea aislada.
9	Apagar “de botón” todos los servidores y activar primero el maestro.
	Cuando el maestro arranca detecta las máquinas protegidas caídas y levanta las que puede. Cuando se arrancan el resto de servidores detectan al maestro.
	Todos los servicios de computación se desactivan temporalmente.
10	Apagar “de botón” todos los servidores y activar todos a la vez.
	Los equipos esclavos se quedan esperando hasta que se activa el maestro. En ese momento se conectan a el y se reinician en cualquiera de ellos las máquinas que el maestro detecta caídas con HA habilitado.
	Todos los servicios de computación se desactivan temporalmente.
11	Apagar “de botón” todos los servidores y activar primero los esclavos.
	Los servidores se quedan esperando a detectar al master, como no le detectan porque se encuentra apagado, uno de ellos (el que arrancó primero) promociona a maestro, en ese momento los demás se conectan a él y las máquinas protegidas por HA se reinician en cualquiera de ellos. Cuando el viejo maestro arranca adopta la posición de esclavo.
	Todos los servicios de computación se desactivan temporalmente.

Tabla 5: Pruebas realizadas sobre los equipos “Volumen-0X”.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
1	Matar cualquiera de los servicios soportados por el *cluster* del servidor activo.
	El servicio se migra al otro nodo y lanza una peticion de reinicio.
	Durante unos segundos las nuevas peticiones no son aceptadas.
2	Apagar la máquina activa.
	El servicio se migra al otro nodo y lanza una peticion de reinicio.
	Durante unos segundos las nuevas peticiones no son aceptadas.
3	Apagar el nodo que no se encuentra activo.
	El nodo activo ordena que se reinicie al otro nodo mediante la XenAPI.
	No se ve afectado.
4	Desactivar el sistema de ficheros compartido en el servidor activo.
	Esto provoca el fallo del servicio tgtd y que el otro nodo asuma los servicios y lance una orden de reinicio.
	Durante unos segundos las nuevas peticiones no son aceptadas.

Tabla 6: Pruebas realizadas sobre los equipos “Frontal-0X”.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
1	Matar el servicio haproxy en el “Frontal-01”.
	La VIP_v208 se migra a “Frontal-02” hasta que el servicio se restablece.
	Durante unos segundos las nuevas peticiones no son aceptadas.
2	Matar el servicio haproxy en el “Frontal-02”.
	La VIP_v200 se migra a “Frontal-01” hasta que el servicio se restablece.
	Durante unos segundos las nuevas peticiones no son aceptadas.
3	Apagar uno de los servidores.
	El servicio activo se migra al otro y las herramientas de HA de XenServer lo reinician.
	Durante unos segundos las nuevas peticiones no son aceptadas.
4	Matar el servicio keepalived en uno de los servidores.
	El servicio activo en ese servidor es migrado al otro. Y no se recupera hasta que interviene el administrador.
	Durante unos segundos las nuevas peticiones no son aceptadas.

Tabla 7: Pruebas realizadas sobre los equipos “Controlador-0X”.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
1	Matar cualquiera de los servicios soportados por el *cluster* del servidor activo.
	El servicio se migra al otro nodo y lanza una peticion de reinicio.
	Durante unos segundos las nuevas peticiones no son aceptadas.
2	Apagar la máquina activa.
	El servicio se migra al otro nodo y lanza una peticion de reinicio.
	Durante unos segundos las nuevas peticiones no son aceptadas.
3	Apagar el nodo que no se encuentra activo.
	El nodo activo ordena que se reinicie al otro nodo mediante la XenAPI.
	No se ve afectado.
4	Desactivar el sistema de ficheros compartido en el servidor activo.
	Esto provoca el fallo del servicio MySQL y que el otro nodo asuma los servicios y lance una orden de reinicio.
	Durante unos segundos las nuevas peticiones no son aceptadas.
5	Matar el servicio de cualquiera de las APIs.
	El balanceador de carga deja de enviar peticiones a ese servidor. El servicio tiene que ser restablecido por el administrador.
	No se ve afectado.
6	Matar el servicio de rabbitMQ en uno de los nodos.
	El balanceador de carga deja de enviar peticiones a ese servidor. El servicio tiene que ser restablecido por el administrador. En ese momento la información se replica.
	No se ve afectado.
7	Matar el servicio de memcached en uno de los nodos.
	El balanceador de carga deja de enviar peticiones a ese servidor. El servicio tiene que ser restablecido por el administrador. En ese momento la información se replica.
	No se ve afectado.

Tabla 8: Pruebas realizadas sobre los equipos “Computo-0X”.

Nº	**Prueba realizada**
	Comportamiento esperado
	Cómo afecta al servicio
1	Matar el servicio de metadata.
	Las máquinas que arranquen no van a poder obtener información de dicho servicio..
	Las máquinas controladas por el computo afectado no obtienen toda la funcionalidad.
2	Matar nova-compute
	Las máquinas actualmente en funcionamiento van a seguir pero no se permiten operaciones. El servicio tiene que ser restablecido por el administrador.
	Se ve afectada la capacidad para levantar nuevas máquinas si hay pocos recursos disponibles.
3	Matar nova-network
	Las máquinas actualmente en funcionamiento pierden la conectividad con el exterior y nuevas máquinas no son creadas. El servicio tiene que ser restablecido por el administrador.
	Se ve afectada la capacidad para levantar nuevas máquinas si hay pocos recursos disponibles y las capacidad de conectividad de las actuales.