UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

"OpenStack: Estudio y despliegue de herramientas de virtualización en la nube"

Carlos Campos Martín

Convocatoria Enero, 2019

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

"OpenStack: Estudio y despliegue de herramientas de virtualización en la nube"

Autor: Carlos Campos Martín

Tutor: José Moreno del Pozo

Co-Tutor/es: Francisco M. Andrés Hernández

Agradecimientos

A mis tutores gracias por haberme proporcionado los recursos necesarios para la realización de este trabajo y por su paciencia y comprensión en los problemas que surgieron. También quería dedicar este proyecto a mis padres y a mí pareja, pues gracias a ellos me he convertido en la persona que soy en la actualidad y me motivaron constantemente para alcanzar siempre mis metas. No olvido a mis compañeros durante estos años en la Universidad, suerte en vuestro camino.

ÍNDICE DE CONTENIDOS

1. Introducción 5
2. Tecnologías actuales de virtualización 6
   2.1. Definición de virtualización 6
   2.2. Técnicas de virtualización 7
   2.3. Hipervisor 8
   2.4. Ventajas y desventajas de la virtualización 10
   2.5. Software de virtualización más utilizado 11
3. Comparativa de los anteriores 14
4. Cloud Computing 14
   4.1. Características 16
   4.2. Tecnologías Cloud 16
5. Openstack 17
   5.1. Comparativa entre OpenStack y Proxmox 18
   5.2. Arquitectura conceptual y lógica de Openstack 19
6. Preparación del entorno OpenStack para virtualización en la nube 20
   6.1. Organización y planteamiento 20
   6.2. Pasos previos 21
   6.3. Instalación del S.O 21
   6.4. Instalación del hipervisor KVM 22
   6.5. Instalación de RabbitMQ 22
   6.6. Instalación de MySQL 23
   6.7. Instalación de la infraestructura OpenStack 23
   6.7.1. Servicio de autenticación y gestión de la identidad. Keystone 23
   6.7.2. Servicio de gestión y registro de imágenes. Glance 26
   6.7.3. Servicio de gestión de instancias (máquinas virtuales). Nova 30
   6.7.4. Servicio de gestión de redes. Neutron 33
   6.7.5. Servicio de Dashboard. Horizon 38
   6.8. Instalación de la infraestructura DevStack 43
   6.9. Funcionamiento básico de OpenStack 44
7. Ejemplo de creación de un nuevo proyecto 52
8. Adaptación y despliegue del proyecto SmartPolitech en OpenStack para virtualización en la nube 61
9. Glosario de términos 65
10. Referencias 67

ÍNDICE DE TABLAS

TABLA 1: VENTAJAS E INCONVENIENTES DE LAS MV 10
TABLA 2: LÍMITES DE XENSERVER 12
TABLA 3: COMPARATIVA DE SOFTWARE DE VIRTUALIZACIÓN 14
TABLA 4: ARQUITECTURAS UTILIZADAS 16

ÍNDICE DE FIGURAS

ILUSTRACIÓN 1: CARACTERÍSTICAS DE LAS MÁQUINAS VIRTUALES 7
ILUSTRACIÓN 2: VIRTUALIZACIÓN COMPLETA (MOLINA COBALLES, 2009) 7
ILUSTRACIÓN 3: PARA-VIRTUALIZACIÓN (MOLINA COBALLES, 2009) 8
ILUSTRACIÓN 4: VIRTUALIZACIÓN POR HARDWARE (MOLINA COBALLES, 2009) 8
ILUSTRACIÓN 5: VIRTUALIZACIÓN A NIVEL DE S.O. (SMALDONE, 2008) 8
ILUSTRACIÓN 6: HIPERVISOR TIPO 1 9
ILUSTRACIÓN 7: HIPERVISOR TIPO 2 9
ILUSTRACIÓN 8: DIAGRAMA DE LA ARQUITECTURA XEN (HTTPS://WWW.CITRIX.ES) 11
ILUSTRACIÓN 9: SERVIDOR ESX/ESXI (HTTP://WWW.VMWARE.COM/PDF/VIRTUALIZATION.PDF) 13
ILUSTRACIÓN 10: COMPUTACIÓN DISTRIBUIDA ENTRE EQUIPOS DISTRIBUIDOS 15
ILUSTRACIÓN 11: CLOUD COMPUTING 15
ILUSTRACIÓN 12: COMPAÑÍAS PLATINO Y ORO DE LA OPENSTACK FOUNDATION [13] 18
ILUSTRACIÓN 13: ARQUITECTURA CONCEPTUAL DE OPENSTACK 20
ILUSTRACIÓN 14: DISTRIBUCIÓN DE LOS SERVICIOS EN CADA NODO 21
ILUSTRACIÓN 15: PANTALLA DE LOGIN 45
ILUSTRACIÓN 16: VISIÓN GENERAL DE ADMINISTRADOR 45
ILUSTRACIÓN 17: PANEL DEL SISTEMA 46
ILUSTRACIÓN 19: HIPERVISORES DEL SISTEMA 46
ILUSTRACIÓN 18: PANEL DE IDENTIDAD 48
ILUSTRACIÓN 20: TOPOLOGÍA DE RED PÚBLICA CON CUATRO INSTANCIAS 51

1. Introducción

El siguiente Trabajo Fin de Grado pretende abarcar el aprendizaje obtenido sobre la tecnología OpenStack que proporciona una infraestructura como servicio (IaaS), consistente en una serie de proyectos relacionados entre sí que permiten el control de recursos de procesamiento, almacenamiento y red a través de un centro de datos, lo que permite a los administradores proporcionar, mediante un panel de control, recursos mediante una interfaz de red a los usuarios.

Antes de plantear en qué consiste OpenStack, ha sido necesario realizar un estudio sobre las principales tecnologías de virtualización, cómo funcionan, una comparativa de las más utilizados y qué aporta la computación en la nube como ventaja a los centros de procesamiento actuales.

Como práctica, se ha realizado un esbozo de cómo podría realizarse un despliegue del proyecto SmartPolytech en OpenStack, en colaboración con la UEx.

2. Tecnologías actuales de virtualización

En la actualidad, muchas empresas piensan cómo reducir el elevado coste que suponen sus CPDs (Centros de Procesamiento de Datos). Estos centros han ido creciendo siempre en función de las necesidades de negocio, ya sea potencia de procesamiento, mayor capacidad u otras necesidades concretas. Como consecuencia estos CPDs han llegado a un punto en el que se han vuelto inmanejables debido al número elevado de servidores en uso, el espacio que ocupan, la energía consumida. Como respuesta a la necesidad de las empresas de evolucionar sus CPDs tradicionales y disminuir costes, surge la idea de **virtualización**, permitiendo a las empresas mediante software, asignar a las aplicaciones que lo necesiten, un conjunto de recursos que separen a los administradores de las operaciones y centrarlos en los servicios, abstrayendo el hardware y eliminando la gestión física de los dispositivos. Esto contribuye a un menor uso de servidores y la optimización de su utilización.

2.1. Definición de virtualización

Algunas definiciones encontradas en la web son:

"La virtualización es el proceso de presentar un subconjunto de recursos físicos agrupados de forma lógica de tal forma que se obtengan beneficios sobre la configuración original." (Grupo IES, 2011).

"Se define como la abstracción de los recursos de un computador, llamada Hypervisor o VMM (Virtual Machine Monitor) que crea una capa de abstracción entre el hardware de la máquina física (host) y el sistema operativo de la máquina virtual (virtual machine, guest), siendo un medio para crear una versión virtual de un dispositivo o recurso, como un servidor, un dispositivo de almacenamiento, una red o incluso un sistema operativo, donde se divide el recurso en uno o más entornos de ejecución." (Wikipedia, 2016).

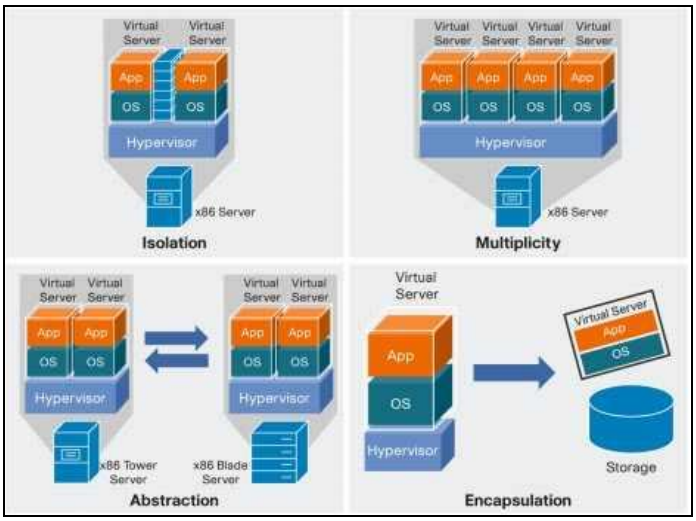


Ilustración 1: Características de las Máquinas Virtuales

2.2. Técnicas de virtualización

Principalmente podemos encontrar los siguientes modelos de virtualización:

- Virtualización nativa o completa: Un sistema operativo anfitrión es capaz de ejecutar sobre él sistemas operativos huésped sin modificar, simulando todo el hardware para ellos (utiliza hipervisores de tipo II).

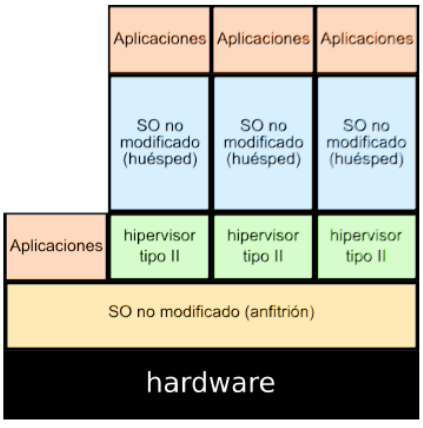


Ilustración 2: Virtualización Completa (Molina Coballes, 2009)

- Para-virtualización: No existe un sistema operativo anfitrión sino que se utiliza un software sobre el hardware (hipervisores de tipo I) sobre el que se ejecutan todas las máquinas virtuales.

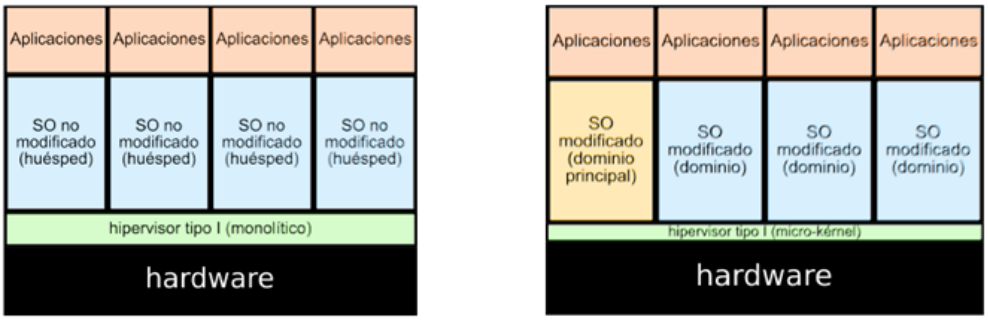


Ilustración 3: Para-virtualización (Molina Coballes, 2009)

- Virtualización asistida por hardware: es más eficiente que la virtualización completa al utilizar hardware adaptado a la virtualización.

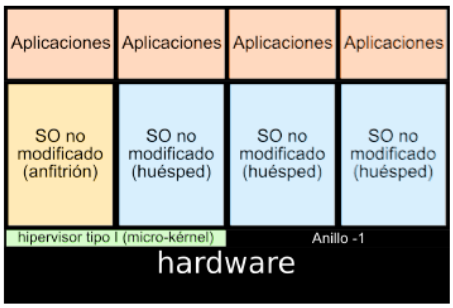


Ilustración 4: Virtualización por hardware (Molina Coballes, 2009)

- Virtualización a nivel de Sistema Operativo: Anfitrión y huésped comparten el mismo S.O.

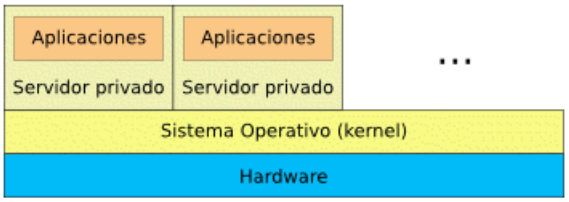


Ilustración 5: Virtualización a nivel de S.O. (Smaldone, 2008)

2.3. Hipervisor

Es el software de virtualización (VMM) y se ejecuta como parte del Sistema Operativo anfitrión. Sus instancias son Máquinas Virtuales. Aíslan unas máquinas de otras abstrayendo los recursos físicos de la máquina anfitriona para proporcionar una interfaz hardware, es decir, se encarga del control de virtualización.

Facilita el mantenimiento al estar desligado del hardware, pues podemos tener máquinas encapsuladas como ficheros y moverlas de un host a otro en caso de desastres sin modificarlas.

Los costes de consumo eléctrico y de mantenimiento se han visto reducidos, al reducir el número de máquinas físicas.

Podemos encontrar dos tipos de hipervisores:

Tipo 1 (bare-metal): software que se ejecuta directamente sobre el hardware. Se comparten los recursos computacionales de todo el sistema entre los sistemas operativos virtualizados.

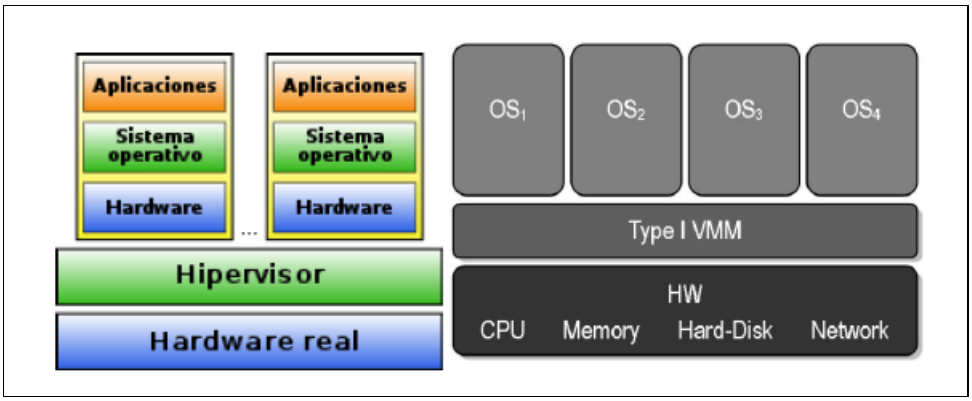


Ilustración 6: Hipervisor Tipo 1

Tipo 2 (hosted): software que se ejecuta sobre un Sistema Operativo. Los recursos computacionales se comparten por medio de la máquina virtual.

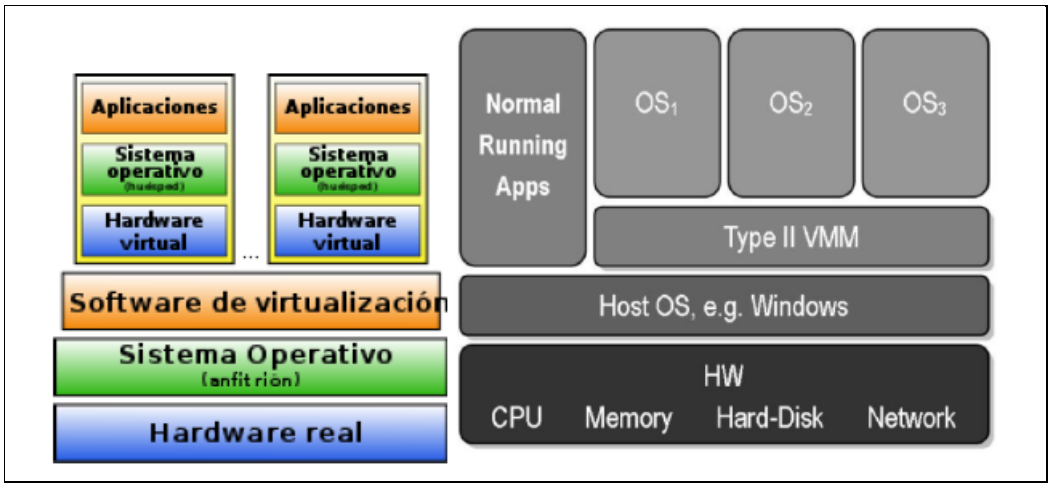


Ilustración 7: Hipervisor Tipo 2

2.4. Ventajas y desventajas de la virtualización

La virtualización permite crear en una máquina física varias máquinas virtuales con sus respectivos sistemas operativos y aplicaciones como si de una máquina real se tratase.

Algunas de sus principales ventajas son:

- Los recursos se ven como si fueran dedicados y se aprovechan mejor.
- Aislamiento respecto a los dispositivos.
- Mejora la tolerancia a fallos y la seguridad.
- Mejora en los costes.
- Mayor eficiencia y menor consumo.

Las desventajas que presentan los sistemas virtuales son:

- Problemas de compatibilidad con el hardware virtualizado.
- Varios sistemas dependen de un solo equipo.
- Dificultad en la configuración de servicios.
- Necesidad de recursos hardware del servidor.

La siguiente tabla muestra un resumen de lo anterior:

Tabla 1: Ventajas e inconvenientes de las MV

	Ventajas	Inconvenientes
MÁQUINAS VIRTUALES	Aprovechamiento de recursos Aislamiento hardware Seguridad Ahorro energético Eficiencia	Incompatibilidad con el hardware virtual Dificultad de configuración Varios sistemas dependen de un solo equipo Necesidad del hardware del servidor

2.5. Software de virtualización más utilizado

Citrix XenServer: es una plataforma, gratuita y de código abierto, de virtualización de nube, servidores y escritorios. Proporciona recursos de microprocesador y memoria RAM a las máquinas virtuales a través de un hypervisor de tipo I instalado en uno o varios hosts físicos. Abstrae del hardware físico del host los recursos a medida para cada una de las máquinas virtuales. A través de una interfaz virtual privilegiada (Control Domain o dom0) se pueden gestionar las herramientas que ofrece Xen.

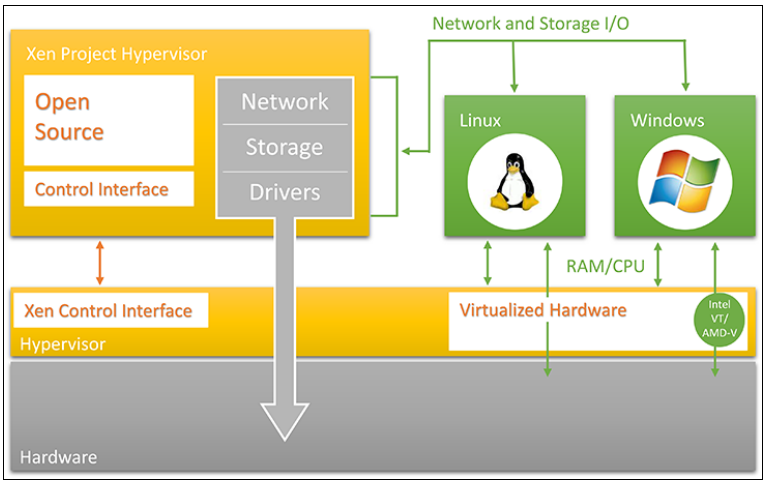


Ilustración 8: Diagrama de la arquitectura Xen (https://www.citrix.es)

Cada máquina virtual es físicamente un fichero almacenado (Storage Repository o SR) en un recurso en disco. Es posible además, agrupar lógicamente más de un host XenServer para ofrecer mayores prestaciones (almacenamiento, seguridad, red), es lo que se denomina Resource Pool.

Para optimizar el rendimiento y los costes, la memoria no utilizada por las máquinas virtuales se comparte entre las que están en ejecución en cada host.

XenServer también tiene sus limitaciones. Se resumen en la siguiente tabla:

Tabla 2: Límites de XenServer

Límites de los hosts	Límites de las MVs	Límites de Resource Pool
Procesadores lógicos: 160 vCPUs por host: 3250 Vms Windows concurrentes por host: 500 Vms Linux concurrentes por host: 650 RAM por Host: 1 TB NICS físicas: 16 NICS virtuales: 512 VLANS por host: 800 Discos virtuales concurrentes: 512 GPus por host: 8	CPUs virtuales por VM: 16 RAM por VM: 128 GB Tamaño disco duro virtual: 2 TB Discos por VM: 7 NICs por VM: 7	Hosts: 16 VLANS: 800 Paths a LUN: 8 Multipathed LUNS por host (usados por SRs): 75 Hosts por controlador de vswitch: 64

Proxmox VE: es un software (hipervisor tipo I) de virtualización de servidor de código abierto, potente y ligero. Soporta tanto KVM (Máquina Virtual basada en Kernel) como LXC (Virtualización basada en Contenedores Linux).

Responde de forma correcta a la asignación de servidores con cargas de trabajo extremas.

Puede efectuar backups de forma inmediata o de forma programada y su restauración es tan simple como arrancar el backup a restaurar y listo. También permite Snapshots en vivo, es decir, realizar una copia exacta (contenido en RAM, configuración y estado de los discos virtuales, etc).

Utiliza templates (plantillas) como base de las máquinas virtuales. El entorno ofrece una interfaz web de gestión con conexión remota a la consola de cada una de las máquinas virtuales alojadas, además de permitir la creación, arranque y parada de las máquinas virtuales.

Permite migraciones en caliente de máquinas virtuales pudiendo movilizar MVs entre cada servidor físico sin tener que apagar la Máquina Virtual.

Todos los nodos cuentan con su propio administrador web, aunque uno de ellos actúa como orquestador de los demás con el objetivo de centralizar el trabajo.

VMware: es un software hipervisor de tipo I (denominado ESXi) para virtualización de servidores producido por la empresa privada VMware, Inc.

El hipervisor ESX se apoya en un sistema Linux basado en Red Hat Enterprise Linux modificado para su ejecución y la de otros componentes de virtualización VMware.

En su versión gratuita proporciona las siguientes características:

- Hasta dos CPU´s físicas (sockets).
- Ilimitados cores por CPU.
- Ilimitada memoria RAM.
- Máximo de ocho vCPU por cada VM.
- No existe fecha de expiración del producto.

Y las siguientes limitaciones:

- No dispone de soporte del fabricante.
- El host ESXi no puede añadirse al Virtual Center.
- No dispone de vStorage APIs.

Aun así también es posible obtener licencias gratuitas VMware ESXi desde su web y activarlas para conseguir los números de serie para licenciar hosts VMware ESXi gratuitos. La instalación se realiza en modo trial y tiene una duración de 60 días. Si al expirar este tiempo no se aplica un nuevo número de serie sobre el host VMware ESXi no podremos iniciar VMs ubicadas en ese host VMware ESXi.

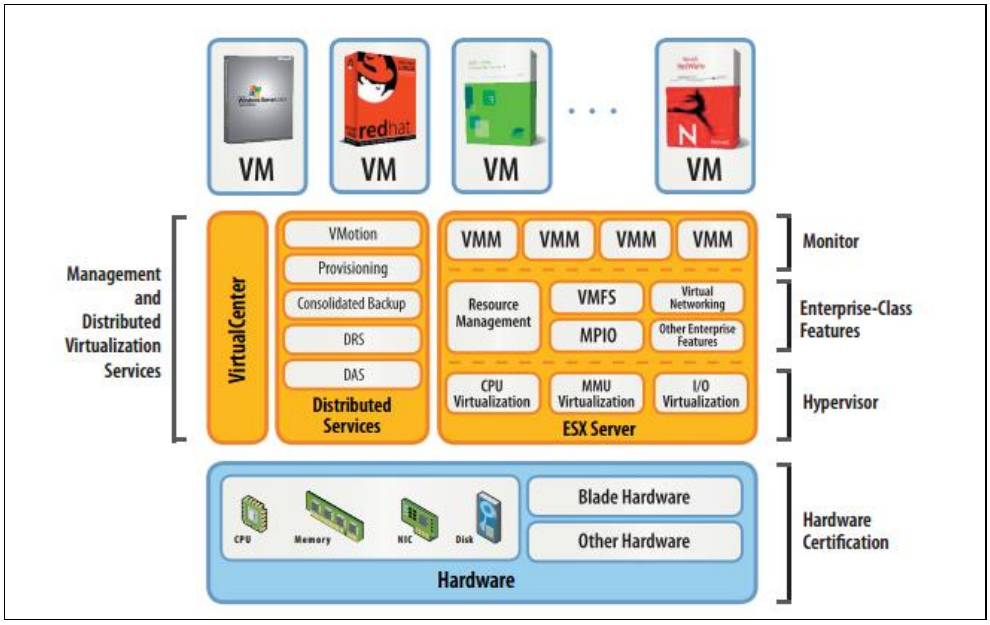


Ilustración 9: Servidor ESX/ESXi (http://www.vmware.com/pdf/virtualization.pdf)

3. Comparativa de los anteriores

La siguiente tabla muestra una comparación de las características más importantes de los competidores del mercado en la virtualización de servidores mencionados anteriormente.

Tabla 3: Comparativa de software de virtualización

	Proxmox VE	VMware (vSphere)	Citrix XenServer
Soporte para el sistema operativo invitado	Windows y Linux (KVM) Otros SO también son soportados y admitidos por la comunidad	Windows, Linux, UNIX	La mayor parte de los SO Windows. Soporte para Linux limitado.
Código abierto	Sí	No	Sí
Linux Containers (LXC) (conocido como SO de virtualización)	Sí	No	No
Interfaz para el control centralizado	Sí	Sí, pero requiere un servidor de administración dedicada (o VM)	Sí
Precisa subscripción	Sí, precio de subscripción para la habilitación de todas las características	No	No
Alta disponibilidad	Sí	Sí	Sí
Snapshops en vivo	Sí	Sí	Sí
Hipervisor Bare-Metal	Sí	Sí	Sí
Migración en vivo	Sí	Sí	Sí
Max. RAM y CPU por host	160 CPU / 2 TB RAM	160 CPU / 2 TB RAM	?

4. Cloud Computing

Actualmente las arquitecturas basadas en Centros de Procesamiento de Datos actuales ya no están preparadas para las demandas, cada vez mayores, de potencia de procesamiento y/o almacenamiento, debido principalmente a que estos sistemas son muy rígidos, escalabilidad baja y complicados de gestionar, además de un alto coste de mantenimiento.

La arquitectura *grid computing* se caracteriza por el uso orquestado de recursos de cómputo, almacenamiento o servicios puntuales que a diferencia de los mainframes de los CPD, se encuentran distribuidos geográficamente y no necesitan de un gestor

centralizado. Los recursos pueden, de hecho, tener un carácter heterogéneo (diferentes S.O, arquitecturas, etc.), utilizando Internet para la interconexión de los sistemas. Todo esto provoca una disminución de la seguridad, el rendimiento, la velocidad de comunicación o la tolerancia a fallos.

Ilustración 10: Computación distribuida entre equipos distribuidos

Como respuesta a los inconvenientes de la arquitectura *grid computing*, aparece la computación en la nube o *cloud computing*. Esta ofrece una infraestructura de software remoto, lo que permite un ahorro en los costes. Los programas utilizados están almacenados en servidores externos a los que se puede acceder desde Internet. Si uno de nuestros equipos tiene problemas, el software seguirá estando disponible. ("La diferencia entre virtualización y computación en la nube", 2011).

Ilustración 11: Cloud computing

Virtualización y *cloud computing* caminan de la mano en muchas ocasiones. Según explica Mike Adams, director de marketing de VMware, *"El cloud computing consiste en la entrega de recursos informáticos compartidos a través de software o datos y que son entregados como un servicio de demanda a través de Internet"*

La diferencia fundamental entre virtualización y *cloud computing* es que con la virtualización el software manipula el hardware, mientras que *cloud computing* consiste en el servicio resultante de la manipulación. (Pérez, 2014)

La siguiente tabla presenta las ventajas y desventajas de las arquitecturas utilizadas:

Tabla 4: Arquitecturas utilizadas (Guijarro Verdura, 2014)

Arquitectura	Características	Ventajas	Inconvenientes
CPD	Concentración de *mainframes* y sus redes de comunicación en una única ubicación	❑ Gran potencia de cómputo ❑ Segura ❑ Veloz en la transmisión de datos	❑ Arquitectura muy rígida ❑ Difícilmente escalable ❑ Caros de gestionar y mantener
Grid computing	Conjunto heterogéneo de recursos colectivos distribuidos geográficamente	❑ Flexible ❑ Escalable	❑ Merma de la seguridad ❑ Pérdida de rendimiento
Cloud computing	Sistema que permite ofrecer servicios de computación a través de Internet (IaaS, Infraestructura como Servicio)	❑ Flexible ❑ Escalable ❑ Segura ❑ Económica	❑ Dependencia de la disponibilidad de acceso a Internet

4.1. Características

La computación en la nube o *cloud computing* permite adquirir recursos de la infraestructura de manera rápida y barata.

Los costes son muy reducidos y se limitan a gastos operativos.

Permiten a los usuarios, independientemente de su ubicación y del dispositivo o equipo utilizado acceder a los sistemas, siempre que se disponga al menos de una conexión a Internet.

Permite compartir recursos entre un gran número de usuarios, lo que permite una mayor eficiencia (10-20%) y una mayor escalabilidad.

La utilización eficiente de los recursos y sistemas hacen que sea más sostenible.

4.2. Tecnologías Cloud

Algunas de las plataformas de computación en la nube más importantes son:

Amazon Web Services (AWS): es una colección comercial de servicios web (colección de servicios de escritorio remoto) que en conjunto forman una plataforma cloud.

El núcleo lo conforma Amazon EC2 o también denominado S3 y permite a los clientes lanzar sus procesos en computadoras virtuales alquiladas por horas y almacenar sus datos en línea.

vCloud: desarrollado por VMware, permite a los clientes migrar procesos ejecutados en hipervisores VMware locales a hipervisores VMware localizados en clouds remotos, brindando una mayor potencia y flexibilidad de computación.

Eucalyptus: su principal característica es la capacidad de despliegue de IaaS debido a su compatibilidad mediante la API S3 de Amazon Web Services (AWS) con sus servicios.

OpenNebula: nació de la mano de un equipo de investigación de Arquitecturas de Sistemas Distribuidos con sede en la Universidad Complutense de Madrid. Fue modificando su licencia hasta hacerse completamente libre. Esta tecnología no depende de ningún tipo concreto de hipervisor y se adapta perfectamente a las infraestructuras ya existentes.

CloudStack: es una plataforma bajo licencia Apache v2 Softwareopen source para crear, gestionar y desplegar IaaS. Soporta varios hipervisores como KVM, vSphere y XenServer/XCP y compatible con las API de Amazon.

5. Openstack

Es una colección de proyectos *cloud computing* cuya misión es cubrir el ciclo completo para despliegues *cloud* tanto privadas como públicas. Surge en 2010 de la mano de las organizaciones Rackspace Cloud y la agencia espacial norteamericana, NASA para proporcionar IaaS, actualmente el proyecto se gestiona por la fundación sin ánimo de lucro Openstack Foundation, encargada de divulgar la distribución, el desarrollo y la adopción de Openstack como sistema operativo *cloud*. La fundación se financia a través de compañías organizadas en categorías corporativas (platino, oro y sponsors corporativos).

Ilustración 12: Compañías platino y oro de la Openstack Foundation [13]

La plataforma Openstack utiliza licencia Apache v2 y es Open Source. La comunidad Openstack colabora para lanzamientos de desarrollo con una frecuencia semestral.

Esta tecnología se impone claramente sobre el resto de plataformas IaaS gracias al apoyo de sus usuarios a pesar de ser una tecnología prácticamente reciente y presentando enormes expectativas de futuro.

5.1. Comparativa entre OpenStack y Proxmox

Aunque la plataforma OpenStack es un producto más sofisticado y acto para la creación de nubes públicas y privadas, también destaca Proxmox por su sencillez. Una cosa importante a destacar en este software de virtualización es el corto periodo de tiempo necesario para crear un servidor (10 minutos por lo general). Al igual que OpenStack, es necesario realizar una planificación de la arquitectura de virtualización (un mapa de VMs) que proveerá la nube para la configuración de redes, storage, partitions, firewalls, security. Proxmox utiliza PaaS stack (plataforma como servicio) como WSO2 o RedHat OpenShift para tal fin. No olvidemos que OpenStack ya cuenta con APIs para este cometido y que lo hace más adecuado para realizar este tipo de despliegues, dado que Openstack es un conjunto de proyectos pensados para Cloud Computing y Proxmox es una herramienta para virtualización (bare-metal).

5.2. Arquitectura conceptual y lógica de Openstack

OpenStack utiliza API públicas (proyectos) para comunicar con todos sus servicios, aunque no es necesaria la instalación de todos ellos, únicamente son necesarios unos cuantos servicios para dar respuesta a unas necesidades particulares.

Los proyectos más importantes de OpenStack son los siguientes:

Horizon: provee un servicio de portal web (Dashboard) para interactuar con los demás servicios desplegados en OpenStack (los proporcionados por los demás proyectos desplegados).

Nova: es el responsable de gestionar el ciclo de vida completo de las instancias de cómputo (Compute) en el entorno OpenStack.

Neutron: provee una API para definir configuraciones de red (Networking) y ofrecer conectividad con los demás servicios OpenStack desplegados.

Swift: almacena y recupera datos (Object Storage) a través de una API REST, basada en HTTP.

Cinder: permite la creación y gestión de dispositivos de almacenamiento (Block Storage) para las instancias en curso.

Keystone: Ofrece un servicio de autenticación y autorización a los servicios desplegados.

Glance: es el encargado de almacenar y recuperar las imágenes en disco de las máquinas virtuales (las utilizadas por el servicio Compute al crear instancias).

Ceilometer: actúa como monitor de la nube desplegada con OpenStack pudiendo realizar tareas de facturación, comparativas de mercado, escalabilidad, etc.

Heat: permite la orquestación con otros servicios de alto nivel como los proporcionados por AWS (Amazon Web Services) en la nube, utilizando plantillas con formato nativo HOT (Heat Orchestration Template).

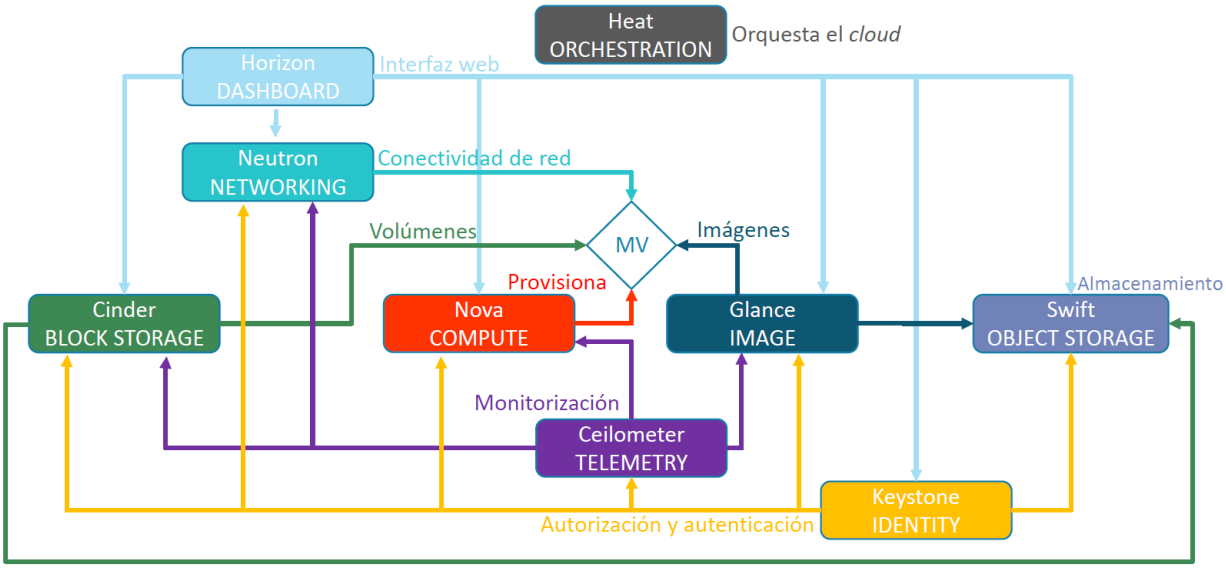


Ilustración 13: Arquitectura conceptual de OpenStack

6. Preparación del entorno OpenStack para virtualización en la nube

Con el fin de entender mejor la infraestructura procederemos a instalar paso a paso cada uno de los módulos OpenStack. Posteriormente utilizaremos DevStack (entorno OpenStack adaptado a un entorno de Desarrollo y Pruebas) que ya contiene todo lo necesario para lanzar instancias y desarrollar pruebas.

6.1. Organización y planteamiento

Dependiendo del tamaño de nuestra organización, debemos tener en cuenta de qué forma vamos a organizar nuestro Cloud, qué servidores se ocuparán del cómputo, almacenamiento, control de red, etc. OpenStack necesita de dos a cuatro CPUs, un mínimo de ocho gigas de RAM, un mínimo de cien gigabytes de almacenamiento y dos tarjetas de red. Para esta instalación se han utilizado dos servidores de ocho núcleos, treinta y dos gigas de RAM y trescientos gigas de disco duro cada uno y que distribuiremos de la siguiente forma:

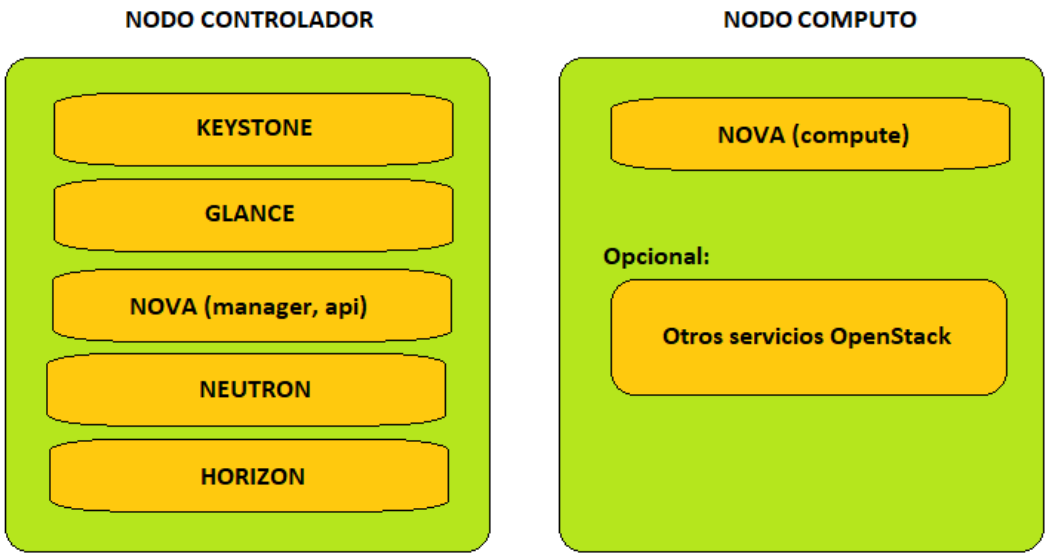


Ilustración 14: Distribución de los servicios en cada nodo

De aquí en adelante llamaremos al servidor controlador "*Florencia"* y al servidor de cómputo "*Toscana"*.

6.2. Pasos previos

Antes de implantar la infraestructura OpenStack hemos instalado algunos recursos que serán necesarios para la correcta instalación. Estos recursos son el Sistema operativo, el hipervisor KVM y el gestor de base de datos MySQL.

6.3. Instalación del S.O

Hemos instalado en los dos servidores, *Ubuntu 14.04 (LTS)* Server aunque OpenStack da soporte a otras distribuciones como RedHat, CentOS, openSUSE y SUSE. También puede instalarse en otros SOs aunque estos son los que poseen una mayor cantidad de guías y tutoriales de instalación. En la instalación de Ubuntu hemos habilitado la opción SSH para permitir la comunicación remota. De una forma u otra (directamente en la máquina o mediante SSH), ejecutaremos los comandos siguientes para tener totalmente actualizados los paquetes del sistema.

```
apt-get update
apt-get upgrade
```

6.4. Instalación del hipervisor KVM

Una vez instalado el sistema operativo, debemos instalar el hypervisor (KVM en nuestro caso) en el servidor Toscana, el servidor que hemos asignado previamente para el cómputo. Todos los nodos de computación deberán utilizar el hypervisor que hayamos elegido. Para ello debemos comprobar si soportan virtualización x86. Podemos comprobarlo con este comando:

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

Comprobaremos que soporta virtualización si el resultado del comando es un valor mayor a 0 (y que esté activada en BIOS la virtualización). En Ubuntu podemos utilizar el comando kvm-ok que nos indicará si KVM puede ser utilizado.
Una vez realizadas estas comprobaciones, instalaremos el módulo de kernel kvm, el toolkit de libvirt para la interacción con el hipervisor con el siguiente comando:

```
apt-get install qemu-kvm libvirt-bin bridge-utils
```

Aunque en nuestro caso no ha habido problemas a la hora de instalar KVM, damos por hecho que se sabe instalar correctamente el hipervisor, no obstante puede verse su instalación detallada en la bibliografía.

6.5. Instalación de RabbitMQ

OpenStack necesita de un gestor que se encargue de la comunicación de sus módulos, para ello utilizaremos RabbitMQ, aunque se puede utilizar Qpid o ZeroMQ. El paquete lo instalamos con la siguiente instrucción:

```
apt-get install rabbitmq-server
```

Añadimos el usuario *openstack*.

```
rabbitmqctl add_user openstack RABBIT_PASS
```

Podemos utilizar otra contraseña distinta a *RABBIT_PASS*.

Proporcionamos permisos de lectura y escritura al usuario openstack:

```
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Si obtenemos un *done* tras cada instrucción es que todo ha ido correctamente.

6.6. Instalación de MySQL

Para guardar la información de los servicios de OpenStack utilizaremos MySQL. Instalaremos la versión Server en el servidor Florencia con esta instrucción.

```
apt-get install python-mysqldb mysql-server
```

Para el resto de nodos de la nube (en este caso solo el servidor Toscana) instalamos el cliente de MySQL con este comando.

```
apt-get install python-mysqldb
```

6.7. Instalación de la infraestructura OpenStack

Con todo lo realizado anteriormente ya tendremos preparados los entornos de los dos servidores para comenzar a instalar los servicios de OpenStack. Primero instalaremos los servicios en el servidor *Florencia*.

6.7.1. Servicio de autenticación y gestión de la identidad. Keystone

Comenzaremos instalando el servicio de identidad Keystone. Nos conectamos a Mysql y creamos una nueva base de datos aplicando los siguientes permisos.

```
$ mysql -u root -p

CREATE DATABASE keystone;

GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
```

Podemos sustituir la contraseña *KEYSTONE_DBPASS* por la que queramos.

Preparamos la configuración inicial generando un token aleatorio (un número aleatorio) para el administrador:

```
$ openssl rand -hex 10
```

Deshabilitaremos el inicio automático para que Keystone no se arranque tras la instalación:

```
# echo "manual" > /etc/init/keystone.override
```

Instalamos Keystone, Apache y un módulo para este:

```
# apt-get install keystone apache2 libapache2-mod-wsgi
```

Editamos el fichero de config de Keystone. Podemos utilizar el editor *nano*. Sustituimos *ADMIN_TOKEN* por el número generado anteriormente.

```
[DEFAULT]
...
admin_token = ADMIN_TOKEN
```

En el mismo fichero de configuración ponemos la url de configuración de mysql:

```
[database]
...
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

Asignamos el provider fernet que ofrece seguridad encriptando los mensajes entre los módulos OpenStack y guardamos todos los cambios del fichero:

```
[token]
...
provider = fernet
```

Cargamos la base de datos creada mediante la siguiente instrucción:

```
# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

Iniciamos las claves de Fernet:

```
# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
```

Hasta este punto Keystone estaría configurado. Para poder recibir peticiones (REST) debemos configurar Apache para ello editamos el fichero de configuración /etc/apache2/apache2.conf y añadiendo el nombre del servidor al nodo controlador:

```
ServerName controller
```

Debemos definir los permisos de acceso a la base de datos de Keystone, para ello creamos el siguiente fichero *wsgi-keystone.conf* en la ruta /etc/apache2/sites-available/ con el siguiente contenido:

```
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone group=keystone
 display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone group=keystone
display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>
```

Por último creamos el enlace siguiente para que el servicio pueda recibir peticiones REST y reiniciamos el servicio para aplicar los cambios:

```
# ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled
```

```
# service apache2 restart
```

6.7.2. Servicio de gestión y registro de imágenes. Glance

Antes de instalar Glance, con el fin de salvar la información de las imágenes para las instancias de máquinas virtuales (arquitectura, formato, tamaños, etc) añadimos una nueva base de datos para el servicio:

```
$ mysql -u root -p

CREATE DATABASE glance;

GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
```

Podemos sustituir la contraseña *GLANCE_DBPASS* por la que queramos.

Antes de continuar con la creación de credenciales, vamos a realizar un paréntesis dado que para que las siguientes instrucciones se realicen correctamente es necesario crear algunas variables de entorno. Estas variables deben referenciarse cada vez que se ejecuten comandos de administrador. Para facilitar esta tarea y así evitar errores, nos crearemos un fichero que denominaremos *admin-openrc* con el siguiente contenido:

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

ADMIN_PASS sería la contraseña del usuario definido admin que pertenece al proyecto (OS_TENANT) admin del servicio definido en controller.

Una vez creado el fichero y cada vez que queramos tener acceso a comandos de administrador, solo debemos escribir lo siguiente:

```
$ . admin-openrc
```

Ahora podemos continuar creando las credenciales de Glance.

Creamos como credenciales de Glance un usuario, un rol y la entidad del servicio Glance:

```
$ openstack user create --domain default --password-prompt glance
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | e38230eeff474607805b596c91fa15d9 |
| name      | glance                           |
+-----------+----------------------------------+
```

```
$ openstack role add --project service --user glance admin
```

```
$ openstack service create --name glance \
  --description "OpenStack Image" image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

Internamente existen unas entradas en la base de datos de Keystone (endpoints) que se utilizan para obtener la información de otros módulos del sistema (disponibilidad, servidor donde se encuentra, módulo, etc). Generamos para Glance los API endpoints cuya finalidad es exactamente la misma:

```
$ openstack endpoint create --region RegionOne \
  image public http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 340be3625e9b4239a6415d034e98aace |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  image internal http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | a6e4b153c2ae4c919eccfdbb7dceb5d2 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  image admin http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 0c37ed58103f4300a84ff125a539032d |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Realizadas estas configuraciones, iniciamos la instalación del servicio Glance:

```
# apt-get install glance
```

Editamos la línea connection del fichero /etc/glance/glance-api.conf:

```
[database]
...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

En el mismo fichero configuramos los datos de acceso al servicio Keystone y editamos las rutas de procedencia de los ficheros de imágenes y donde los vamos a almacenar:

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = GLANCE_PASS

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

Sustituimos la contraseña *GLANCE_PASS* por la que introdujimos en el servicio Keystone.

```
[glance_store]
...
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Editamos ahora el fichero *glance-registry.conf* de la ruta /etc/glance:

```
[database]
...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

Damos acceso a la base de datos y sustituimos la contraseña *GLANCE_DBPASS* por la que elegimos para la base de datos de Glance.

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = GLANCE_PASS

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

Poblamos la base de datos:

```
# su -s /bin/sh -c "glance-manage db_sync" glance
```

Reiniciamos los servicios de Glance:

```
# service glance-registry restart
# service glance-api restart
```

6.7.3. Servicio de gestión de instancias (máquinas virtuales). Nova

Creamos una nueva base de datos para Nova en el nodo controlador y aportamos los permisos que procedan:

```
$ mysql -u root -p


CREATE DATABASE nova_api;
CREATE DATABASE nova;
```

```
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

Recordemos que podemos cambiar la cadena *NOVA_PASS* por la contraseña que queramos.

Hacemos referencia al fichero que creamos con las variables de entorno para tener acceso a los comandos de administrador.

```
$ . admin-openrc
```

Ahora creamos un usuario para Nova

```
$ openstack user create --domain default \
  --password-prompt nova
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | 8c46e4760902464b889293a74a0c90a8 |
| name      | nova                             |
+-----------+----------------------------------+
```

Añadimos el rol *admin*

```
$ openstack role add --project service --user nova admin
```

Creamos el servicio de identidad de Nova

```
$ openstack service create --name nova \
  --description "OpenStack Compute" compute
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 060d59eac51b4594815603d75a00aba2 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

Creamos los endpoints

```
$ openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 3c1caa473bfe4390a11e7177894bcc7b        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | e3c918de680746a586eac1f2d9bc10ab        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 38f7af91666a47cfb97b4dc790b94424        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | e702f6f497ed42e6a8ae3ba2e5871c78        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

Una vez realizado todo esto podemos comenzar con la instalación del módulo

```
# apt-get install nova-api nova-conductor nova-consoleauth \
  nova-novncproxy nova-scheduler
```

Editamos la siguiente configuración en el fichero de la ruta /etc/nova/nova.conf

```
[DEFAULT]
...
enabled_apis = osapi_compute,metadata
```

```
[api_database]
...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
```

Sustituimos *NOVA_DBPASS* por la contraseña que pusimos anteriormente en la creación de la base de datos.

También debemos configurar el gestor de RabbitMQ, debemos sustituir *RABBIT_PASS* por la contraseña que pusimos al usuario *openstack.*

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS
```

Reemplazamos *NOVA_PASS* por el pass que se introdujos en el servicio de Identidad.

Añadimos en la sección [DEFAULT] la siguiente línea.

```
[DEFAULT]
...
my_ip = 10.0.0.11
```

La IP introducida se corresponde, al igual que en las anteriores configuraciones, a la de la red de gestión.

```
[vnc]
...
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip

[glance]
...
api_servers = http://controller:9292

[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

Salvamos el fichero de configuración y a continuación poblamos la base de datos de nova con las siguientes instrucciones:

```
# su -s /bin/sh -c "nova-manage api_db sync" nova
# su -s /bin/sh -c "nova-manage db sync" nova
```

Finalmente reiniciamos Nova y sus componentes.

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

6.7.4. Servicio de gestión de redes. Neutron

Procedemos a instalar y configurar Neutron. Como en los módulos anteriores añadimos una nueva base de datos SQL para Neutron:

```
$ mysql -u root -p

CREATE DATABASE neutron;

GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
```

No olvidemos que podemos utilizar otra contraseña sustituyendo *NEUTRON_DBPASS* por la que nos interese.

Ejecutamos admin-openrc para tener acceso a los comandos de administrador:

```
$ . admin-openrc
```

Creamos las credenciales del servicio, añadimos el rol *admin* y creamos el servicio de identidad para Neutron:

```
$ openstack user create --domain default --password-prompt neutron
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | b20a6692f77b4258926881bf831eb683 |
| name      | neutron                          |
+-----------+----------------------------------+

$ openstack role add --project service --user neutron admin

$ openstack service create --name neutron \
  --description "OpenStack Networking" network
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | f71529314dab4a4d8eca427e701d209e |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+
```

Antes de comenzar con la instalación del módulo Neutron y sus componentes, debemos indicar los API endpoints para Neutron:

```
$ openstack endpoint create --region RegionOne \
  network public http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 85d80a6d02fc4b7683f611d7fc1493a3 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  network internal http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 09753b537ac74422a68d2d791cf3714f |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+

$ openstack endpoint create --region RegionOne \
  network admin http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 1ee14289c9374dffb5db92a5c112fc4e |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Hasta este punto tendríamos el entorno preparado para instalar Neutron. Hay que destacar que aquí podríamos configurar la instalación para ofrecer el nivel 3 del modelo OSI de manera virtual, además de conexión de las instancias con el exterior (acceso a Internet), permitiendo diseñar redes sin la adquisición de nuevos equipos.

En este caso realizaremos la instalación básica que permite a las instancias de las máquinas virtuales comunicarse con el exterior (acceso a Internet).

Instalamos el módulo y sus componentes:

```
# apt-get install neutron-server neutron-plugin-ml2 \
  neutron-linuxbridge-agent neutron-dhcp-agent \
  neutron-metadata-agent
```

Editamos el fichero de configuración /etc/neutrón/neutrón.conf de la iguiente forma:

```
[database]
...
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
```

Debemos reemplazar *NEUTRON_DBPASS* por la contraseña que pusimos en la base de datos.

```
[DEFAULT]
...
core_plugin = ml2
service_plugins =

[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

Cambiamos *RABBIT_PASS* por la contraseña del usuario *openstack* que pusimos en la instalación de RabbitMQ.

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

Cambiamos *NEUTRON_PASS* por la contraseña del usuario neutrón del servicio de identidad.

```
[DEFAULT]
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True

[nova]
...
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

Reemplazamos *NOVA_PASS* por la contraseña del usuario nova del servicio de identidad.

Ahora editamos el fichero ml2_conf.ini ubicado en /etc/neutrón/plugins/ml2/ para definir los tipos de redes que tenemos (VLAN y flat), habilitar el linuxbridge para intercomunicarnos con la red del proveedor y el ipset para mejor eficiencia de las reglas de los grupos de seguridad.

```
[ml2]
...
type_drivers = flat,vlan

[ml2]
...
tenant_network_types =

[ml2]
...
mechanism_drivers = linuxbridge

[ml2]
...
extension_drivers = port_security

[ml2_type_flat]
...
flat_networks = provider

[securitygroup]
...
enable_ipset = True
```

Editamos el fichero linuxbridge_agent.ini de la ubicación /etc/neutrón/plugins/ml2:

```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

PROVIDER_INTERFACE_NAME define la interfaz que nos dará acceso al proveedor de red, con lo que sustituimos este nombre por el de la interfaz que nos da acceso a internet (recordar que disponemos de dos interfaces de red, una dedicada a la red interna y otra para la red externa). Tendrá un nombre similar a eth0.

Configuramos el agente DHCP para la red virtual:

```
[vxlan]
enable_vxlan = False


[securitygroup]
...
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
```

Para dar credenciales a las instancias, configuramos el agente de metadatos:

```
[DEFAULT]
...
nova_metadata_ip = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

METADATA_SECRET es una clave para el proxy de metadatos.

Configuramos Nova para que utilice Neutron:

```
[neutron]
...
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
```

Reemplazamos *NEUTRON_PASS* y *METADATA_SECRET* por la contraseña del usuario *neutron* en el servicio de identidad y la que hemos introducido para el proxy de metadatos.

Poblamos y definimos la base de datos:

```
# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \
  --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

Reiniciamos Neutron y sus componentes:

```
# service nova-api restart


# service neutron-server restart
# service neutron-linuxbridge-agent restart
# service neutron-dhcp-agent restart
# service neutron-metadata-agent restart
```

6.7.5. Servicio de Dashboard. Horizon

Como último paso en el nodo controlador, instalaremos el módulo Dashboard.

Para ello escribimos la siguiente instrucción:

```
# apt-get install openstack-dashboard
```

Editamos el fichero de configuración local_settings.py en /etc/openstack-dashboard para definir lo siguiente:

El nodo que ejecutará Dashboard (en nuestro caso el nodo Controlador).

```
OPENSTACK_HOST = "controller"
```

Los permisos para que todos los host accedan al Dashboard.

```
ALLOWED_HOSTS = ['*', ]
```

La configuración del servicio de almacenamiento de sesiones (memcached).

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
    'default': {
         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
         'LOCATION': 'controller:11211',
    }
}
```

Activamos la API de identidad v3.

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

Activamos el soporte para dominios.

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

Configuramos las versiones de las API.

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}
```

Definimos el dominio y rol por defecto para los usuarios que creemos mediante Dashboard.

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
```

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

Deshabilitamos el nivel 3 de los servicios de red.

```
OPENSTACK_NEUTRON_NETWORK = {
    ...
    'enable_router': False,
    'enable_quotas': False,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_lb': False,
    'enable_firewall': False,
    'enable_vpn': False,
    'enable_fip_topology_check': False,
}
```

Reiniciamos Apache para aplicar los cambios.

```
# service apache2 reload
```

Hasta aquí el nodo controlador *Florencia* estaría listo y configurado para funcionar.

A continuación seguiremos con la instalación y configuración del nodo que hemos destinado para computación *Toscana*.

Comenzamos instalando el cliente de OpenStack con la instrucción:

```
# apt-get install python-openstackclient
```

Instalamos el paquete Nova que permitirá la ejecución de instancias.

```
# apt-get install nova-compute
```

Editamos el fichero nova.conf ubicado en la ruta /etc/nova

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS
```

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Sustituimos las variables *RABBIT_PASS* por la contraseña que pusimos al usuario *openstack* de RabbitMQ (en el nodo controlador), *NOVA_PASS* por la que se asignó al usuario *nova* (en el nodo controlador) y MANAGEMENT_INTERFACE_IP_ADDRESS por la IP de la interfaz de red que utilizamos para gestión.

```
[DEFAULT]
...
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[vnc]
...
enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html

[glance]
...
api_servers = http://controller:9292
```

```
[oslo_concurrency]
...
lock_path = /var/lib/nova/tmp
```

Comprobamos si el tipo de aceleración hardware que soportarán las máquinas virtuales es el estándar con el siguiente comando:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Si devuelve valor 1 o mayor no necesitamos hacer más configuraciones. En el caso de que la instrucción devuelva un 0, debemos editar el fichero nova.conf y añadir la siguiente configuración:

```
[libvirt]
...
virt_type = qemu
```

El valor qemu se utiliza para equipos que no disponen de soporte nativo para virtualización.

Guardamos los cambios y reiniciamos los servicios para que surtan efecto:

```
# service nova-compute restart
```

Por último, para terminar con la configuración instalamos el módulo que gestionará la red de las instancias:

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

Editamos el fichero de configuración en la ruta /etc/neutron/neutron.conf:

```
[DEFAULT]
...
rpc_backend = rabbit

[oslo_messaging_rabbit]
...
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

Sustituimos *RABBIT_PASS* por la contraseña que fijamos para el usuario *openstack* en RabbitMQ del controlador y *NEUTRON_PASS* por la contraseña que pusimos al usuario *neutron* en el servicio de identidad del nodo controlador.

Como hicimos con el nodo controlador, configuraremos la opción de interconectar las instancias con el exterior editando el fichero ubicado en /etc/neutron/plugins/ml2/linuxbridge_agent.ini.

```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

```
[vxlan]
enable_vxlan = False


[securitygroup]
...
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Reemplazamos *PROVIDER_INTERFACE_NAME* por la interfaz que da acceso a Internet (recordemos que disponemos de dos interfaces, una para la gestión interna y otra para el acceso a Internet).

Editamos el fichero nova.conf en la ruta /etc/nova para que utilice Neutron para acceder al exterior:

```
[neutron]
...
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

Reemplazamos NEUTRON_PASS por la contraseña que pusimos en el servicio de identidad al usuario *neutron* (en el controlador).

Reiniciamos para que los cambios realizados surtan efecto:

```
# service nova-compute restart

# service neutron-linuxbridge-agent restart
```

Con esto daríamos por finalizada la instalación de OpenStack de forma manual cuya arquitectura estaría formada por un nodo controlador *Florencia* y un nodo para computación, *Toscana*.

Este tipo de instalación presenta una gran flexibilidad puesto que podríamos añadir sin problema más nodos de computación, así como inicialmente haber escogido otro tipo de base de datos, gestor cola de mensajes, etc.

A continuación veremos una forma de instalar OpenStack de forma rápida, a través de la distribución DevStack para entornos reducidos y académicos, y conocer las características principales que presenta el Dashboard de OpenStack.

6.8. Instalación de la infraestructura DevStack

Para la instalación de DevStack se ha utilizado un único nodo con las siguientes características:

- Procesador: Intel Core I7
- Modelo: Acer Aspire
- Velocidad del procesador: 2.0Ghz, hasta 3.1GHz con TurboBoost
- Numero de procesadores: 1
- Total de Cores: 4
- Memoria: 16GB DDR3

Comenzamos con tener una instalación limpia de Ubuntu Server 14.04 LTS en nuestro nodo. Damos por hecho que se sabe realizar una instalación de este sistema operativo. Podemos obtener este S.O del siguiente enlace:

http://releases.ubuntu.com/trusty/

Si no hemos instalado git en la instalación del S.O, podemos instalarlo abriendo un terminal y escribir el siguiente comando:

```
$ apt-get install git
```

1. Descargamos devstack del repositorio de git:

```
git clone https://git.openstack.org/openstack-dev/devstack
```

2. Una vez termine la instrucción, tendremos en nuestro directorio actual una carpeta llamada devstack. Esta carpeta tiene todo lo necesario para realizar la instalación de OpenStack. Accedemos a ella y ejecutamos el siguiente script:

```
./stack.sh
```

La instalación puede tardar unos minutos, dependiendo del equipo. En el transcurso de la instalación debemos concretar la contraseña para la base de datos, la del servicio de identidad y del gestor de colas de mensajes entre módulos. Una vez terminada esta instalación podremos acceder al Dashboard de OpenStack mediante el navegador ingresando la dirección IP del equipo.

6.9. Funcionamiento básico de OpenStack

Una vez instalado OpenStack, debemos conocer la dirección IP que tenemos asignada para poder acceder al Dashboard desde otro equipo o podemos acceder desde el mismo equipo con la dirección de loopback (127.0.0.1 o escribiendo localhost). Para acceder desde otro equipo, acceder primero al equipo donde se instaló OpenStack, abrir un terminal y realizar un ifconfig para ver que IP tiene la interfaz principal (eth0 o eth1).

```
$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:60:9e:e8
          Direc. inet:10.0.2.15  Difus.:10.0.2.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe60:9ee8/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:17721 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:6895 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:20681092 (20.6 MB)  TX bytes:469405 (469.4 KB)
```

Conocida la IP a la que hay que acceder, abrimos una ventana en el navegador y la ingresamos. Aparecerá una pantalla de login del Dashboard como esta:

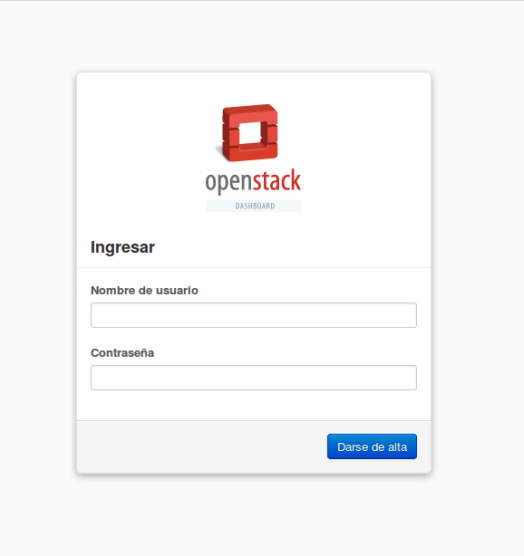


Ilustración 15: Pantalla de login

Introduciremos como usuario *admin* y como contraseña la introducida para el servicio de identidad al inicio de la ejecución de Devstack (en nuestro caso *devstack*). Se mostrará la página principal de administrador del sistema:

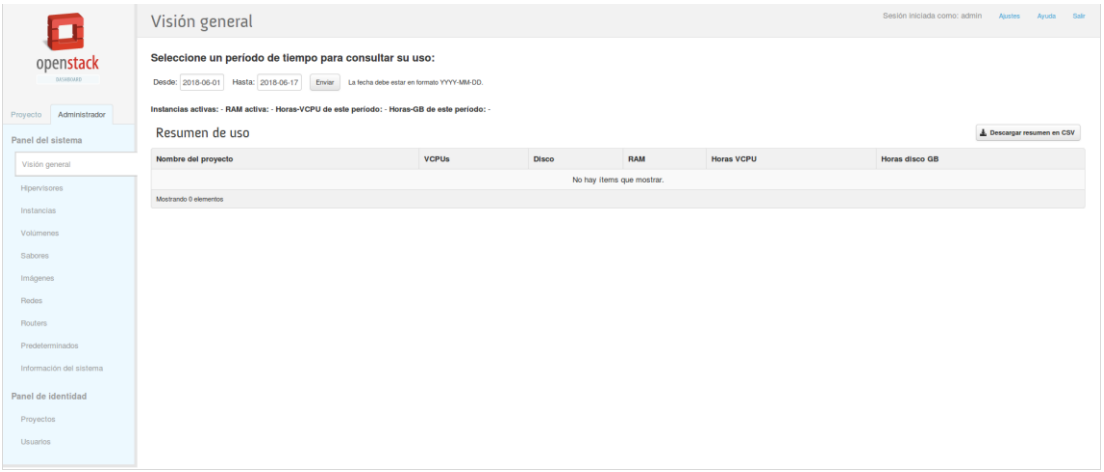


Ilustración 16: Visión general de administrador

En esta pantalla (Visión general) se mostrarán los recursos asignados a cada proyecto. Esos recursos son los utilizados por las instancias para los diferentes objetivos.

En la parte izquierda contamos con una barra de navegación que incluye dos pestañas, Administrador y Proyecto.

La pestaña Administrador nos permite tener un control genérico (dependiendo de los permisos de los que disponga el usuario logado) de todo lo que está pasando.

A su vez consta de dos paneles, el Panel de sistema donde podemos ver los proyectos con instancias en ejecución, detalle de instancias activas, recursos utilizados por las

mismas, hipervisores utilizados y gestionar los tipos de volúmenes para almacenamiento, gestión de imágenes, gestión de redes y routers, gestión de sabores (organizadores de varios proyectos) y también ver la información de todo el sistema.

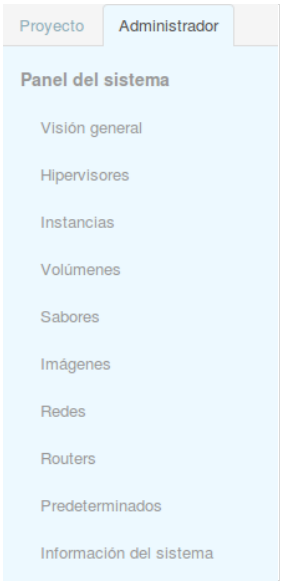


Ilustración 17: Panel del sistema

Dentro del Panel del sistema, tenemos las siguientes opciones:

La opción Hipervisores, permite ver el estado actual del hipervisor del sistema, RAM utilizada, almacenamiento utilizado y número de instancias actualmente activas.

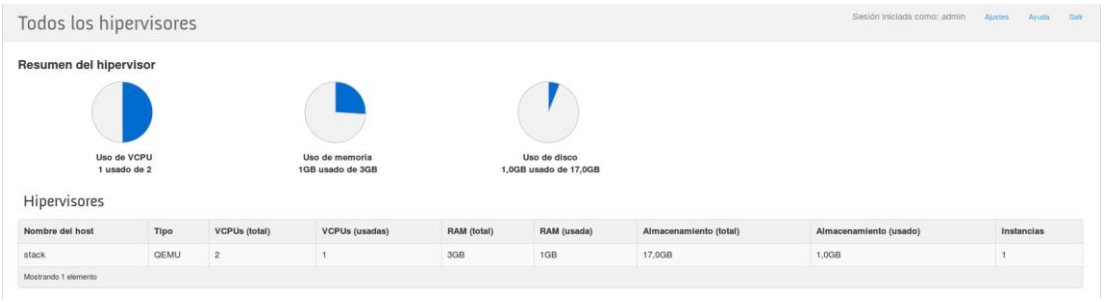
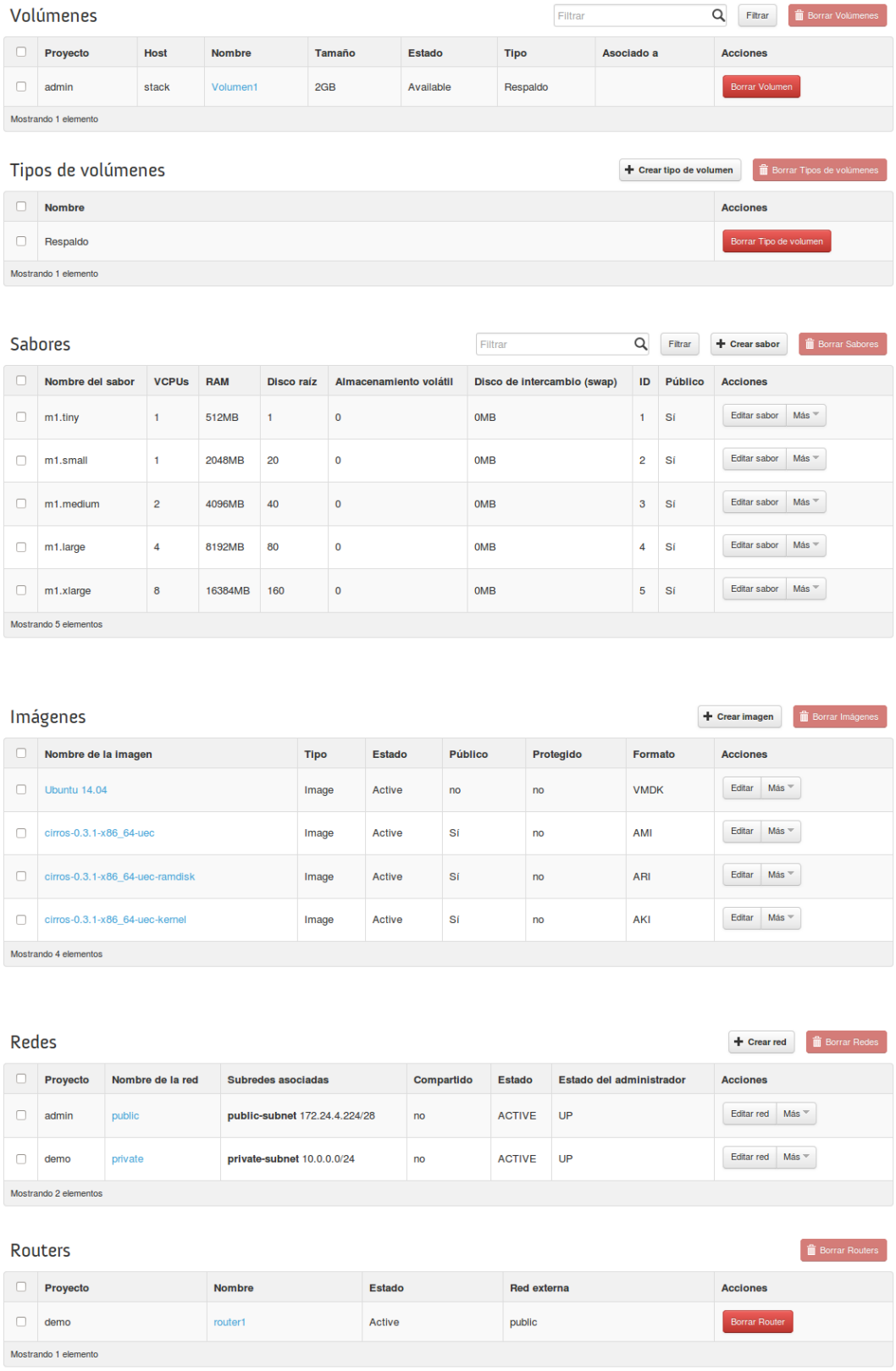


Ilustración 18: Hipervisores del sistema

La gestión de recursos de sistema, independientemente del proyecto, se puede realizar desde las opciones Instancias, Volúmenes, Sabores, Imágenes, Redes y Routers.



Volúmenes

Filtrar | Filtrar | Borrar Volúmenes

	Proyecto	Host	Nombre	Tamaño	Estado	Tipo	Asociado a	Acciones
☐	admin	stack	Volumen1	2GB	Available	Respaldo		Borrar Volumen

Mostrando 1 elemento

Tipos de volúmenes

+ Crear tipo de volumen | Borrar Tipos de volúmenes

	Nombre	Acciones
☐	Respaldo	Borrar Tipo de volumen

Mostrando 1 elemento

Sabores

Filtrar | Filtrar | + Crear sabor | Borrar Sabores

	Nombre del sabor	VCPUs	RAM	Disco raíz	Almacenamiento volátil	Disco de intercambio (swap)	ID	Público	Acciones
☐	m1.tiny	1	512MB	1	0	0MB	1	Sí	Editar sabor | Más
☐	m1.small	1	2048MB	20	0	0MB	2	Sí	Editar sabor | Más
☐	m1.medium	2	4096MB	40	0	0MB	3	Sí	Editar sabor | Más
☐	m1.large	4	8192MB	80	0	0MB	4	Sí	Editar sabor | Más
☐	m1.xlarge	8	16384MB	160	0	0MB	5	Sí	Editar sabor | Más

Mostrando 5 elementos

Imágenes

+ Crear imagen | Borrar Imágenes

	Nombre de la imagen	Tipo	Estado	Público	Protegido	Formato	Acciones
☐	Ubuntu 14.04	Image	Active	no	no	VMDK	Editar | Más
☐	cirros-0.3.1-x86_64-uec	Image	Active	Sí	no	AMI	Editar | Más
☐	cirros-0.3.1-x86_64-uec-ramdisk	Image	Active	Sí	no	ARI	Editar | Más
☐	cirros-0.3.1-x86_64-uec-kernel	Image	Active	Sí	no	AKI	Editar | Más

Mostrando 4 elementos

Redes

+ Crear red | Borrar Redes

	Proyecto	Nombre de la red	Subredes asociadas	Compartido	Estado	Estado del administrador	Acciones
☐	admin	public	**public-subnet** 172.24.4.224/28	no	ACTIVE	UP	Editar red | Más
☐	demo	private	**private-subnet** 10.0.0.0/24	no	ACTIVE	UP	Editar red | Más

Mostrando 2 elementos

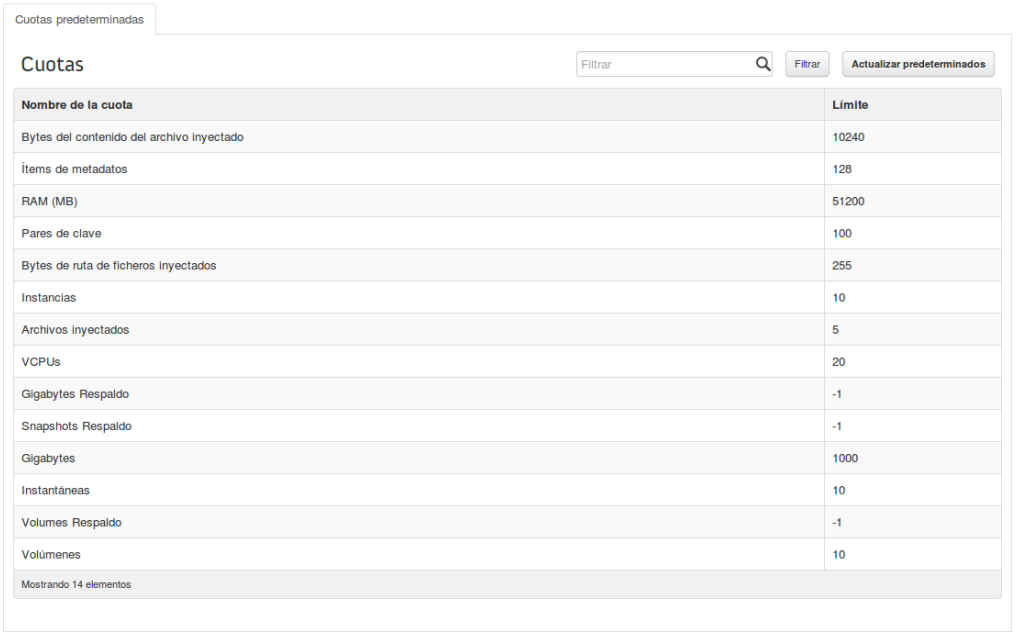
Routers

Borrar Routers

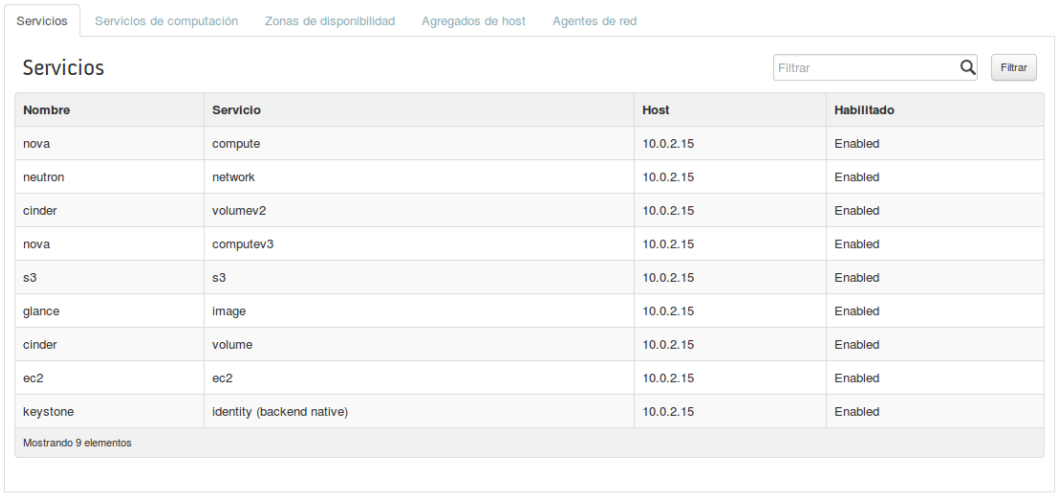
	Proyecto	Nombre	Estado	Red externa	Acciones
☐	demo	router1	Active	public	Borrar Router

Mostrando 1 elemento

Por último podemos visualizar las cuotas (límites de recursos disponibles) en la opción Predeterminados.



Y los servicios, zonas de disponibilidad y agentes de red que componen nuestro sistema en la opción Información del sistema.



Justamente debajo del Panel del sistema tenemos el Panel de identidad desde el cual podemos gestionar nuevos proyectos y gestionar usuarios.

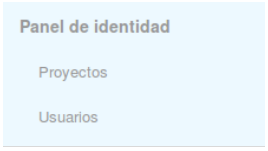
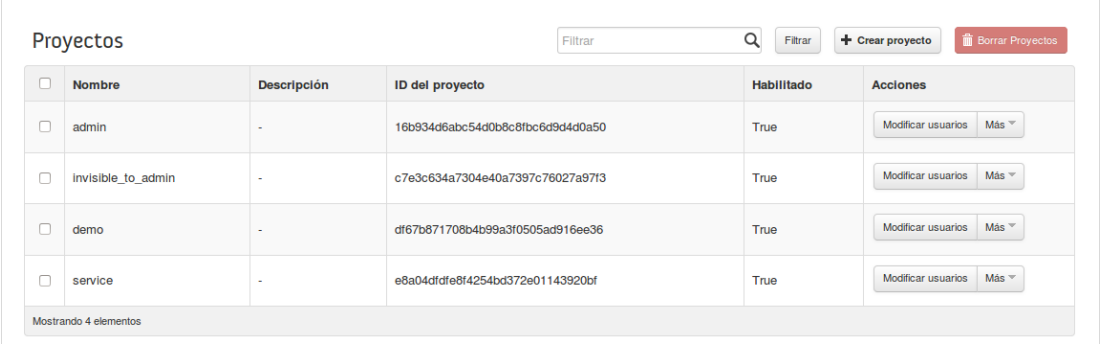
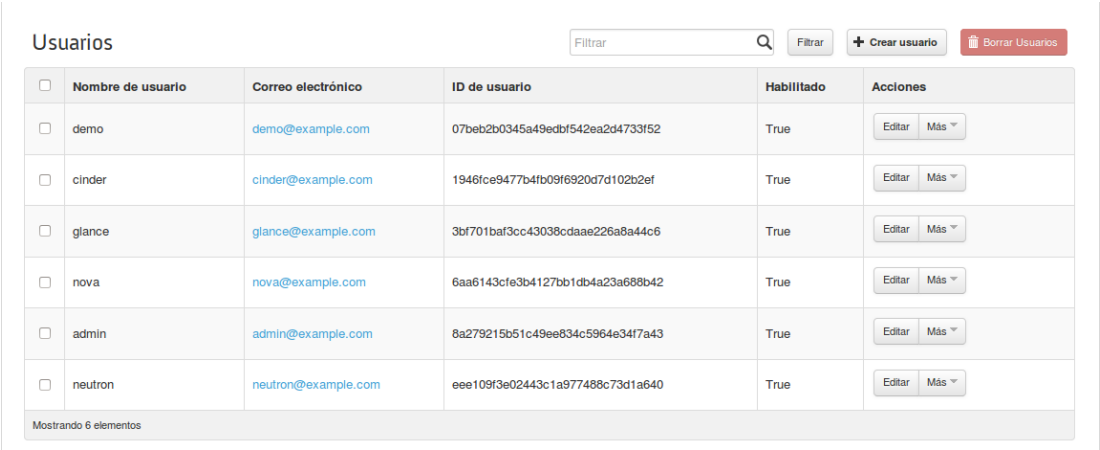


Ilustración 19: Panel de identidad

Dentro del Panel de identidad tenemos la opción Proyectos.



Y la opción Usuarios donde podremos crear y gestionar como Administrador los usuarios necesarios, asignarles roles y un proyecto principal.



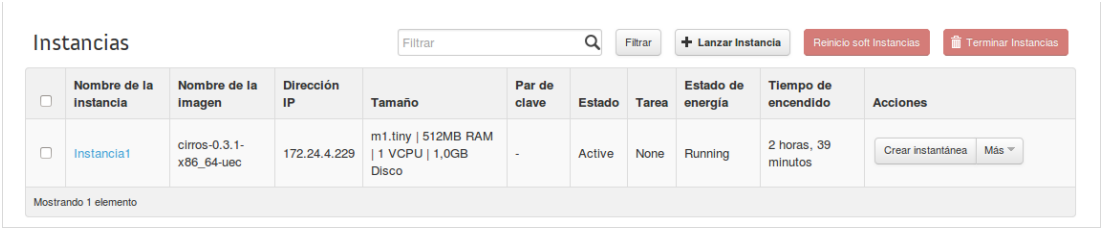
A continuación explicaremos las opciones de la pestaña Proyecto.

Desde la pestaña proyecto podremos escoger el proyecto actual que queramos gestionar. Consta en este caso de dos partes (depende de los módulos que se hayan instalado, Compute, Glance, Cinder, etc…): Administrar compute, donde podremos gestionar todo lo relativo a instancias, imágenes volúmenes, acceso y seguridad, y Administrar red donde podremos crear redes y routers al proyecto y ver la topología de red actual.

En la parte de administración de compute, tenemos la opción visión general desde la que podremos ver de forma gráfica los recursos consumidos por el proyecto actual.

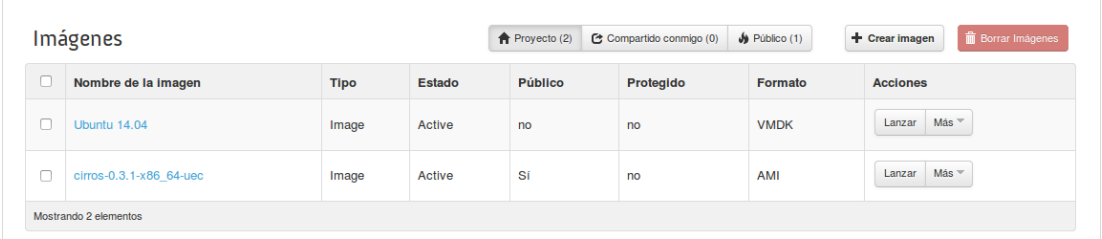
Las limitaciones de recursos con las cuotas máximas serán las que se definieron cuando se creó el proyecto (en la pestaña de Administrador).

Dentro del proyecto en el que estemos podremos gestionar las instancias del proyecto, es decir, lanzarla, pararla, eliminarla, hacer una snapshot, etc.



Lo mismo para para gestionar el almacenamiento, desde volúmenes podremos asociar espacios de almacenamiento a una instancia. Solo puede asociarse un volumen a una única instancia al mismo tiempo, pero podemos desasociarlo de la instancia y asociarlo de nuevo a una instancia diferente.

En el apartado de Imágenes e instantáneas podemos subir las imágenes de los sistemas operativos que queramos lanzar con nuestras instancias. Por defecto se incluye una imagen de prueba del sistema operativo CirrOs, sistema operativo liviano utilizado fundamentalmente para realizar pruebas de virtualización.



Desde aquí podremos además ver y gestionar las snapshots (instantáneas) realizadas a volúmenes.

La última opción de la parte de administración de compute es la de Acceso y seguridad en el que podremos definir grupos de seguridad basados en reglas y acceso a la API.

En la parte de administración de red encontramos tres opciones:

Topología de red, en la que podemos ver de forma gráfica el aspecto que tienen las instancias desplegadas, su estado y a que red o redes pertenecen.

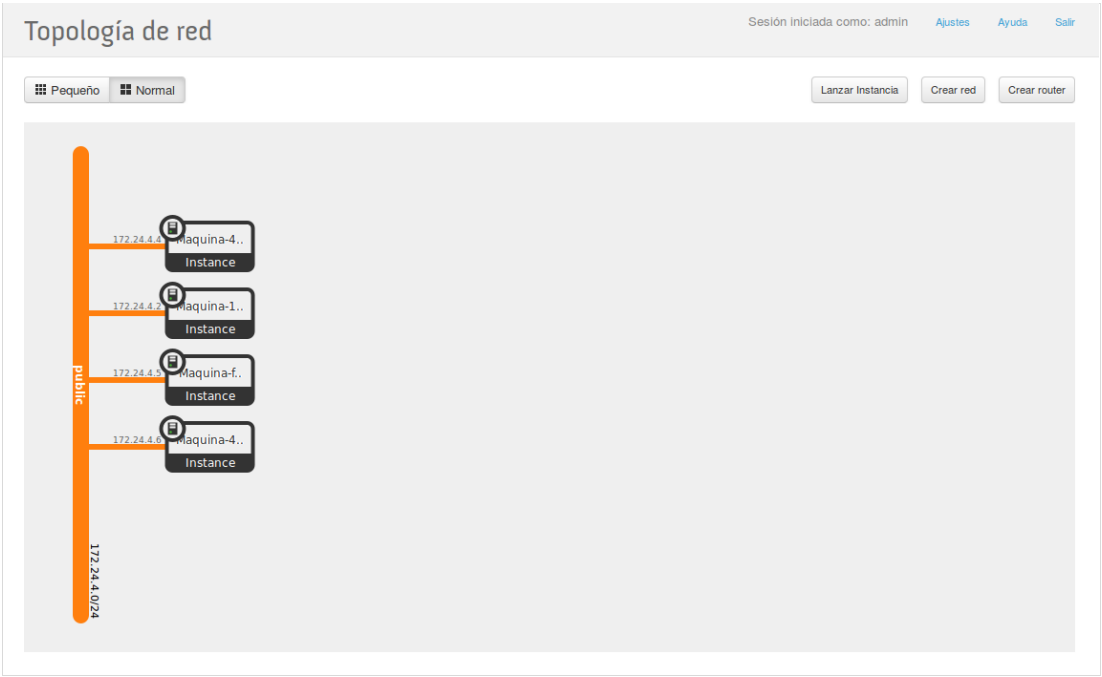


Ilustración 20: Topología de red pública con cuatro instancias

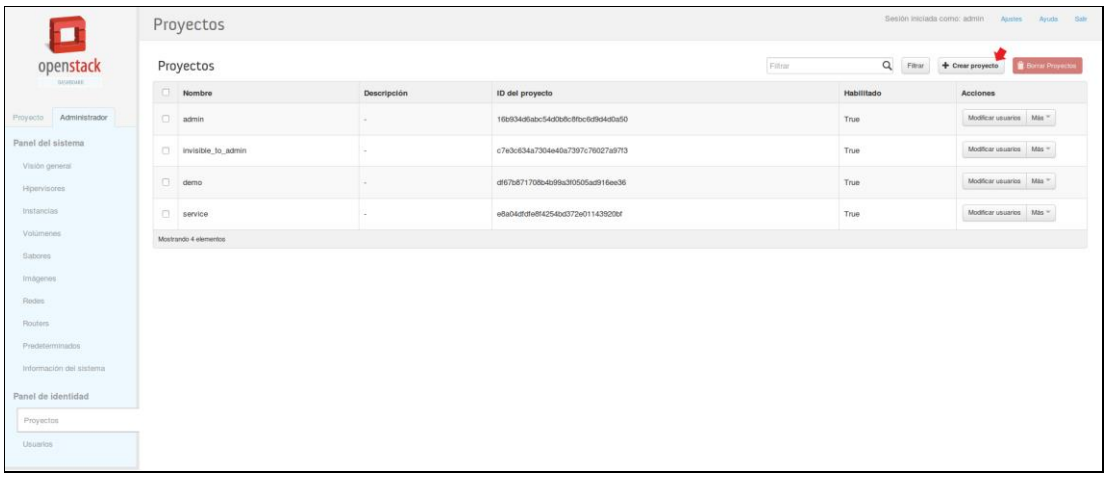
Desde esta pantalla además podremos lanzar nuevas instancias y crear nuevas redes y routers.

7. Ejemplo de creación de un nuevo proyecto

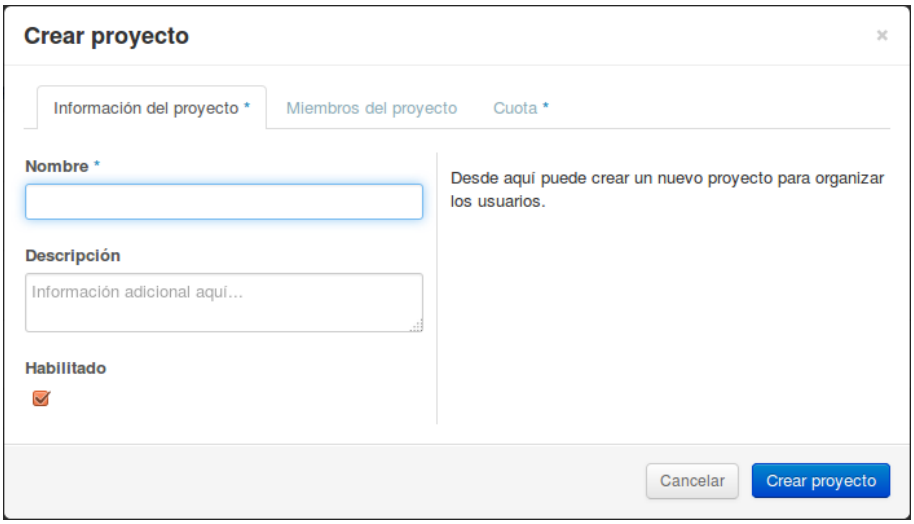
El administrador podrá asignar proyectos a los usuarios existentes o para usuarios que se creen posteriormente. Vamos a crear un proyecto de ejemplo y lo asignaremos a un usuario, y además lanzaremos algunas instancias sobre él.

Debemos iniciar sesión como administrador, y acceder al apartado Proyectos del panel de identidad de la pestaña Administrador.

Pulsamos sobre el botón Crear proyecto en la parte superior de la pantalla.

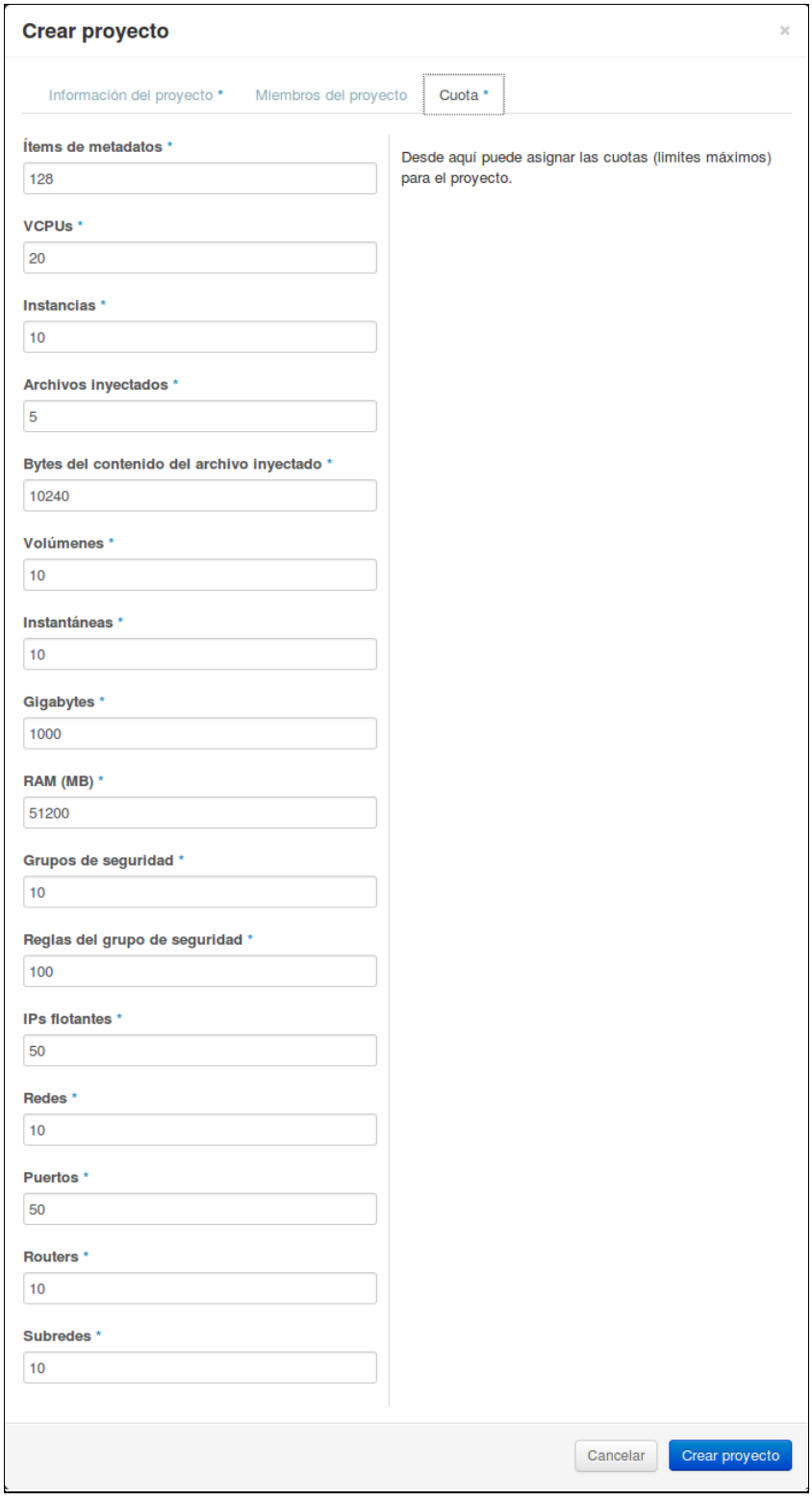


Nos saldrá un popup de tres pestañas, en la primera definiremos el nombre del proyecto y una breve descripción del mismo.



Dado que aún no hemos creado usuarios para asignarlos a este proyecto vamos a saltarnos la pestaña Miembros del Proyecto.

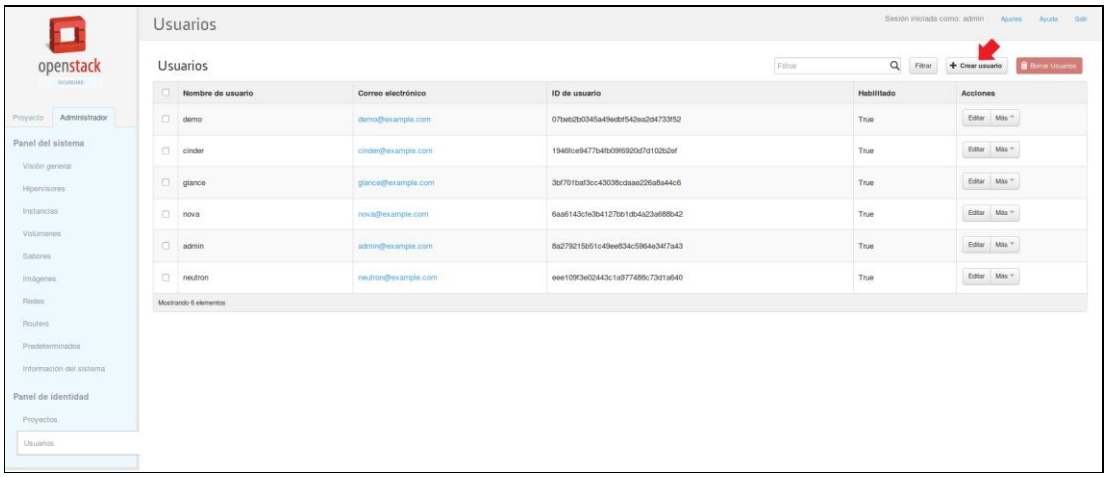
En la tercera pestaña, Cuota, definiremos los límites máximos que tendrá el proyecto, es decir, máximo de instancias, VCPUs, Volúmenes, RAM, grupos, puertos, routers. Una vez definidos los máximos pulsamos en el botón de la parte inferior crear proyecto.



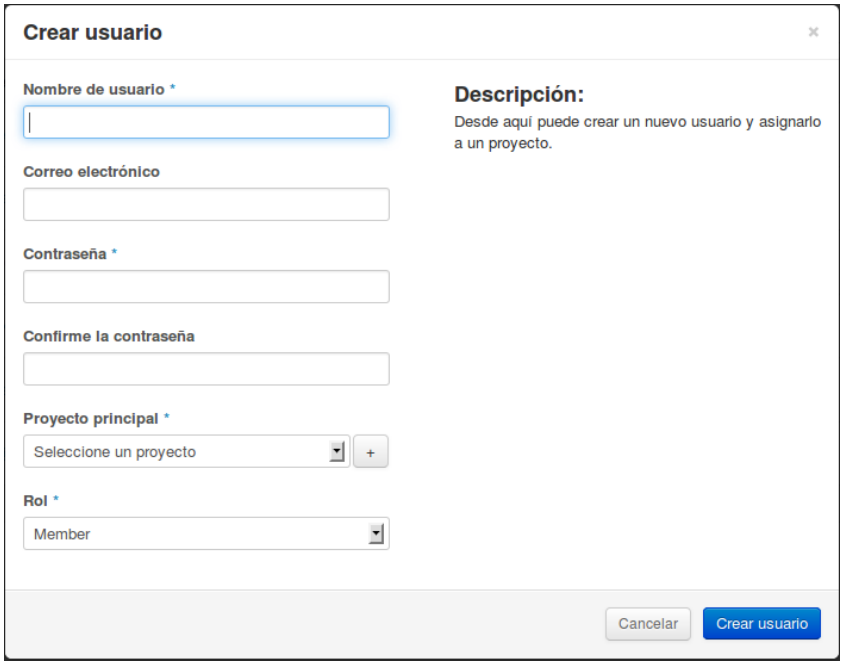
Hecho esto el proyecto saldrá en el listado de Proyectos.

Vamos a crear ahora un usuario para este proyecto, para ello pulsamos en la pestaña Usuarios del Panel de identidad.

El funcionamiento es parecido al de creación de Proyectos, pulsamos sobre el botón Crear usuario.



Nos saldrá un popup donde definiremos el nombre del usuario, correo electrónico, contraseña, el proyecto principal al que está asignado (podemos crearlo directamente aquí también) y el rol. Pulsamos el botón Crear usuario y ya habremos acabado.

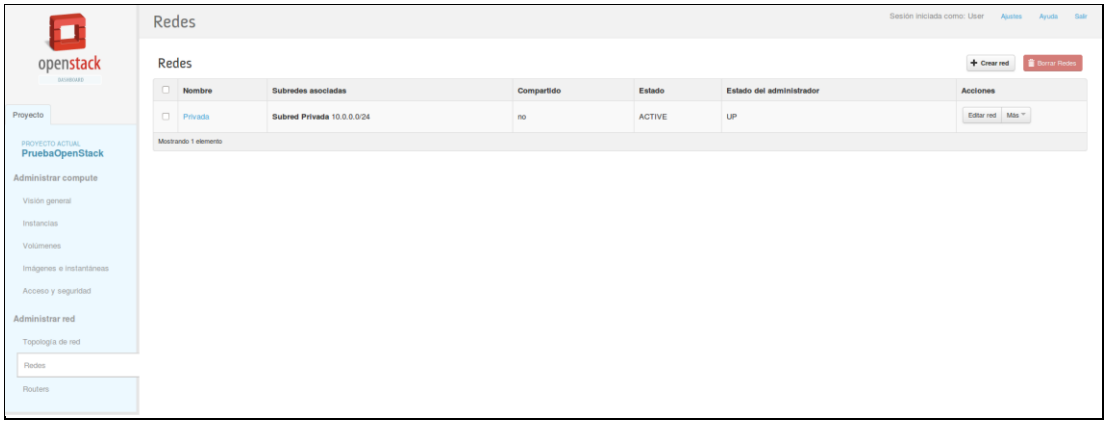


Si nuestro usuario sale en el listado de usuarios es que lo hemos hecho correctamente. Si ahora cerramos la sesión e intentamos iniciarla de nuevo con este nuevo usuario deberíamos tener acceso a las operaciones de Openstack y al proyecto asignado. En nuestro caso hemos creado al usuario User, con rol Member y asignado al proyecto PruebaOpenStack.

Como vemos, al ser un usuario de rol Member no tiene pestaña de Administrador y solo tiene acceso al proyecto al que ha sido asignado.

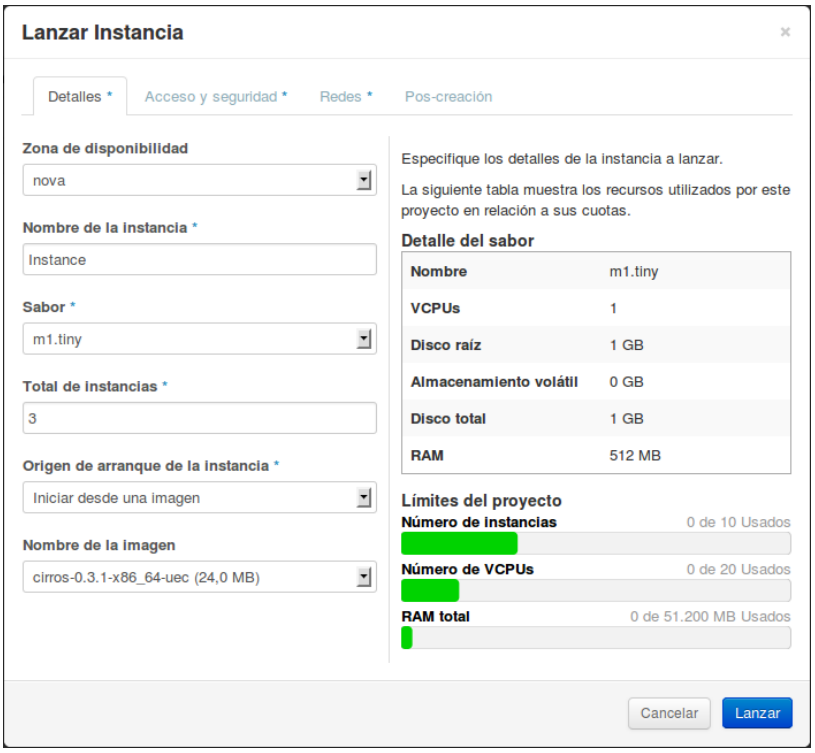
Vamos a montar para nuestro nuevo proyecto una red privada y lanzaremos tres instancias en dicha red. Para ello accedemos al apartado Redes en el menú de la izquierda, y pulsamos en el botón Crear Red de la derecha:

Aparecerá un popup con tres pestañas, en la primera pondremos Privada como nombre para la Red, en la segunda pestaña introducimos los parámetros de subred, Subred Privada como nombre de la Subred, las direcciones de red en formato CIDR (Ej. 192.168.0.0/24), escogemos IPV4 como versión ip y 10.0.0.1 como IP de la puerta de enlace. La tercera pestaña podremos configurar DHCP, DNS y pools pero en este caso dejaremos todo con la configuración por defecto. Tendríamos algo como esto:

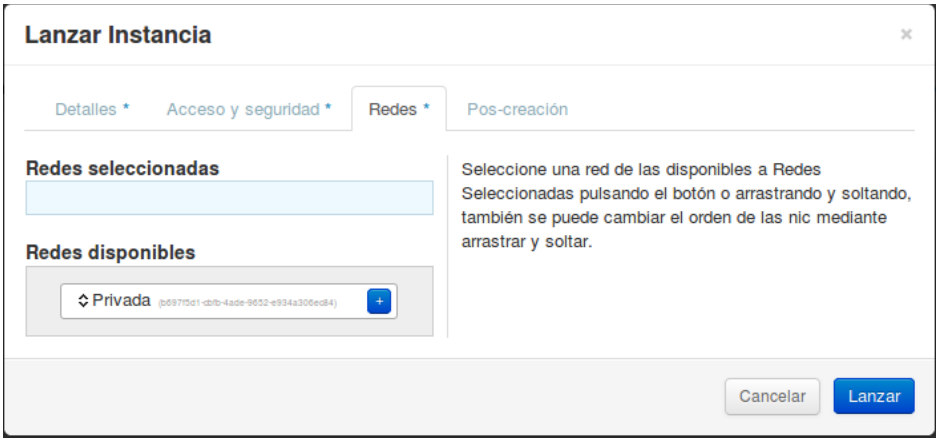


Vamos a continuar con la creación de las instancias para la red que acabamos de crear. Accedemos al apartado Instancias de la parte de Compute y pulsamos sobre el botón Lanzar Instancias.

Introducimos los siguientes campos para crear las tres instancias a la vez puesto que inicialmente van a ser iguales:

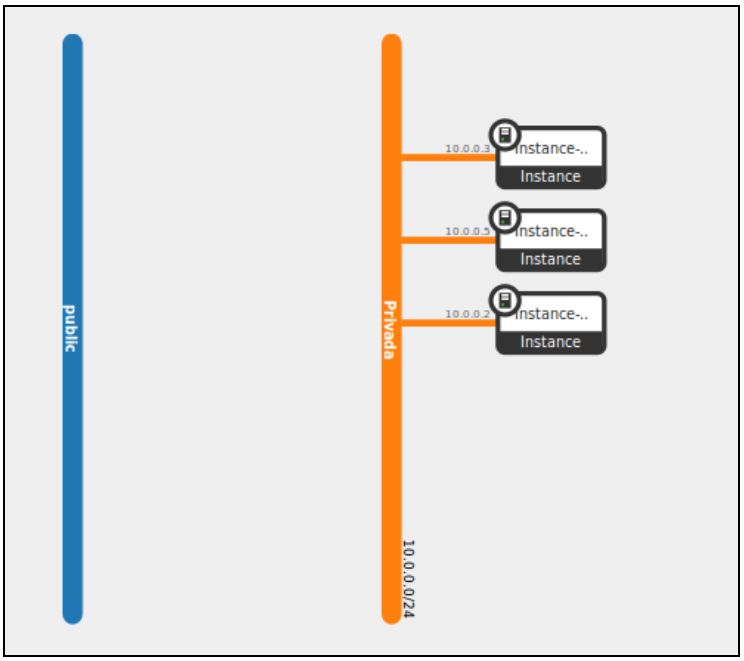


Debemos cambiar el valor Total de Instancias a tres para que se creen las tres instancias y marcar como origen Iniciar desde una imagen. Esta imagen viene cargada por defecto con un sistema operativo liviano basado en comandos, pero podría ser una imagen de cualquier otro sistema (Windows, Unix, etc) Accedemos al apartado Redes para asignar las instancias a la red privada que creamos anteriormente. Como solo existe la red privada que creamos solo aparecerá ésta en la parte de Redes disponibles para añadir.

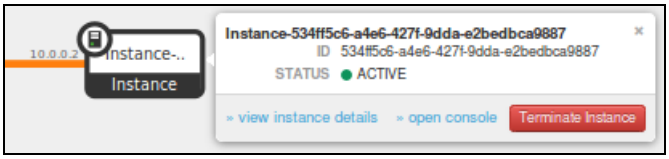


Pulsamos sobre el botón Lanzar y directamente volveremos al apartado de Instancias donde podremos ver cómo va cambiando el estado de cada instancia hasta que alcance los estados Active y Running.

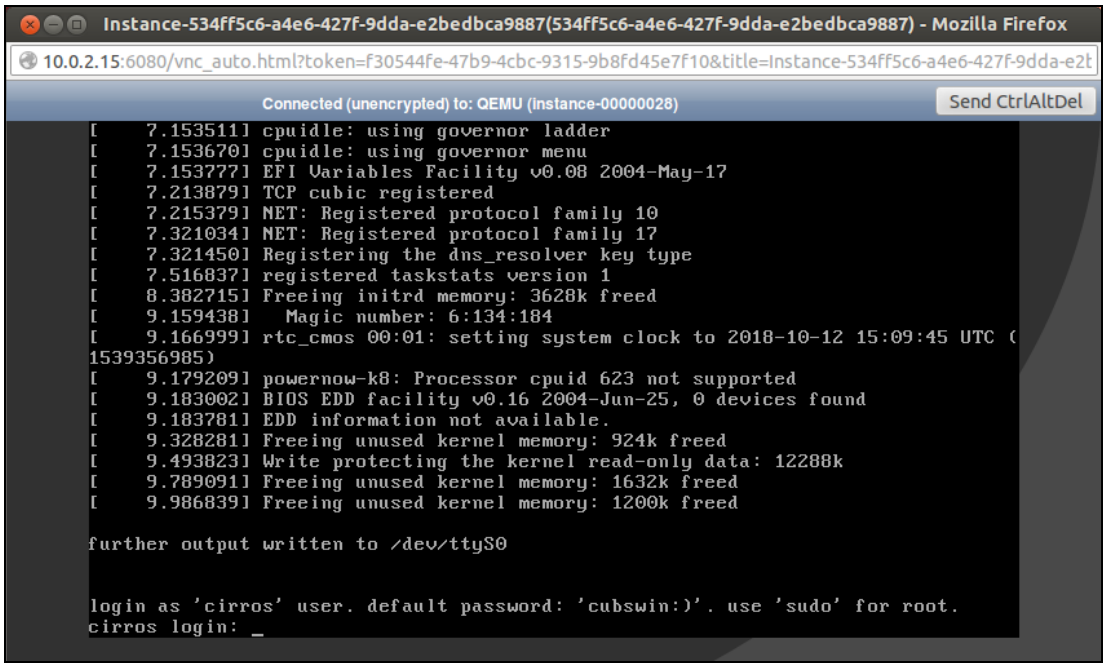
Si accedemos al apartado topología de red tendremos algo como esto:



Dado que están en una red local las instancias se pueden ver entre ellas, podremos comprobarlo haciendo un ping a otra de las máquinas. Abrimos una consola (en nuestro caso la de la máquina 10.0.0.2) señalando una de las instancias en esta misma pantalla y pulsamos sobre el enlace open console.



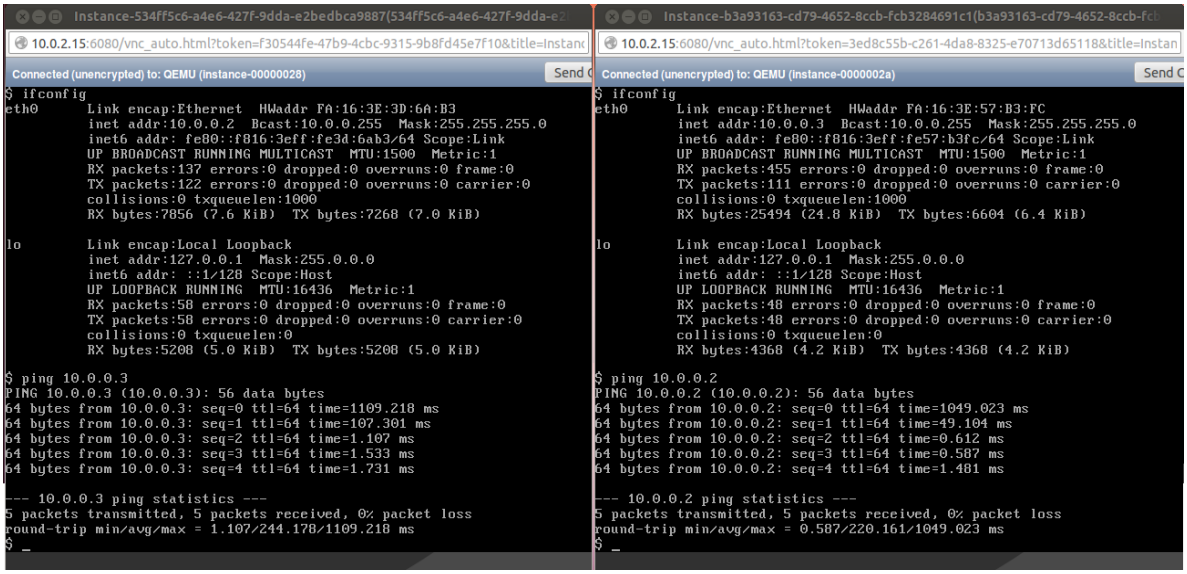
Se abrirá un escritorio remoto a la máquina (VNC).



Esta imagen de cirros ya viene con un usuario y password predefinidos. En este caso ingresaremos como usuario ‘cirros’ y contraseña ‘cubswin:)’.

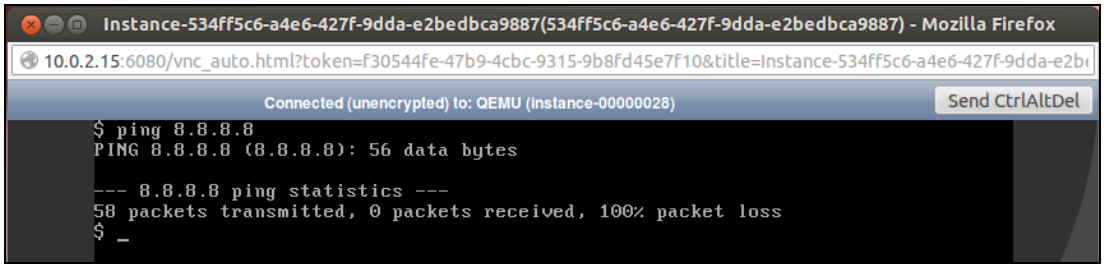
Si ingresamos ‘ifconfig’ veremos la configuración de red de la máquina, que será exactamente la que configuramos con OpenStack.

Vamos a verificar si dos de las máquinas de la red (10.0.0.2 y 10.0.0.3) que creamos tienen visibilidad utilizando el comando ping en ambas.



Observamos que las dos máquinas son visibles entre ellas.

Imaginemos que queremos que las máquinas de la red privada que creamos, tengan acceso a Internet. Actualmente podemos comprobar en las máquinas si tenemos acceso a Internet haciendo un ping a la dirección 8.8.8.8 (Google Public DNS IPv4). Los paquetes se pierden en el ping realizado.

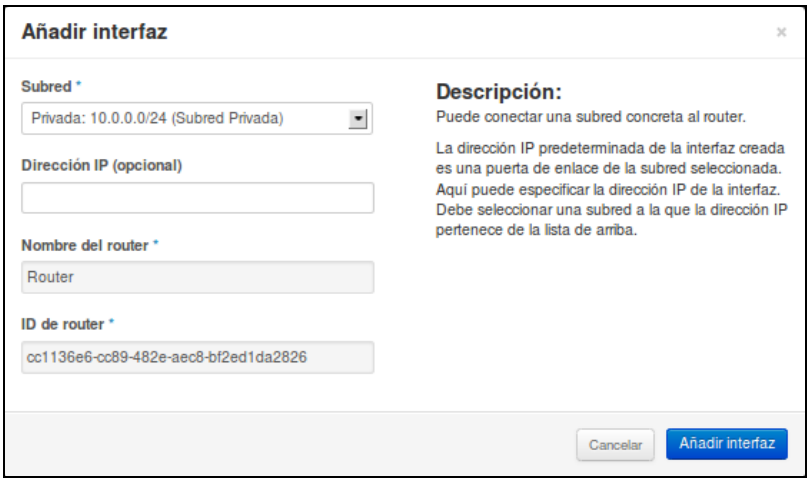


Para hacer que los equipos de la red privada tengan acceso a Internet debemos configurar una puerta de enlace al exterior. Para ello debemos acceder al apartado Routers y pulsar el botón Crear Router o si nos encontramos en el apartado Topología de Red simplemente pulsar el botón Crear Router en la parte superior derecha. En este paso solo tendremos que dar un nombre a nuestro Router y aparecerá como un elemento que configuraremos a continuación.

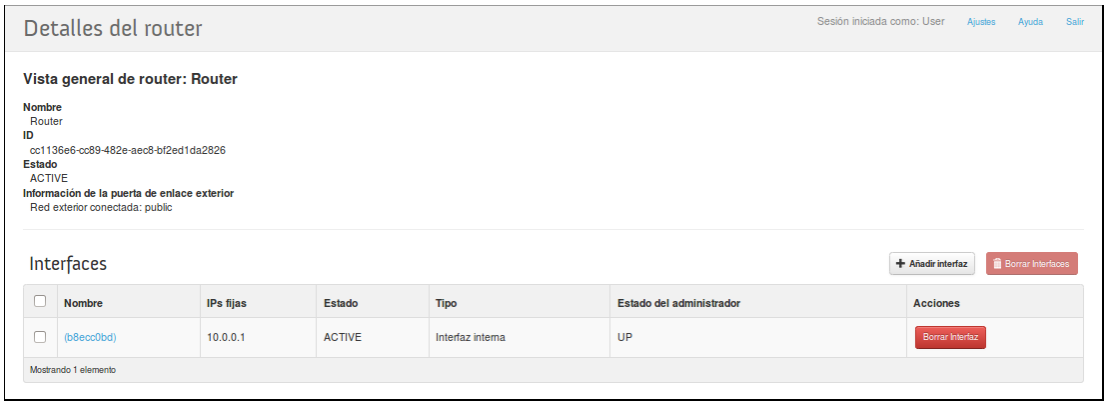
Antes de configurar las interfaces de nuestro Router, accedemos al apartado Routers y pulsaremos sobre el botón Establecer puerta de enlace. Se abrirá un popup en el que indicaremos la Red Externa public. Aceptaremos los cambios pulsando en el botón azul.

Esta acción hará que nuestro router configure una interfaz con la dirección IP (puerta de enlace) de la red public (red por defecto que representa Internet en nuestra nube).

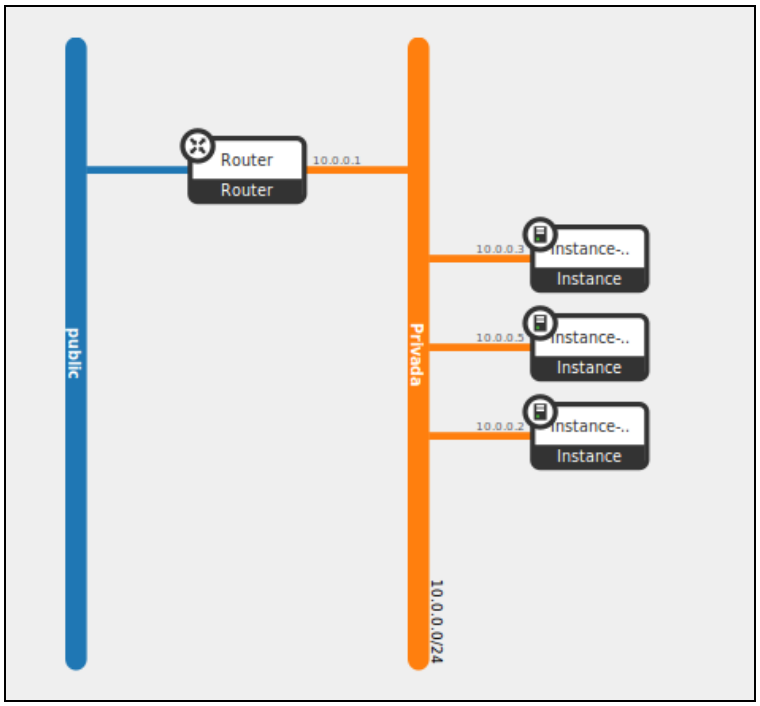
Ahora configuraremos la interfaz a nuestra red privada. Para ello pulsamos sobre el nombre de nuestro Router. Pulsamos sobre el botón Añadir interfaz, se abrirá un popup donde indicaremos únicamente la subred que creamos de la red privada (la dirección IP es opcional), quedando de la siguiente forma:



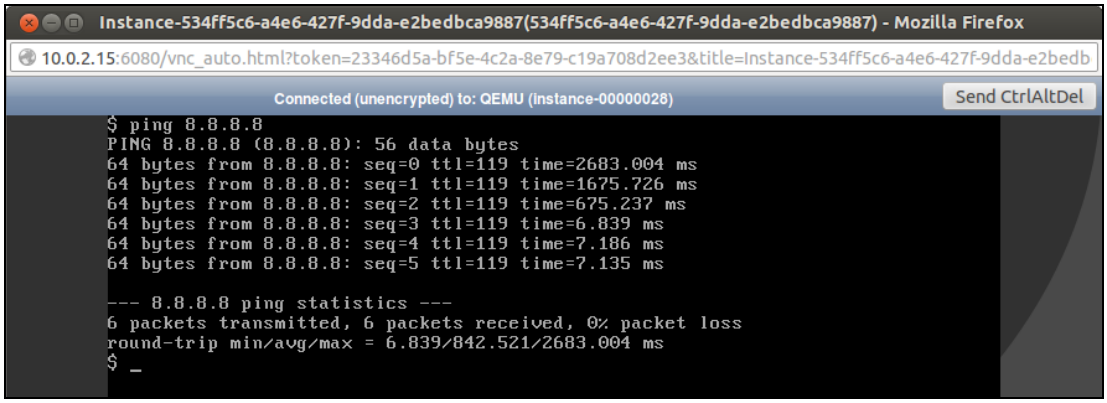
Al pulsar el botón Añadir Interfaz, esta se creará quedando como IP fija la de la subred privada:



Podremos ver los resultados de esta configuración en el apartado Topología de Red:



Comprobaremos de nuevo si las maquinas tienen acceso a Internet haciendo un ping a la dirección 8.8.8.8 (también valdría otra de Internet). Ahora sí que se envían y reciben paquetes.



8. Adaptación y despliegue del proyecto SmartPolitech en OpenStack para virtualización en la nube

Actualmente y desde hace ya algunos años, se han desarrollado multitud de ideas para convertir la Escuela Politécnica de la UEx en un ecosistema experimental tecnológico basado en tecnologías SmartX (living-lab), creando espacios inteligentes, utilizando energías eficientes y facilitando la vida social y académica. Tanto es así que desde entonces, estudiantes que pasaban por estos espacios se quedaban maravillados al pasar y ver en pasillos, aulas y bibliotecas, pequeños dispositivos como cámaras y sensores que automatizaban el paso de personas del lugar, medían el consumo de agua y gas, pantallas que mostraban resultados de la calidad del aire, temperatura, entre otros novedosos sistemas que han hecho de la Escuela Politécnica un referente nacional dando opción a las empresas de conocer *in situ* el funcionamiento de estas tecnologías y colaborando directamente en el desarrollo de soluciones para estas. (Universidad de Extremadura, 2018).

Alrededor de más de 20 grupos de investigación de Ingeniería Civil, Edificación, Informática y Telecomunicaciónes, perfiles que brinda la Escuela Politécnica, colaboran en el desarrollo de un proyecto multidisciplinar que comprende cuatro fases. Gracias a una inversión de 140.000 euros se ha logrado comenzar con la primera fase, la sensorización de las zonas más importantes de la escuela.

Salvar y jerarquizar la información recogida de los sensores en bases de datos para que esté siempre actualizada y disponible se correspondería con la segunda fase.

La tercera fase consiste en la monitorización en tiempo real de los sistemas de control de consumo energético, detección de anomalías y seguridad del centro vía Web.

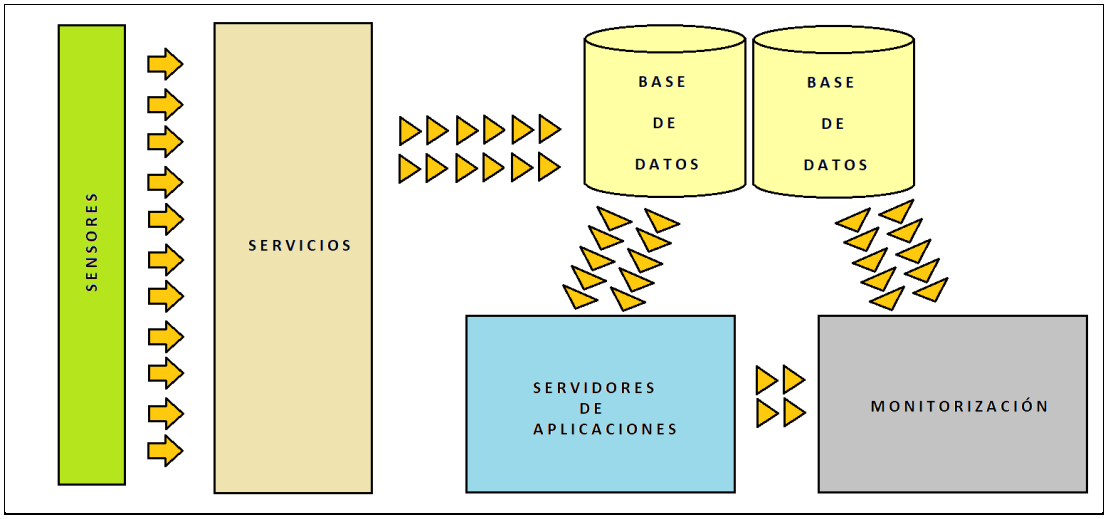
Por último, la fase cuatro se centra en la capacidad de realización de diagnósticos, cálculos y predicciones de cara al futuro de las necesidades energéticas del edificio en tiempo real y se adecue un sistema autónomo e inteligente basado en BigData. (La Vanguardia, 2017).

No cabe duda de que el objetivo es el ahorro energético de toda la Escuela Politécnica, pues solo en energía gasta más del 80% del presupuesto.

Todo el seguimiento de monitorización, sensores, etc, requiere de un número de servidores de bases de datos, Web, correo electrónico localizados en algunos puntos de la Escuela (CPDs) que proporcionen las capacidades descritas y que no requiera un consumo de energía excesivamente alto.

Dado que los proyectos de SmartPoliTech tienen una tendencia transversal, convendría migrar su infraestructura a otra de forma más horizontal. Es decir, en el momento de necesitar mayor capacidad de cómputo o almacenamiento, podríamos simplemente conectar el hardware y añadirlo con un clic al resto sin que esto interfiriese en redireccionar webs, configurar discos duros o configurar nuevas redes. Aquí es donde OpenStack entra en juego, proporcionando recursos a través de Internet.

A continuación se describe un ejemplo de una posible infraestructura de nube en OpenStack para SmartPoliTech:



Contamos con un montón de sensores cuyo cometido es obtener, cada cierto tiempo, múltiples datos que se almacenen directamente en base de datos o quizás necesiten almacenarse de cierta forma realizando cálculos previos, manteniendo cierta estructura o ignorando esta. Es decir que nuestra nube necesitará de un lugar donde toda la parte data SQL y NoSQL se almacene.

Los grupos de investigación tendrán a su disposición los servidores necesarios para correr sus aplicaciones, que adaptarán, calcularán y expondrán los resultados obtenidos vía web.

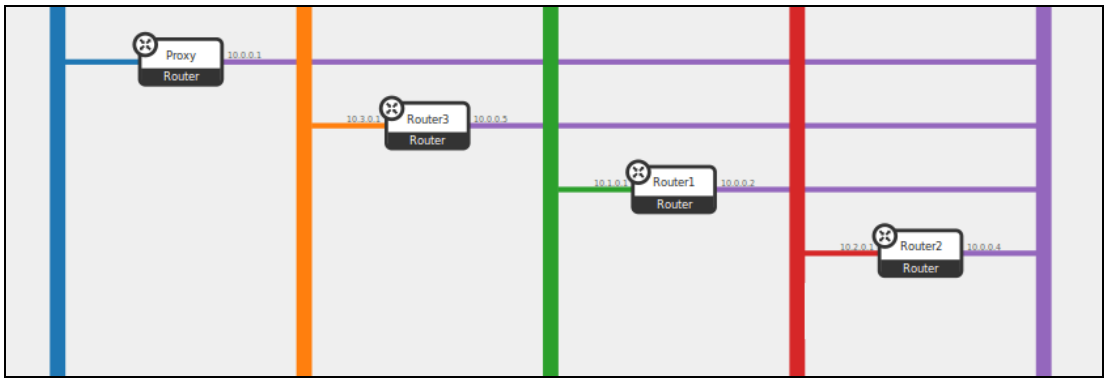
Estos resultados finalmente son los que interesan realmente, los que se mostrarán en los monitores de la escuela de manera informativa, o en los laboratorios de los grupos que estén encargados de tal tarea.

Comenzaremos por agrupar en redes los recursos que son necesarios:

- Red de Base de Datos
- Red de Aplicaciones
- Red de Monitorización

Estas redes estarán internamente conectadas para facilitar a los grupos el acceso a cada capa. De hecho podríamos hacer que todas estuvieran conectadas a una Red General. Crearíamos dichas redes con OpenStack.

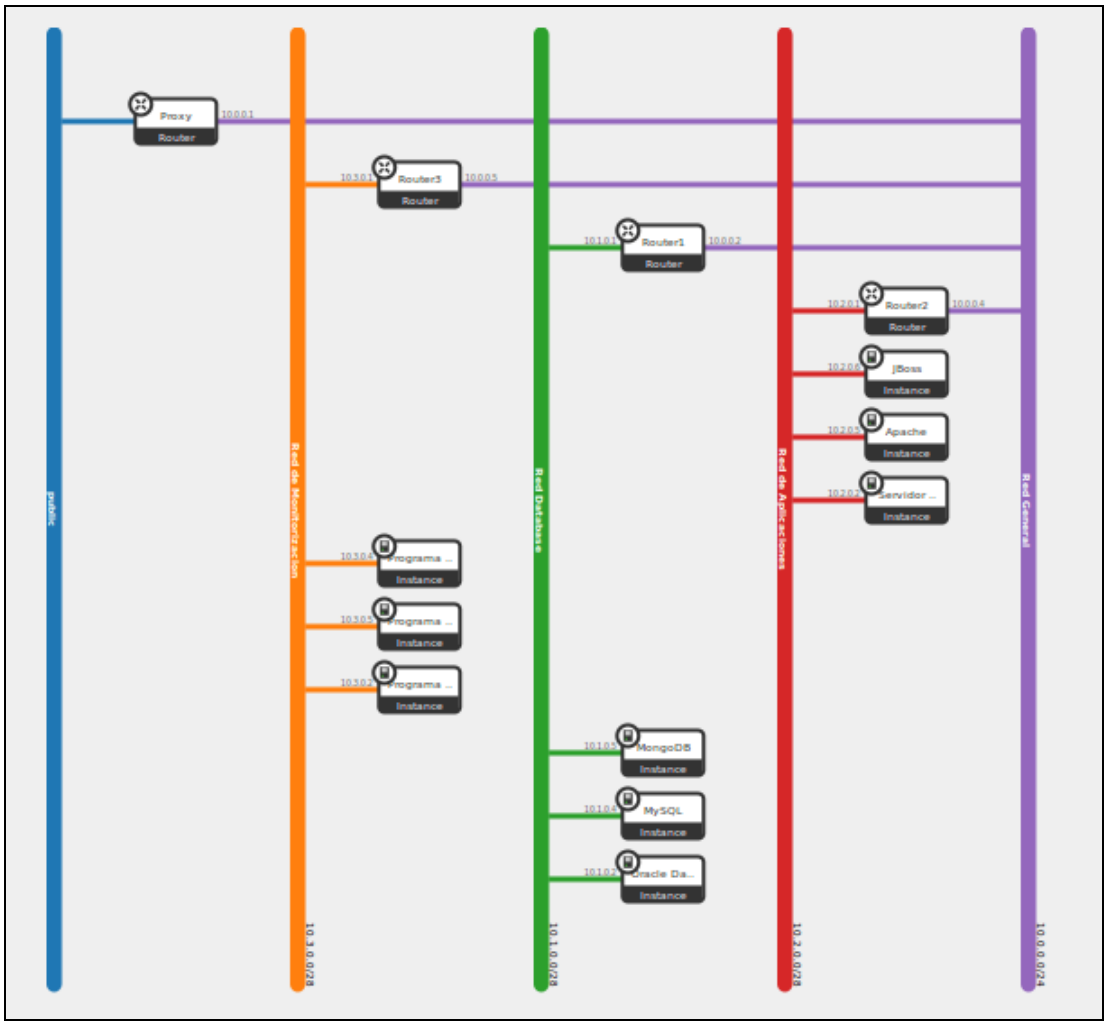
Tendríamos además un Router para cada capa encargado de gestionar posibles subredes locales a cada capa y un proxy para el acceso desde Internet a la Red General desde el que se podría controlar la seguridad.



Prácticamente la infraestructura ya estaría creada, pues desde este punto ya podrían lanzarse instancias de máquinas para roles distintos, servidores de aplicaciones, servidores de correo, bases de datos.

En el caso de que se precisara de mayor capacidad de disco, procesamiento u añadir alguna red (capa), sólo sería añadirla en OpenStack a la instancia/volumen involucrado.

El resultado final sería algo como lo siguiente:



La migración del proyecto SmartPolitech a la nube con OpenStack facilitaría la última fase "*Inteligencia del Edificio*", las tecnologías BigData (Hadoop) para el almacenamiento masivo de datos (NoSQL), podrían desplegarse sin problema para realizar cálculos rápidos (Hadoop), proporcionar indexación (Lucene) para un rápido acceso a los datos y despliegue de servicios mediante tecnologías de contenedores (Docker) también tendrían un mínimo impacto energético. Todo ello para todos los grupos que tendrían todas sus aplicaciones y recursos en la nube, con lo que no precisarían de otros servidores de apoyo en sus edificios.

9. Glosario de términos

Centro de Procesamiento de datos (CPD): lugar donde se encuentran los recursos para el procesamiento de la información de una organización concreta.

Máquina Virtual (MV): contenedor con características virtuales (CPU, RAM, Disco Duro, Interfaces de Red) que puede comportarse y ejecutar software (Sistemas Operativos, Aplicaciones) como si fuera un ordenador físico.

Anfitrión (Host): es el sistema operativo que controla el hardware real y ejecuta el software de virtualización.

Huésped (Guest): es el sistema operativo virtualizado. Son aislados unos con otros y con el propio anfitrión.
Resource Pool: agrupamiento lógico de servidores que permite su administración conjunta.

KVM (Kernel-based Virtual Machine): es una solución de software libre para virtualización completa con Linux (es un hipervisor de tipo I). Permite ejecutar MVs con sistemas operativos sin modificar utilizando imágenes en disco. Cada máquina virtual tiene su propio hardware virtualizado.

LXC (Linux Containers): es una tecnología de virtualización en el nivel de sistema operativo (SO) para Linux. LXC permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales (SPV o VPS en inglés) o Entornos Virtuales (EV). LXC no provee de una máquina virtual, más bien provee un entorno virtual que tiene su propio espacio de procesos y redes.

OpenVZ: es una tecnología de virtualización en el nivel de sistema operativo para Linux. OpenVZ permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales (SPV o VPS en inglés) o Entornos Virtuales (EV).

VMware Inc.: es una filial de EMC Corporation (propiedad a su vez de Dell Inc) que proporciona software de virtualización disponible para ordenadores compatibles X86.

Virtual Center Server: componente que permite gestionar múltiples servidores VMware y máquinas virtuales.

Devstack: conjunto de shell scripts que nos permiten instalar OpenStack de forma automática ya sea en una máquina virtual o en una máquina física. Está orientado a entornos de prueba con los servicios OpenStack.

Member: tipo de rol que se asigna a un usuario de la infraestructura para que sólo pueda realizar determinadas operaciones en los proyectos en los que está asignado.

Admin: tipo de rol que se asigna a un usuario de la infraestructura para que pueda realizar operaciones especiales (se muestra la pestaña Administrador) en los proyectos en los que está asignado.

Volumen: hace referencia al contenido o almacenamiento de uno o varios discos en un determinado tipo de formato de sistema de archivos.

Snapshot: instantánea de un volumen con los mismos recursos y opciones que tenía este en el momento que se hizo. Es utilizado en restauración y recuperación rápida de los datos de un sistema.

Instancia: objeto de la imagen lanzada por el administrador que contiene una imagen de sistema operativo o un volumen.

10. Referencias

Cita: (Molina Coballes, 2009)

Molina Coballes, A. (2009). Introducción a la virtualización. *Desde lo alto del cerro*. Consultado 22 Julio 2016

https://albertomolina.wordpress.com/docs/

https://albertomolina.files.wordpress.com/2009/10/virtualizacion.pdf

Cita: Traducción de un artículo publicado en IBM Developerworks, escrito por M. Tim Jones. (Tim Jones, 2007)

Tim Jones, M. (2007). *IBM Developerworks*. *Institut Puig Castellar*. Consultado 22 Julio 2016, http://elpuig.xeill.net/Members/vcarceler/articulos/virtual-linux/index_html

Cita: (Lugo Cardozo, 2014)

Lugo Cardozo, N. (2014). TECNOLOGÍAS DE VIRTUALIZACIÓN EN LOS SISTEMAS INFORMÁTICOS DE LAS ORGANIZACIONES EMPRESARIALES DEL ESTADO ZULIA. *TELEMATIQUE*, (2). Consultado: 22 Julio 2016

http://publicaciones.urbe.edu/index.php/telematique/article/view/3224/html

Cita: (Grupo IES, 2011)

IES Gonzálo Nazareno, IES Los Albares, IES La Campiña, IES Ingeniero de la Cierva (2011). *Virtualización de servidores*. Sevilla-Murcia: IES Gonzálo Nazareno, IES Los Albares, IES La Campiña, IES Ingeniero de la Cierva. Consultado 22 Julio 2016

Apartado Introducción a la Virtualización:

http://www.gonzalonazareno.org/cloud/material.html

http://www.gonzalonazareno.org/cloud/material/IntroVirtualizacion.pdf

Cita: ("Virtualización", 2016)

Virtualización. (2016). *Es.wikipedia.org*. Consultado 22 Julio 2016

https://es.wikipedia.org/wiki/Virtualización

Cita: (Ares Martín, 2013)

Ares Martín, J. (2013). *Virtualización y cloud computing en la PYME* (1st ed.). Barcelona: Universitat Oberta de Catalunya. Consultado: 22 Julio 2016

Disponible en: http://www.recercat.cat/handle/2072/206333

http://hdl.handle.net/10609/19276

Cita: (Lissette Girón, 2012)
Lissette Girón, A. (2012). Virtualización de servidores. *Calameo*. Consultado: 22 Julio 2016
Disponible en: http://es.calameo.com
http://es.calameo.com/books/001376059171c6615a2ec

Cita: (González Cifuentes, 2014)
González Cifuentes, M. (2014). *Despliegue de una infraestructura Cloud basada en XenServer para Bilib*. Ruidera, Albacete.
Consultado 22 Julio 2016
Disponible en: https://ruidera.uclm.es
http://hdl.handle.net/10578/4253

Cita: (M. Peña, 2015)
M. Peña, J. (2015). *Virtualización*. Universidad Politécnica de Madrid, Madrid.
Consultado: 22 Julio 2016
http://laurel.datsi.fi.upm.es
http://laurel.datsi.fi.upm.es/_media/docencia/asignaturas/asi/virtualizacion.pdf

Cita: (Smaldone, 2008)
Smaldone, J. (2008). Virtualización de hardware. *Blog de Javier Smaldone*. Consultado 22 Julio 2016 https://blog.smaldone.com.ar/2008/09/20/virtualizacion-de-hardware/

Cita: (Miniacademia, 2014)
Formación y manuales sobre virtualización, sistemas operativos y aplicaciones útiles., M. (2014). Introducción a Citrix XenServer. *Miniacademia*. Consultado: 26 Julio 2016.
http://www.miniacademia.es/introduccion-a-citrix-xenserver/

Cita: ("XenServer - Información técnica", 2016)
XenServer - Información técnica. (2016). *Citrix.com*.
Consultado 26 Julio 2016, https://www.citrix.es/products/xenserver/tech-info.html

Cita: ("Proxmox Virtual Environment", 2004)
Proxmox Virtual Environment. (2004). *PROXMOX*. Consultado 28 Julio 2016
https://www.proxmox.com/en/proxmox-ve
https://hipertextual.com/archivo/2010/10/virtualizando-el-centro-de-datos-en-software-libre/

Cita: ("VMWare: ESXi gratuito limitaciones", 2016)
VMWare: ESXi gratuito limitaciones. (2016). *SYSADMIT*. Consultado 24 Agosto 2016 http://www.sysadmit.com/2016/03/vmware-esxi-gratuito-limitaciones.html

Cita: ("VMware ESX", 2016)
VMware ESX. (2016). *Es.wikipedia.org*. Consultado 24 Agosto 2016
https://es.wikipedia.org/wiki/VMware_ESX

Cita: (Guijarro Verdura, 2014)
Guijarro Verdura, L. (2014). *DESPLIEGUE DE UNA NUBE DE COMPUTACIÓN PRIVADA OPENSTACK EN UN ENTORNO ACADÉMICO* (Doctorado). Universidad Carlos III de Madrid. Consultado 25 Agosto 2016. Disponible en: http://hdl.handle.net/10016/22438

Cita: (Pérez, 2014)
Pérez, A. (2014). 'Cloud computing' y virtualización, ¿cuál es la diferencia?. *TICBEAT*. Consultado 30 Agosto 2016.
http://www.ticbeat.com/cloud/cloud-computing-virtualizacion-cual-es-la-diferencia/

Cita: ("La diferencia entre virtualización y computación en la nube", 2011)
La diferencia entre virtualización y computación en la nube. (2011). *ERP SoftwareBlog*. Consultado: 30 Agosto 2016
http://www.erpsoftwareblog.com/2011/09/la-diferencia-entre-virtualizacion-y-computacion-en-la-nube/

Cita: (Lastras Hernansanz, J., Lázaro Requejo, J. Mirón García J. D., 2009)
Lastras Hernansanz, J., Lázaro Requejo, J. Mirón García J. D., (2009). Arquitecturas de red para servicios en Cloud computing. Trabajo de clase de Sistemas Informáticos (no publicado), Curso 2008-2009, Facultad de Informática. Consultado: 30 Agosto 2016. http://eprints.ucm.es/9452/

Cita: (Molina Coballes & Muñoz Rodríguez, 2015)
Entornos de pruebas para OpenStack. Consultado 28 Noviembre 2017.
http://iesgn.github.io/cloud/curso/u3/devstack

Cita: (Universidad de Extremadura, 2018).
El proyecto SmartPoliTech de la Escuela Politécnica, referente nacional para la gestión eficiente de los recursos.
https://www.unex.es. Consultado el 28 de Septiembre de 2018.

Cita: (La Vanguardia, 2017). *Con SmartPoliTech logramos grandes ahorros energéticos en edificios antiguos*.
https://www.lavanguardia.com/monograficos/nuevas-tecnologias/con-smartpolitech-logramos-grandes-ahorros-energeticos-en-edificios

Cita: (OSUNA FONTAN, 2016)
OSUNA FONTAN, L. (2016). *Creación de sistema cloud con OpenStack* (Trabajo Fin de Grado). Universitat Politécnica de Valencia.
https://riunet.upv.es/handle/10251/68958

Cita: Galarza, B., Zaccardi, G., Encinas, D., & Martín Morales, D. (2015). *Análisis de despliegue de una IaaS utilizando Openstack*. Lecture, XXI Congreso Argentino de Ciencias de la Computación. Universidad Nacional del Noroeste de la Provincia de Buenos Aires (UNNOBA) Sede Junín.
http://sedici.unlp.edu.ar/handle/10915/50514