Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Laboratorios virtuales: una solución con infraestructura de virtualización

Proyecto Final de Carrera

Ingeniería Informática

Autor: Antonio Salmerón Bermúdez

Director: Jose Ramón García Escrivá

27-09-2013

Resumen

El objetivo de este proyecto es estudiar la viabilidad de una pequeña infraestructura de computación bajo demanda usando software libre, estudiando los servicios que puede ofrecer y compararlos con otras alternativas disponibles. Tras su implementación con OpenStack se analizan los resultados obtenidos y se proponen posibles mejoras.

Palabras clave: computación remota, virtualización, nube, OpenStack.

Tabla de contenidos

1. Introducción

Presentación

Con el auge de internet, estamos viviendo un crecimiento enorme en cuanto al trabajo remoto en el ámbito de la informática. Este tipo de servicios permite trabajar con software especializado, desde cualquier lugar y en cualquier momento, suponiendo una gran facilidad para los usuarios. Sin embargo, en el ámbito informático universitario no se ofrecen estos servicios tanto como cabría esperar, por diversos motivos técnicos y de recursos. Este proyecto pretende afrontar este problema e implementar una solución para los estudiantes de la Universidad Politécnica de Valencia (UPV).

Las ventajas del trabajo remoto son muchas. Permiten una gran flexibilidad de horarios para estudiantes con empleo. También permiten el acceso a software específico sin necesidad de desplazarse hasta el campus, propicio para alumnos geográficamente distantes. Además, supone una experiencia similar a la que se puede dar en un trabajo real, donde se suelen contratar servidores con alta disponibilidad externos a la empresa.

Características del problema

Para ofrecer una situación lo más cercana a la del entorno laboral, deben estar disponibles una serie de facilidades:

- Disponibilidad ininterrumpida: El acceso a las máquinas deberá ser lo más constante posible, sin restricciones de horario, y permitiendo que los servicios ofrecidos por los usuarios estén disponibles en todo momento.

- Acceso como administrador: Los usuarios deberán tener acceso a la gestión avanzada del sistema para poder realizar su trabajo, con permisos para instalar software y modificar parámetros del sistema operativo.

- Soporte a múltiples usuarios: Deberá haber un soporte para usuarios concurrentes independientes, de forma que varios usuarios aislados puedan realizar su trabajo sin afectar en ningún momento al trabajo de los demás.

Opcionalmente, se ofrecerán también ciertas facilidades como monitorización de la red, almacenamiento compartido, etcétera.

Aproximaciones

Entre las diversas opciones que existen para este problema, destacamos:

- Un equipo por alumno: Esta solución, en principio ideal, resulta completamente impracticable por falta de recursos, además de suponer un desperdicio enorme de los mismos durante el tiempo en el que el alumno no lo está utilizando. Un caso cada vez más interesante es que cada alumno aporte su propio equipo portátil.

- Una imagen local por alumno: Esta solución, implementada en varios laboratorios de la UPV, implica una instalación de sistema operativo independiente para cada usuario. Sin embargo, requiere de la presencia física en el aula a la hora de arrancar el equipo para seleccionar dicha imagen, entre otras limitaciones.

- Una imagen extraíble por alumno: En este caso, el alumno dispondría de una imagen modificable en un medio de almacenamiento extraíble, capaz de ser montado en cualquier equipo. Supone total control para el alumno, pero no permite ofrecer servicios a través de la red.

- Una imagen virtual en red por alumno: Si se implementa correctamente, esta solución ofrece todas las ventajas de un equipo por alumno por una fracción de su coste. Desde un equipo en red puede acceder y controlar su imagen que se encuentra funcionando en un servidor remoto.

- Una imagen virtual por alumno en un *cluster*: Ofrece una solución que, además de tener todas las ventajas anteriormente descritas, es escalable y ofrece una mayor fiabilidad ante fallos.

Propósito

Teniendo esto en cuenta, el propósito del proyecto será ofrecer un soporte para el uso bajo demanda de máquinas virtuales en un *cluster* de equipos. Además se compararán los resultados con lo que hubiéramos obtenido con otras alternativas y se analizarán las capacidades que es capaz de ofrecer a los alumnos. Asimismo se realizarán las modificaciones que se consideren necesarias al software para cumplir con nuestros objetivos, aprovechando las ventajas del uso de software libre.

2. Estado del arte

Software de computación remota

El propósito principal de cualquier nube de computación es ofrecer un servicio a través de internet. Para ello, se ofrece una red de computación de alta disponibilidad con acceso a través de la red y acceso libre bajo demanda. Dependiendo del tipo de servicio que se quiera ofrecer, se han clasificado estos sistemas en tres capas principales[1] :

- Software como servicio (*Sofware as a Service*, SaaS): La capa más alta, ofrece software directamente a los usuarios finales. Los usuarios sólo tienen que abrir su navegador o programa cliente y conectarse a la nube para usar sus servicios.

- Plataforma como servicio (*Platform as a Service*, PaaS): La capa intermedia, ofrece una infraestructura de trabajo con facilidades a los desarrolladores. Típicamente se ofrece una plataforma informática con un sistema operativo, entorno de desarrollo, base de datos y servidor web ya instalados.

- Infraestructura como servicio (*Infrastructure as a Service*, IaaS): La capa inferior, ofrece una infraestructura básica de trabajo dando acceso a máquinas (típicamente máquinas virtuales) y a una serie de herramientas relacionadas como imágenes virtuales, espacio en disco, balanceadores de carga y VLANS.

Al concentrarse este proyecto en esta última categoría (IaaS), se procede a comparar diversas tecnologías que proporcionan este servicio.

OpenStack

A día de hoy, hablar de software de computación en la nube supone hablar de *OpenStack*[2]. *OpenStack* es un proyecto de software libre para proporcionar computación en la nube de tipo *IaaS*. Nacido de una iniciativa conjunta de *Rackspace* y NASA en julio de 2010, su popularidad y uso desde su lanzamiento se ha disparado. Más de 200 compañías, incluyendo a IBM, Cisco, Canonical o Red Hat[3] apoyan el proyecto en la actualidad.

Este ascenso de debe, entre otras cosas, a la licencia *Apache*[4] en la que de distribuye. OpenStack es un software completamente libre, con una licencia más permisiva que la licencia GPL[5] (el estándar de facto del software libre). Esto implica que cualquiera puede notificar *bugs*, desarrollar complementos o incluso programas enteros que utilicen su interfaz de programación de aplicaciones (API, por sus siglas en inglés). Además, *OpenStack* provee de compatibilidad con la API de Amazon EC2, lo cual facilita el proceso de portar software desde esta popular plataforma.

Otro de sus grandes puntos a favor es su modularidad. *OpenStack* está formado por diferentes componentes. Estos componentes pueden ser instalados en cualquier máquina de la nube e incluso movidos de una máquina a otra sin problemas. Algunos de ellos pueden ser incluso sustituidos por otras alternativas, siempre que utilicen una API soportada.

- *OpenStack Compute* (alias *Nova*), es el servicio de control de computación. A través de Nova se controlan las diversas herramientas de virtualización en uso por la nube. Se utiliza tanto en las máquinas de cómputo como en las máquinas de control.

- *OpenStackObject Storage* (alias *Swift*) es el servicio de almacenamiento redundante. Permite almacenar o recuperar ficheros (pero no montar directorios). Objetos y ficheros son distribuidos y replicados automáticamente a través de la red a los nodos de almacenamiento.

- *OpenStack Block Storage* (alias *Cinder*) es el servicio de almacenamiento de bloques. Permite la persistencia de los datos entre ejecuciones de la instancia.

- *OpenStackNetworking* (alias *Quantum*) es el servicio de control de red. Además de controlar la red física de las máquinas, permite a los usuarios crear redes y con ellas conectar instancias entre sí.

- *OpenStackDashboard* (alias *Horizon*) es la interfaz gráfica web de todos los componentes de OpenStack. A través de Horizon se pueden realizar la mayoría de operaciones de control, como asignar direcciones IP o lanzar una instancia.

- *OpenStackIdentityService* (alias *Keystone*) es la plataforma de autentificación y autorización que usan todos los componentes de OpenStack. También ofrece un catálogo de los servicios disponibles en uso en una nube.

- *OpenStackImageService* (alias *Glance*) es el servicio de gestión de imágenes de disco. Ofrece un catálogo y repositorio de imágenes disponibles para el uso en computación con Nova.

El uso y configuración de los diversos módulos hace de *OpenStack* una opción muy versátil, aplicable en multitud de situaciones diferentes.

Otras tecnologías

Además de *OpenStack*, hoy en día hay muchas tecnologías disponibles para el despliegue de una nube de computación. En este apartado compararemos algunos proyectos y tecnologías disponibles como software libre que complementan o presentan una alternativa a *OpenStack*.

- *DevStack*[6]: Se trata de un script bash de despliegue de Openstack en Ubuntu o Fedora de forma desatendida. Utiliza el software de control de versiones git para acceder al repositorio oficial y descargar la última versión estable. Entre sus opciones podemos ver el soporte a XenServer, OpenVZ y LibVirt.

- *StackOps*[7]: Distribución Linux especializada en OpenStack, basada en Ubuntu. La facilidad que ofrece se trata de una interfaz web (instalada localmente) que permite una instalación simplificada paso a paso. No ofrece ninguna otra ventaja ya que tras el despliegue queda una distribución Ubuntu con Openstack que se debe manejar usando Horizon. La documentación no es muy específica y al tratarse de la versión "Community" (en lugar de la "Enterprise", la versión comercial de pago) no hay mucho soporte disponible.

- *CloudStack*[8]: Al igual que OpenStack, CloudStack es un software de gestión con soporte para hipervisores como KVM y XenServer para la virtualización, soporte para la API de AWS y soporte para Swift (el almacenamiento de Objetos de OpenStack). Tras convertirlo a software libre, recientemente fue donado a Apache para evitar que el proyecto terminara.

- *RockClusters*[9]: Distribución Linux para clusters de computadores de alto rendimiento. Basada en CentOS, es muy popular por su versatilidad, su soporte a diferentes servicios y su facilidad de ampliación. Es relativamente fácil de usar dado que es un software para investigadores y no para informáticos, pero al ser tan concreto para el cálculo no resulta apropiado para otros servicios de los que puede disponer una nube.

- *Unicore*[10]: Se trata de un Middleware en Java para Grid Computing. Al estar desarrollado en Java permite que los equipos en la red sean heterogéneos, pudiendo usar sistemas Windows, Linux o Mac. La herramienta de manejo está basada en Eclipse aunque también dispone de un cliente para línea de comandos.

- *Abiquo*[11]: Es un toolkit basado en la web desarrollado en Java. Soporta una gran variedad de hipervisores y sistemas de almacenamiento. Dispone de una versión de pago (Enterprise Edition) y una versión Libre (CommunityEdition). Actualmente la versión libre no está disponible para descarga sino sólo el código fuente.

	StackOps	**DevStack**	**CloudStack**	**RockClusters**	**Unicore**	**Abiquo**
Soft. Libre	Sí (*)	Sí	Sí	Sí	Sí	Sí (*)
OpenStack	Sí	Sí	No	No	No	No
Cloud-oriented	Sí	Sí	Sí	No	No	Sí

(*) Disponen de versión Enterprise y versión Community

3. Planteamiento y objetivos

Objetivo

El objetivo final de este proyecto es ofrecer a los usuarios una infraestructura para que puedan administrar y ejecutar máquinas virtuales en cualquier momento a través de internet. Para ello se ofrece un soporte para el uso bajo demanda de máquinas virtuales en un *cluster* de equipos. Como se ha mencionado en el apartado anterior, para ello se utilizará *OpenStack* por su alto soporte y su flexibilidad.

A partir de los recursos que nos ofrece *OpenStack*, podemos concretar la funcionalidad a ofrecer. Se dará acceso a la nube a múltiples usuarios, que no tendrán en ningún momento acceso directo al hardware. Los usuarios podrán lanzar instancias a través de la interfaz web y acceder a ellas remotamente. Además se proporcionará unas imágenes virtuales predefinidas como plantilla y puntos de partida de esas instancias, las cuales podrán guardar datos y el estado de la instancia en disco.

Arquitectura

En la evaluación inicial se decidió optar por una infraestructura de red de cuatro nodos, cada uno conectados a una red compartida. Las cuatro máquinas tienen un procesador de 4 núcleos y 64 bits y de 4 a 8 GB de memoria RAM. La división de los servicios será la siguiente: Una máquina de nodo controlador, para los servicios de identificación, interfaz y gestión de recursos, a la que llamaremos “Uno”. Dos máquinas de nodos de computación, para correr las instancias, a las que llamaremos “Dos” y “Tres”. Finalmente, una máquina de nodo de almacenamiento, para guardar las imágenes virtuales y otros objetos, a la que llamaremos “Cuatro”.

Los nodos se conectan entre ellos a un switch *Ethernet*, que a su vez está conectado a una máquina proxy. Esta máquina es la que utiliza el redireccionamiento de puertos para servir de pasarela ssh a cada uno de los nodos individuales. La Figura 1 muestra esta arquitectura de red.

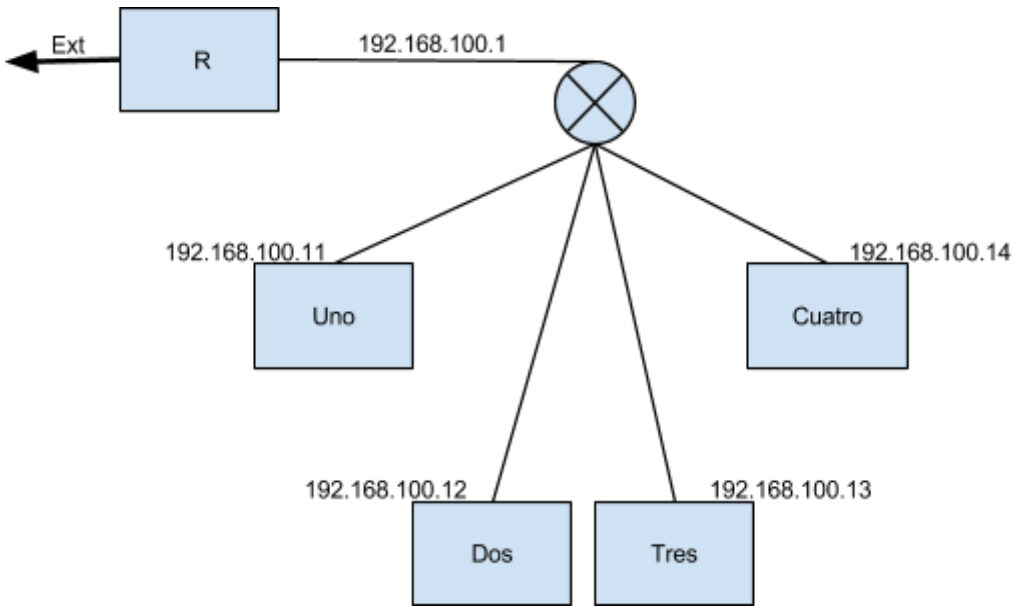

Figura 1: Arquitectura inicial
Fuente: Elaboración propia

4. Memoria y metodología

DevStack

Se escoge *DevStack* para realizar la instalación y despliegue al estar recomendado tanto en la web de OpenStack como en la de Ubuntu. *DevStack* es un script de ejecución bash que, usando el software de control de versiones *Git*[12], descarga y compila la última versión de *OpenStack*. Tras editar un fichero de configuración con las variables apropiadas, el proceso de instalación y despliegue está completamente automatizado de forma desatendida.

Se realiza pues un primer despliegue en una carpeta local. En el momento de utilizar *Keystone* para identificarse como usuario se encuentra el primer problema. Usando el token de servicio para ignorar la identificación se crean nuevos usuarios y roles e indicando la ubicación en red y los servicios de las otras máquinas. Se configura a su vez *Horizon*, el panel de control web de *OpenStac*k y la herramienta principal que usarán los usuarios. Sin embargo, al no estar funcionando correctamente la identificación, no se puede acceder al mismo y por tanto usar el software.

El problema que se encuentra es que cuando la máquina en la que se ejecuta *DevStack* se reinicia no todos los servicios de *OpenStack* se ponen en marcha. Esto es una decisión de diseño por parte de *DevStack*, que se considera una herramienta de pruebas y no una instalación permanente. La solución es ejecutar *rejoin-stack.sh* de la carpeta de instalación, para que todos los servicios se comprueben y se lancen aquellos que no se encuentren en ejecución. Sin embargo *Devstack* en su versión actual tiene un error en ese script, que se queda en intentando montar la partición loopback de Swift y fallando en un bucle infinito y por tanto no puede reiniciarse correctamente.

Tras este problema, se intenta una desinstalación de DevStack, deteniendo los servicios y eliminando los ficheros de la carpeta de instalación. Sin embargo, la instalación había dejado ficheros de configuración por todo el sistema de ficheros y modificado los ficheros de otros servicios secundarios, por lo que decidimos optar por una estrategia conservadora y formatear las máquinas con un Ubuntu Server desde cero.

OpenStack desde repositorio

A partir de la habituación que se consigue usando la instalación creada por *DevStack*, seguimos el proceso de instalación oficial de *OpenStack*[13] para esta versión. Se comienza por el nodo controlador "Uno" ya que es el más complejo y el que más servicios requiere.

Tal y como se muestra en la documentación, se escoge *MySQL*[14] como proveedor para las bases de datos y se crean las tablas a usar por *OpenStack*. También se instala *RabbitMQ*[15] como manejador de colas. Tras configurar estos servicios secundarios, se instala *Keystone*, el servicio de identificación, y se configura adecuadamente. También se ejecuta un script[16] de la documentación para introducir los usuarios, servicios y puntos de acceso en la base de datos de la autentificación.

Tras esto se realiza la instalación y configuración de *Glance*, siguiendo la misma guía. Como comprobación se descarga una pequeña imagen ISO de *CirrOS* (una distribución *Linux* ligera) y se añade a *Glance*, el gestor de imágenes virtuales. Con un pequeño comando podemos comprobar que se ha subido con éxito y está disponible para todos los usuarios. En el siguiente paso se instala y configuran *Nova* y *Cinder*, los servicios de cómputo y almacenamiento persistente del estado de las instancias. Mientras tanto, se procede a la instalación de *Nova* y *Quantum* en los nodos "Dos", "Tres" y "Cuatro", y se dejan listos para recibir las peticiones del nodo controlador "Uno".

Tras esto se procede a la instalación y configuración de la red de "Uno". Sin embargo, al utilizar una única interfaz de red, surgen problemas con la configuración de la red privada virtual (VLAN, por sus siglas en inglés), configurada usando *OpenVSwitch*[17]. Al crear un puente y puerto sobre la interfaz de red en uso, se pierde la conexión SSH que se estaba usando para realizar la configuración. Tras documentarse al respecto de este problema, se decide que la máquina está incomunicada sin remedio. Para solucionarlo se hace uso del teclado físico del equipo y se desinstala *OpenVSwitch* mientras se barajan otras alternativas de conexión.

Modificación de la arquitectura de red

Tras una larga búsqueda de documentación al respecto, se determina que es demasiado complejo intentar mantener la arquitectura de red inicial. El acceso SSH y la comunicación entre máquinas vía VLAN no es posible por la misma interfaz de red. Por lo tanto, se añade un nuevo switch a los ordenadores internos, quedando por tanto una red (192.168.100.0) para la gestión directa y acceso a internet, y otra red (192.168.200.0) para la comunicación interna. Además, debido a un problema inesperado de hardware, hubo que retirar la máquina "Cuatro", quedando la arquitectura final como se muestra en la Figura 2.

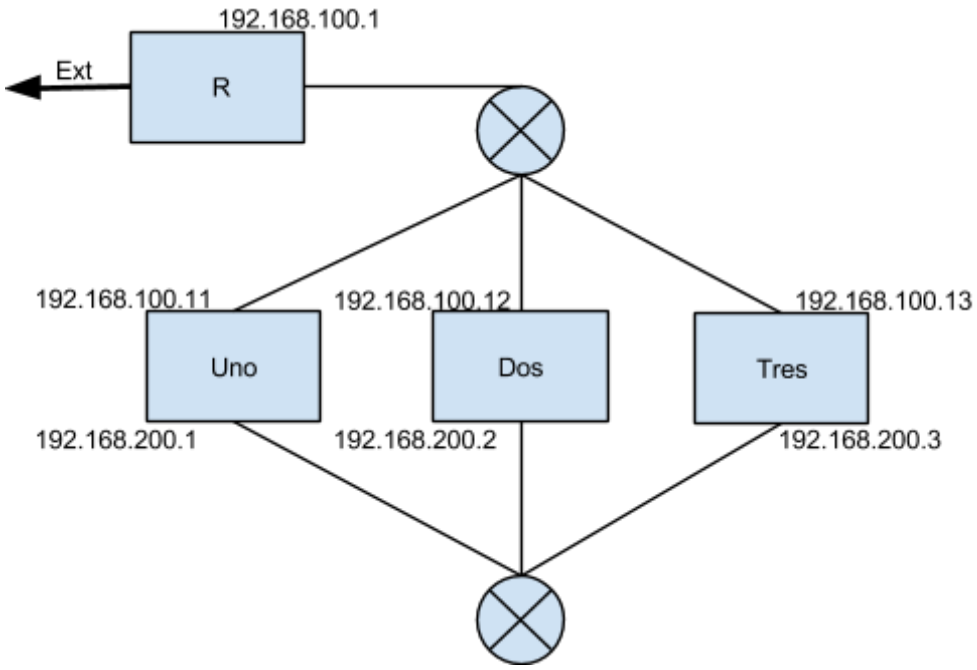

Figura 2: Arquitectura de red modificada
Fuente: Elaboración propia

La nueva red interna es configurada adecuadamente en el fichero de configuración "/etc/network/interfaces", tal que la máscara de red se procese por la nueva interfaz ethernet "eth1". Tras este cambio se reconfiguran los servicios de las máquinas para que usen las direcciones IP de la red interna. Esto produce un error que no se logra identificar en la máquina "Uno", al que se le aplica una reinstalación de los servicios. Más tarde se determina que el error se produjo al intentar sincronizar los cambios con la base de datos sin ejecutar como super-usuario (el usuario administrador del sistema *Linux*). Aunque la mayoría de programas de *OpenStack* utilizan variables de entorno para realizar la identificación a través de *Keystone*, algunos de los programas de gestión utilizan el super-usuario para aplicar los cambios necesarios directamente sobre la base de datos.

Tras esto surge un nuevo problema, esta vez con *Horizon*. Al intentar entrar en la interfaz web, *Apache*[18] devuelve una página de error con una excepción *Python*[19]. Se logra localizar el error en una orden *import* de una de las bibliotecas de *Python*. Tras una búsqueda de casos similares en internet, se determina que se trata de un problema de la versión más actual de *Django*[20]. Se desinstala el paquete *python-django* y reinicia el servicio sin más contratiempos, pasando *Horizon* a utilizar la versión de *Django* específica instalada al instalar el propio *Horizon*.

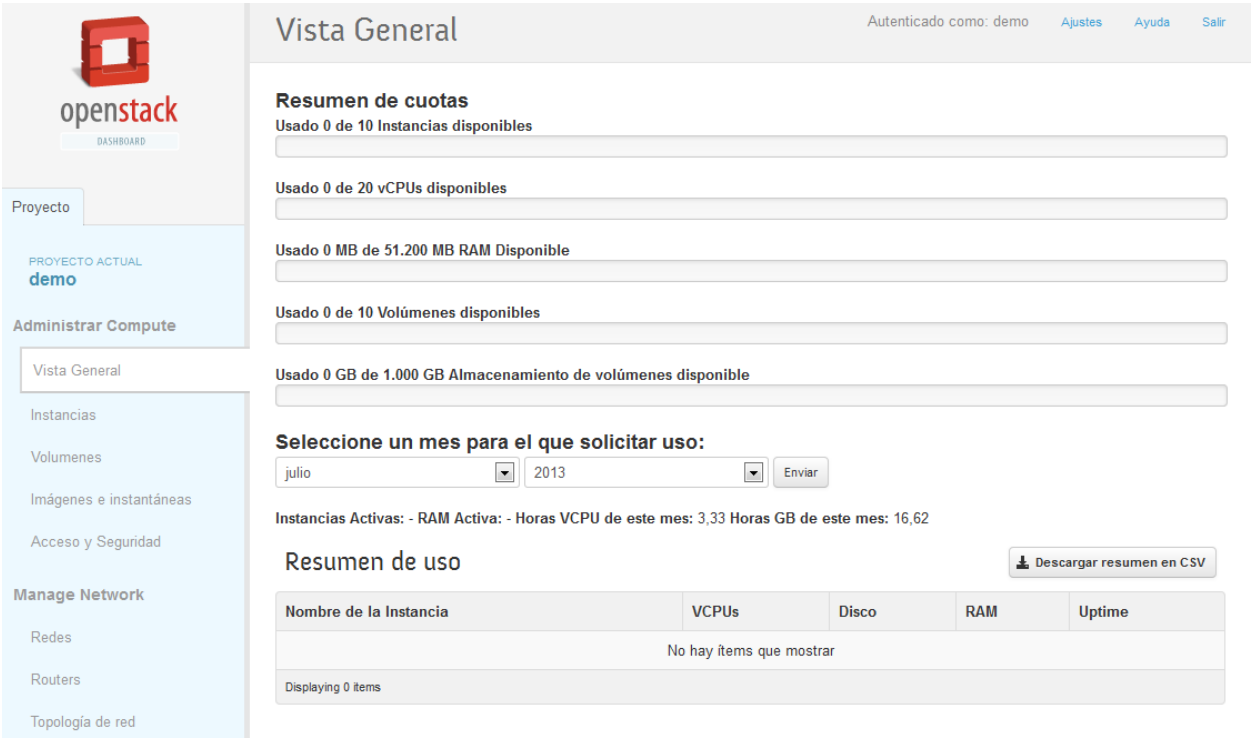

Figura 3: Página inicial de Horizon
Fuente: Elaboración propia

Redes virtuales con OpenVSwitch

Tras tener todos los componentes instalados y configurados correctamente en los nodos, se pasa a intentar que se comuniquen. Gracias a la nueva arquitectura de red se puede seguir con la guía de la documentación oficial. Para ello se instala de nuevo el plugin de OpenVSwitch en los nodos y se configura un puerto virtual en las interfaces eth1, las interfaces de la red interna. Además, se añade una interfaz de red virtual "br-ex" que sustituye a eth0 en "Uno". Conectando a "Dos" y usando ssh para conectarse por la red interna a "Uno" seguimos con la guía. La interfaz virtual "br-ex" se redirige a "eth0" mediante tres entradas en la tabla *iptables* (la tabla de enrutamiento de *Linux*). Sin embargo tras muchas pruebas e intentos, esta redirección no logra funcionar y por lo tanto "Uno" pierde su conexión de red a internet. Tras una búsqueda intensiva de problemas en la configuración o logs, se determina que el problema es el soporte para VLANS del driver y/o la tarjeta de red[21].

Conexión con IP fija

Tras los problemas encontrados usando OpenStack, se intenta una aproximación más directa aunque más inusual: configuración directa con direcciones IP estáticas, siguiendo de nuevo la documentación oficial al respecto[22]. Tras unos cuantos intentos fallidos de comunicación, finalmente se encuentra una referencia al paquete "openstack-conductor" como intermediario entre la base de datos y "openstack-compute". Tras instalarlo, una simple ejecución de "nova-manage service-list" muestra un servicio *nova-compute* disponible en la máquina "Dos". Tras un reinicio de la máquina "Tres", ésta también se muestra disponible.

Tras terminar la configuración, se intenta comprobar la conexión mandando ejecutar una instancia desde *Horizon*[23]. Sin embargo, al intentarlo la instancia se coloca en un estado de error. Tras revisar el procedimiento que se lleva a cabo para ejecutar una instancia y revisar los logs de los servicios que se utilizan, se determina que se trata de un error del sistema de paso de mensajes *RabbitMQ*[24]. Esto se resuelve finalmente habilitando el bridge "br-int" y reiniciando el servicio de *OpenVSwitch* en las máquinas "Dos" y "Tres". Al no usar un puerto virtual, *OpenVSwitch* puede comunicarse por la red sin usar en exclusiva la tarjeta de red. Finalmente se tiene una nube funcional que ejecuta las imágenes sin problemas, conectadas usando túneles GRE, sin un nodo controlador de red.

Nombre de la Instancia	Dirección IP	Tamaño	Par de clave	Estado	Tarea	Estado de Energía	Acciones
instancia2		m1.tiny | 512MB RAM | 1 VCPU | 0 Disco	-	Build	Networking	No State	Asociar IP Flotante | Más
instancia1	192.168.200.10	m1.tiny | 512MB RAM | 1 VCPU | 0 Disco	-	Active	None	Running	Crear instantánea | Más

Displaying 2 items

Figura 4: Estado de las instancias en el Dashboard
Fuente: Elaboración propia

El siguiente paso es conectarse a las instancias en ejecución. Para ello se utiliza una redirección de puertos adicional en el servidor proxy, que se encamina al puerto 6080 de la máquina controlador, "Uno". Editando la URL de acceso en los nodos de computación a la url del proxy, se integra el acceso VNC en el dashboard. De esta forma se logra acceso a las instancias en ejecución, a pesar de que las mismas no adquieran conexión. A continuación se intenta resolver este problema y otorgar una conexión SSH.

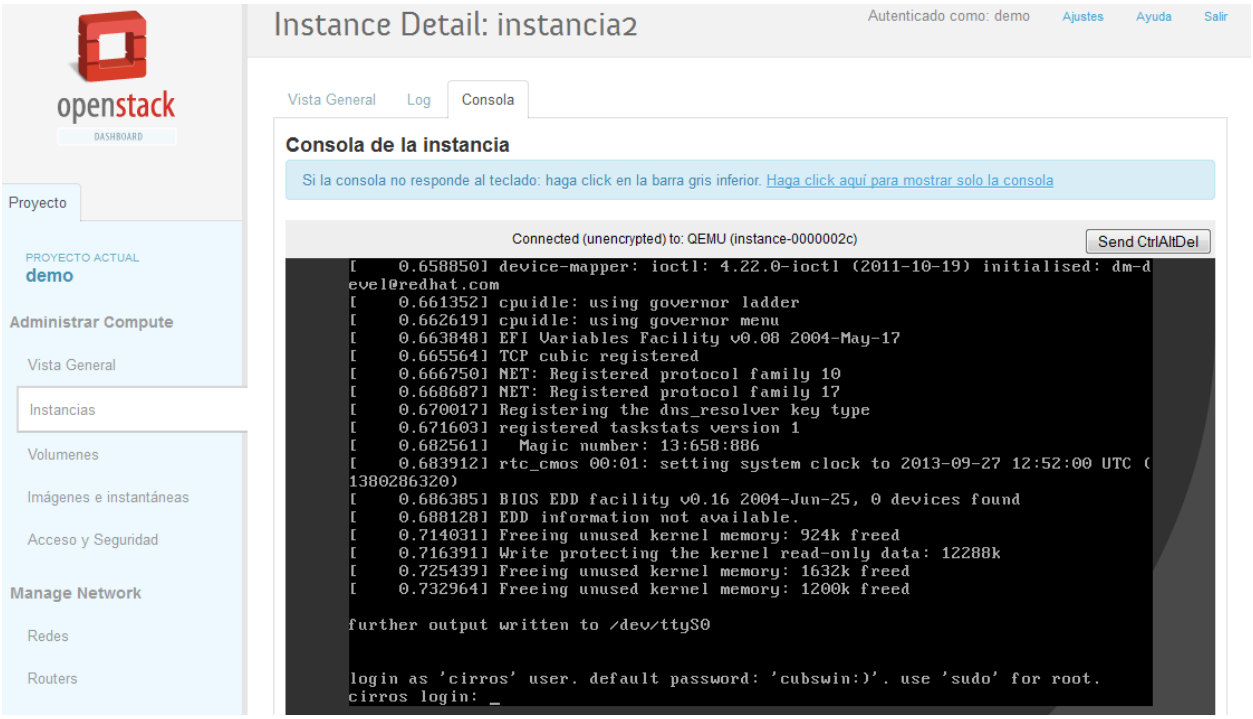

Figura 5: Acceso VNC a una instancia a través del Dashboard
Fuente: Elaboración propia

En este momento se encuentra con un problema de *ip-netns*, el servicio de namespaces de Linux. Un bug[25] impide que, al eliminar redes y routers virtuales en *Quantum*, éstos se eliminen del servicio de namespaces de Linux y por tanto que el direccionamiento en redes virtuales (y, a su vez, entre máquinas virtuales) no funcione. Sin embargo, el uso de este servicio es opcional[26], ya que sólo está disponible desde Ubuntu 12.04 en adelante. Por lo tanto se procede a desactivar este servicio y reiniciar las máquinas.

Tras esto se encuentra con otro problema más. Los servicios no se encuentran en sus puertos y no hay forma de alcanzarlos. Resulta ser un problema de *OpenVSwitch*: al reiniciar las máquinas se aplica una actualización del kernel que impide el correcto funcionamiento del mismo. Se instala el módulo necesario para corregir este problema[27] (*openvswitch-datapath-source*) y reiniciamos. Sin embargo, el problema persiste.

Finalmente se determina que la actualización de kernel ha actualizado a su vez ficheros de configuración críticos no compatibles con el software instalado en las máquinas. Esto implica por tanto un downgrade a la versión anterior del kernel no solucionaría el problema tampoco. Se procede pues a formatear las máquinas de nuevo y repetir la instalación hasta donde se había llegado. Siendo nuestro problema principal la conexión a red, se decide cambiar el nodo "Dos" de nodo de cálculo a nodo de red para una mayor independencia de servicios y poder analizar mejor los problemas de conexión.

Se instala en el nodo "Dos" por tanto los servicios *dhcp-agent* y *l3-agent*, los cuales dan direcciones IP a las máquinas virtuales y conexión a internet respectivamente. Sin embargo, al configurar los puertos virtuales perdemos la conexión a las máquinas "Uno" y "Tres", dejando la máquina de servicios de red aislada de la nube. Finalmente, tras notar que la conexión a internet no se había perdido, determinamos que es un problema de enrutamiento. Las conexiones a los nodos "Uno" y "Dos" no pasan a la red ya que se enrutan hacia la red virtual. Eliminamos los puertos virtuales que se habían creado con la herramienta de *OpenVSwitch* para consola *ovs-vsctl* y se solventa el problema.

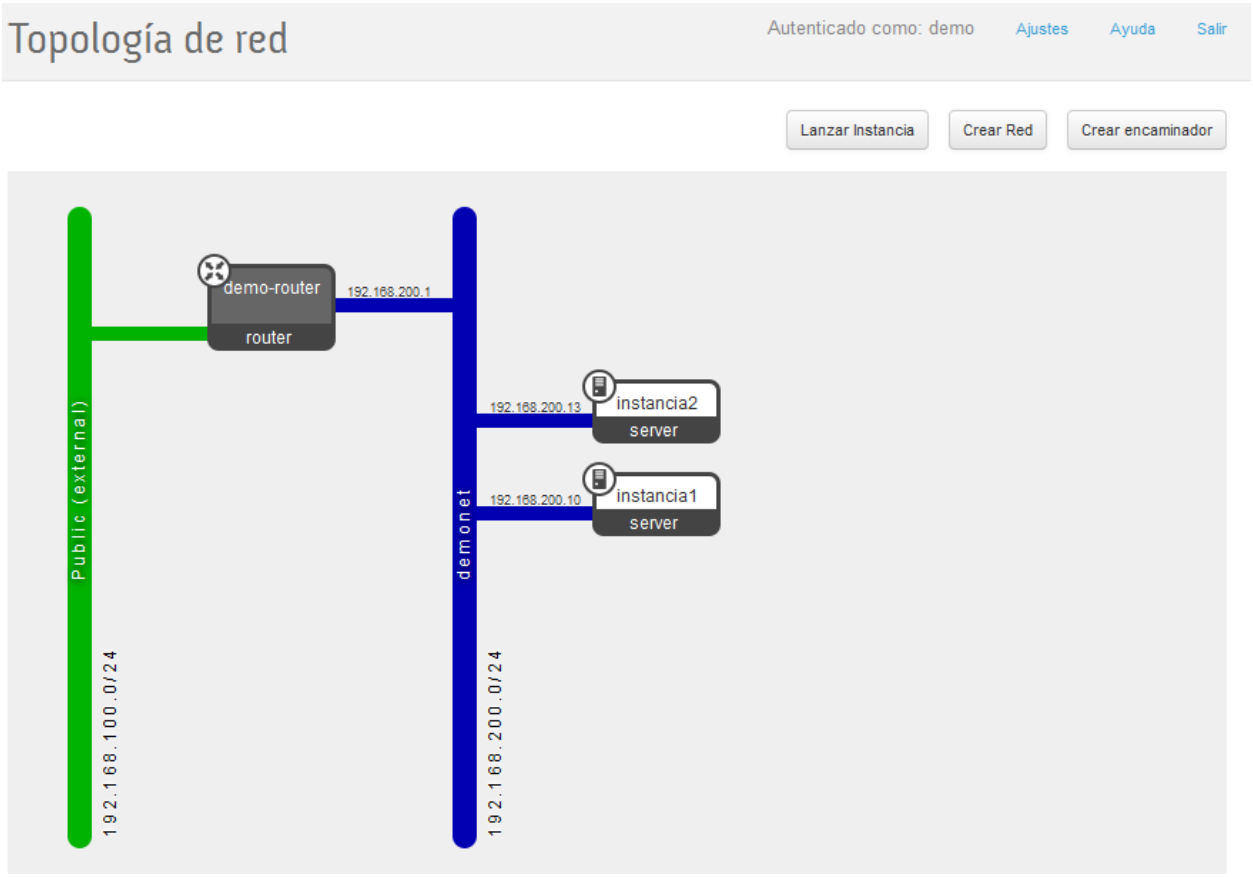

Figura 6: Topología de la red como se muestra en el Dashboard
Fuente: Elaboración propia

Tras documentarse mejor sobre el funcionamiento del propio *dhcp-agent*, se eliminan las direcciones IP de la red interna 192.168.200.0/24 de forma que la máquina host pueda aceptar las peticiones dhcp enviadas a las máquinas virtuales. A través del servicio ip-netns y el namespace virtual se consigue enviar un ping a la instancia en ejecución y recibir respuesta. También se prueba a acceder mediante SSH y también se consigue con éxito. Las instancias reciben con éxito las direcciones IP asignadas por *dhcp-agent* y son capaces de comunicarse entre sí (estando en la misma red virtual).

Limitaciones del hardware

Finalmente se determina que uno de los propósitos fundamentales del proyecto, ofrecer máquinas virtuales con conectividad a internet, resulta imposible según el hardware disponible. Como se ha mostrado en anteriores diagramas, disponemos de dos tarjetas de red por nodo y dos redes en total. Según esto, se ha intentado aplicar la siguiente arquitectura:

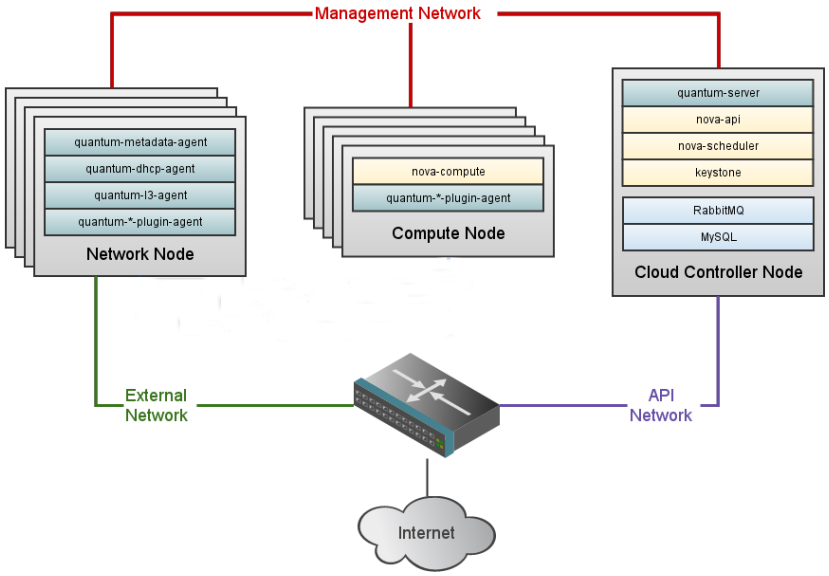

Figura 7: Arquitectura de red simple
Fuente: http://docs.openstack.org/grizzly/basic-install/apt/content/basic-install_architecture.html

Sin embargo, esto no ha sido posible ya que no se ha podido compartir la conexión de datos y la conexión de control en la misma interfaz, habiéndose intentado mediante túneles GRE y VLANs. El problema de conectividad se puede reducir a esa simple cuestión: compartir la línea de control y la línea de datos en una misma tarjeta de red.

Las soluciones propuestas por OpenStack para este problema pasan por usar OpenVSwitch para tomar control de la interfaz de red y separar los tráficos mediante identificadores GRE o VLAN, permitiendo a *OpenVSwitch* redirigir todo el tráfico según sus *iptables*. Como ya se ha mencionado anteriormente, esto no es una opción disponible para nuestras máquinas, que no soportan VLAN y redirección de tráfico según identificadores GRE. Otros plugins de control de red, como *LinuxBridges*, son mucho más limitados ya que no ofrecen túneles GRE y su documentación de uso con *OpenStack* es muy escasa. Tras un infructuoso intento de utilizar *LinuxBridges*, se pierde la conexión entre los nodos. Se restaura pues el plugin de red a *OpenVSwitch* y se restaura la conexión a su estado anterior. Se determina pues que *OpenVSwitch* sigue siendo la mejor alternativa de la que se dispone actualmente y que el estado actual del proyecto es el más avanzado que se puede conseguir con el hardware disponible.

5. Presentación de resultados

En la iteración final del proyecto se dispone de tres máquinas conectadas entre sí. Una de ellas actúa de controlador y proporciona la interfaz web, otra proporciona la conexión a las máquinas virtuales y la tercera es la máquina de cómputo.

Se ofrece pues una plataforma para el trabajo remoto, disponible ininterrumpidamente y sin restricciones de horario, abierta al uso bajo demanda. A través de la interfaz web se ofrecen diversas opciones de potencia de cálculo y tamaño de memoria a elegir dependiendo de la tarea a realizar. Además, se ofrecen diversas imágenes de arranque diferentes según la preferencia de los usuarios: CirrOS 0.3.1, Ubuntu 12.04, Fedora 19.

Figura 8: Imágenes virtuales e instantáneas de instancias disponibles según el Dashboard
Fuente: Elaboración propia

El acceso a las máquinas virtuales puede hacerse mediante la propia interfaz web, vía VNC, o mediante un túnel SSH desde la máquina “Dos”. Cada usuario tiene acceso a todas las instancias en ejecución sobre las que tenga permiso, y en todas ellas tendrá acceso administrador para poder llevar a cabo las tareas de configuración que necesite. El acceso es concurrente, de forma que varios usuarios pueden trabajar en varias instancias al mismo tiempo sin interferirse entre ellos.

A pesar de no haberse realizado una prueba de carga exhaustiva, el sistema soporta al menos cuatro máquinas virtuales pequeñas sin presentar problemas ni congestión de tráfico.

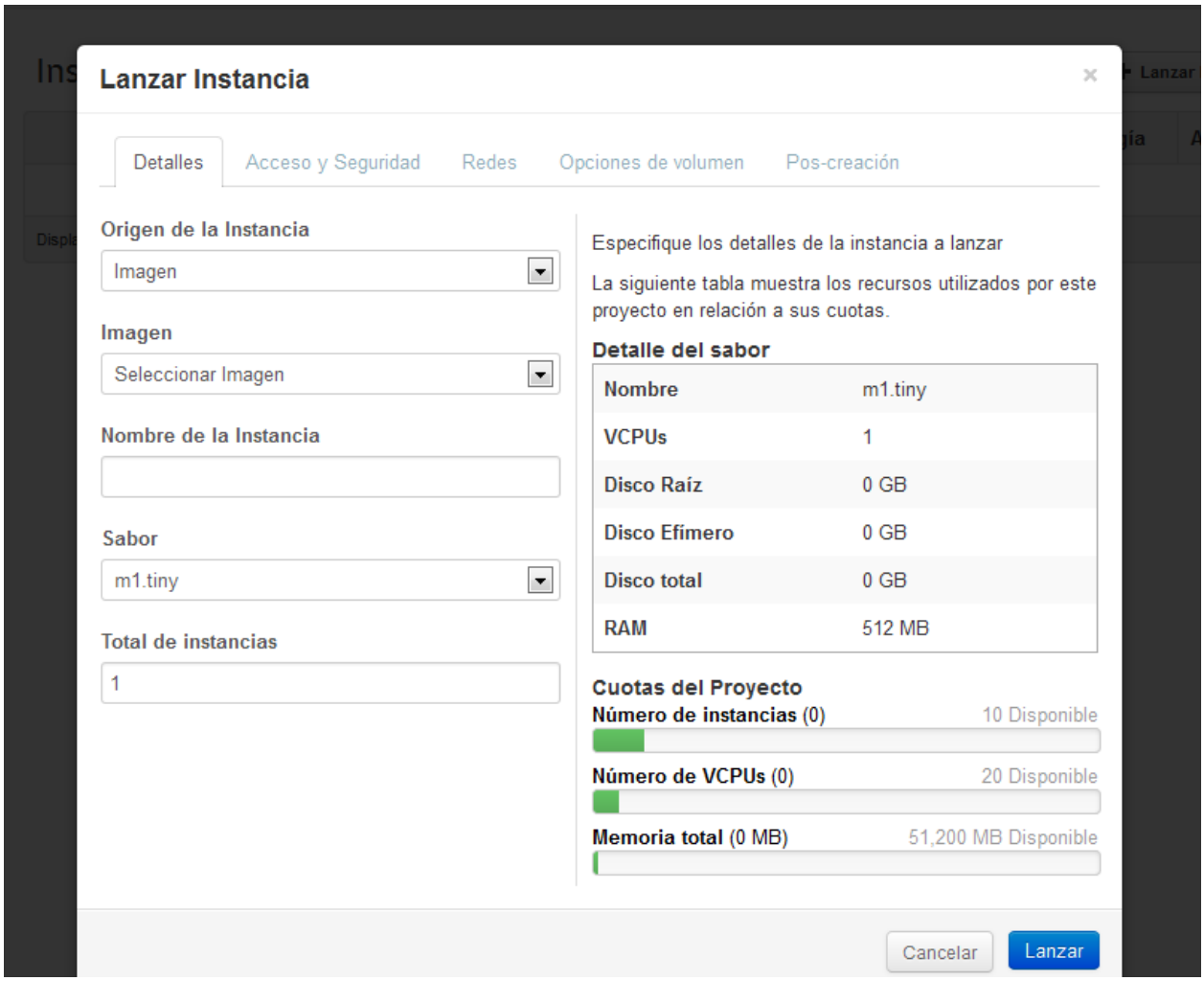

Figura 9: Diálogo para lanzar una instancia desde el Dashboard
Fuente: Elaboración propia

En cuanto a la conectividad a internet, este aspecto limita mucho el aspecto práctico del sistema. Mientras que en nuestra propuesta inicial se pretendía que los usuarios pudieran arrancar y detener a su gusto servicios como páginas web, descarga de ficheros o servidores de correo, en la práctica nada de esto es realizable. Se dispone pues de una “caja cerrada”, que permite la entrada de usuarios para que realicen las tareas que necesiten y vuelvan a salir. Esto se puede aplicar a cálculos o compilaciones en los que se pueda introducir los ficheros mediante SSH, realizar la computación y retirar los resultados mediante SSH de nuevo. Mediante la creación de imágenes virtuales personalizadas se puede optimizar este tipo de servicio de computación desatendida.

6. Mejoras propuestas

La solución propuesta para conseguir la conectividad se trata de añadir una tercera interfaz de red al nodo de red y conectarlo al nodo de computación, como se muestra en el siguiente diagrama:

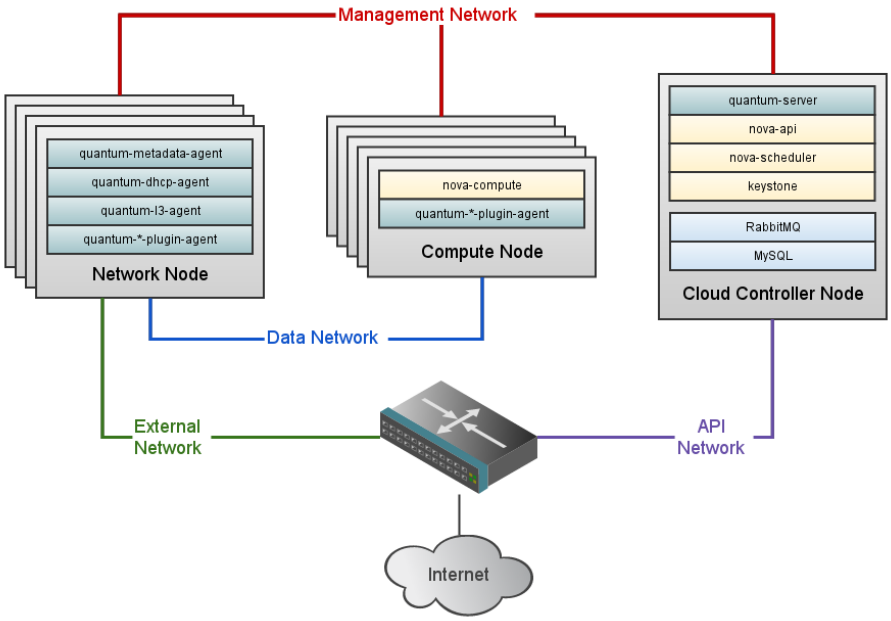

Figura 10: Arquitectura de red con redes independientes
Fuente: http://docs.openstack.org/grizzly/openstack-network/admin/content/app_demo_single_router.html

De implementarse esta arquitectura, los problemas de red se solventarían ya que cada una de las redes estaría perfectamente separada de las demás mediante tarjetas de red independientes. Se seguiría aplicando el modelo DHCP para las máquinas virtuales, permitiendo una escalabilidad sin más problemas que replicar los nodos de cálculo. El punto débil de esta arquitectura queda en el nodo de red, del cual en caso de perder una de sus conexiones dejaría al sistema prácticamente inutilizable. Éste nodo también es replicable, aunque al necesitar una tarjeta de red más y no proporcionar potencia de cómputo resulta muy caro como simple salvaguarda.

7. Conclusiones

La experiencia que se ha podido sacar de este proyecto es que crear y mantener una infraestructura de este tipo no es algo trivial. A pesar de la abrumadora cantidad de soporte que tiene OpenStack en la red, cada caso es esencialmente diferente ya que cada sistema tiene su propia arquitectura de red, su propia división de los servicios por máquinas, y su uso de diferentes plugins y tecnologías subyacentes.

Dicho esto, cada error ha sido una oportunidad más de aprender sobre el funcionamiento y la relación que hay entre los diferentes módulos de OpenStack.

También se debe destacar que, haciendo uso del propósito del proyecto, este trabajo se ha llevado a cabo en gran parte a distancia, a través de una conexión SSH. Y aunque ha sido una experiencia positiva, hay ciertas tareas para las que no hay sustituto de un monitor y un teclado físicos, sobre todo en tareas de mantenimiento y configuración de interfaces de red.

8. Referencias bibliográficas

- [1] "The NIST Definition of Cloud Computing". National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
- [2] “About OpenStack”. OpenStack Foundation, http://www.openstack.org/software/
- [3] “Companies Supporting the OpenStack Foundation”. OpenStack Foundation, http://www.openstack.org/foundation/companies/
- [4] “Apache License, Version 2.0”. The Apache Software Foundation, http://www.apache.org/licenses/LICENSE-2.0.html
- [5] “GNU General Public License”. Free Software Foundation, http://www.gnu.org/licenses/gpl.html
- [6] “DevStack Overview”. OpenStack Foundation, http://devstack.org/overview.html
- [7] “What is StackOps”.StackOps, http://www.stackops.org/
- [8] “About Apache CloudStack”. The Apache Software Foundation, http://cloudstack.apache.org/about.html
- [9] “About Rocks”, rocksclusters.org, http://www.rocksclusters.org/wordpress/?page_id=57
- [10] “Unicore - Objectives”. UNICORE, http://www.unicore.eu/unicore/
- [11] “Product Overview”. Abiquo, http://www.abiquo.com/product/overview/
- [12] “Git - About”. Git, http://git-scm.com/about
- [13] “OpenStack Basic Installation Guide for Ubuntu 12.04 (LTS)”, OpenStack Foundation, http://docs.openstack.org/grizzly/basic-install/apt/content/index.html
- [14] “MySQL Editions”, Oracle Corporation, http://www.mysql.com/products/
- [15] “What can RabbitMQ do for you?”.GoPivotal Inc., http://www.rabbitmq.com/features.html

- [16] “Basic Install Controller Node”. OpenStack Foundation, http://docs.openstack.org/grizzly/basic-install/apt/content/basic-install_controller.html#basic-install_controller-keystone

- [17] “About”. Open VSwitch, http://openvswitch.org/

- [18] “About the Apache HTTP Server Project”, The Apache Software Foundation, http://httpd.apache.org/ABOUT_APACHE.html

- [19] “About Python”. Python Software Foundation, http://www.python.org/about/

- [20] “Django at a glance”. Django Software Foundation, https://docs.djangoproject.com/en/1.5/intro/overview/

- [21] “Bridging breaks VLAN support for RTL8111/8168B PCI Express Gigabit Ethernet controller on Karmic”. Mark Ziesemer, https://bugs.launchpad.net/ubuntu/+source/linux/+bug/499766

- [22] “Configurin Flat Networking”, OpenStack Foundation, http://docs.openstack.org/admin-guide-cloud/content/configuring-flat-networking.html

- [23] “Launching instances using Dashboard”, OpenStack Foundation, http://docs.openstack.org/admin-guide-cloud/content/Launching_Instances_using_Dashboard.html

- [24] “Why can't Nova Conductor connect to rabbitmq?”.OmidKosari, https://ask.openstack.org/en/question/1065/why-cant-nova-conductor-connect-to-rabbitmq/

- [25] “Quantum: root cannot access network namespaces created by Quantum service”. Dan Prince, https://bugzilla.redhat.com/show_bug.cgi?id=872689

- [26] “Chapter 11. Limitations”. OpenStack Foundation, http://docs.openstack.org/trunk/openstack-network/admin/content/ch_limitations.html

- [27] “OpenStack – Quantum – Open vSwitch – datapath for tunnels or patch ports”, VentzPetkov, http://blog.vpetkov.net/2013/08/31/openstack-quantum-open-vswitch-datapath-for-tunnels-or-patch-ports/