

Requisitos funcionales

Los usuarios podrán identificar e ingresar en el sistema OpenStack gracias a las credenciales almacenadas en el Directorio Activo de Microsoft.

Cada usuario dispondrá de distintos permisos y funciones acordes a su rol en el Directorio activo.

Dos distintos niveles de acceso según los grupos configurados en el Directorio Activo, 'admin' y 'member'.

Cada usuario del sistema tendrá un proyecto propio en donde podrá crear sus máquinas de forma privada.

Los usuarios podrán subir sus propias imágenes al sistema a partir de las cuales se podrán crear instancias de máquinas virtuales.

Los usuarios dispondrán de una serie de imágenes generadas con antelación para poder crear instancias rápidamente a partir de ellas.

Toda configuración realizada será correctamente documentada para poder ser imitada por una persona distinta al autor de este proyecto. La documentación estará escrita de forma clara y sencilla, explicando cada uno de los puntos. Cualquier persona con unos conocimientos mínimos de redes, sistemas operativos y virtualización será capaz de replicar un sistema similar al expuesto, en cualquier otro entorno a partir de esta documentación.

Requisitos no funcionales

El sistema estará alojado en uno de los servidores físicos de la Facultad de Informática de la Universidad de Oviedo.

El sistema estará accesible desde cualquier lugar a través de una dirección IP pública dentro de la Red de la Universidad de Oviedo.

Se dispondrá de un único servidor físico en el que se alojarán todas las máquinas virtuales que formarán parte del proyecto.

El servidor físico sobre el cual se montará toda la infraestructura, estará gobernado por Xen.

El sistema soportará el acceso simultáneo de varios usuarios a la vez.

Multiplataforma (interfaz web, accesible desde cualquier dispositivo con un navegador capaz)

Desarrollo de este proyecto

Principio del Menor Privilegio

Durante el desarrollo de este proyecto se ha intentado seguir el Principio del Menor Privilegio Posible también conocido por Principle of Least Privilege (término originalmente en inglés). Aplicar este principio a un proyecto como éste significa que todas las configuraciones realizadas han intentado cubrir única y exclusivamente, aquello para lo que inicialmente han sido inicialmente pensadas y nada más. Es decir, a la hora de configurar reglas de acceso, permisos, grupos, usuarios, etc se ha intentado proveer únicamente de los permisos mínimos y necesarios para desempeñar cada una de las tareas para las que

han sido diseñadas desde un primer momento. A pesar de que la aplicación de esta práctica pueda parecer obvia y de sentido común, la realidad nos proporciona innumerables ejemplos de exactamente lo contrario y es por ello por lo que quiero hacer hincapié en esto. Configuraciones donde por error u omisión, mala práctica o simplemente por desconocimiento, dejan al expuesto al público más de lo que inicialmente debería. Por tanto una configuración de seguridad basada en mínimos siempre nos proporcionará mayor seguridad en caso de que nuestro sistema se vea comprometido.

Un ejemplo práctico de la aplicación de este principio en este proyecto lo tenemos en el caso de los accesos. Para las conexiones remotas a los equipos GNU/Linux, fundamentales debido a la idiosincrasia del proyecto, se ha optado por accesos cifrados mediante el protocolo SSH con clave pública/privada.

También se ha aplicado en el caso de los *firewalls* o cortafuegos, donde cada una de las excepciones, necesarias para el correcto funcionamiento de algunos servicios, han sido estudiadas y reducido su impacto únicamente el uso desde ciertas máquinas controladas de nuestro entorno.

Este es sólo un ejemplo de como ha sido aplicado este principio. Aunque en el uso diario, esto pueda parecer una práctica incómoda para el administrador, su no aplicación genera configuraciones adicionales y molestias para el usuario, a la larga está demostrado que genera sistemas software mucho más consistentes, seguros y con una menor exposición a ataques, errores y fallos.

Una vez finalizada la configuración del sistema el acceso mediante usuario root o superusuario se ha intentado evitar y se han configurado usuarios con permisos más limitados sobre el sistema.

Desarrollos Ágiles (Agile Development)

El desarrollo de este proyecto ha intentado seguir algunos de los principios de los desarrollos ágiles.

Que es como funciona, algunos métodos de agile development que se usan. SCRUM

Desarrollo de esta documentación

El ciclo de desarrollo de esta documentación también ha intentado seguir algunos los principios del *Agile Development*. con el objetivo principal de evitar el conocido -y temido- síndrome de la hoja en blanco. Durante el proceso de documentación se ha aplicado un modelo adaptado de revisiones -o *sprints*- propio de los modelos de desarrollo ágil como SCRUM.

Para llevarlo a cabo en cada uno de los apartados de este documento, en un primer momento, justo antes de ponerse a escribir lo único que se tendrá claro es el tema del que va a tratar el párrafo pero no como se estructurará o se detallará la información. Ese paso de reestructuración y revisión vendrá después, en una de las futuras revisiones.

De esta forma en un primera fase se comienza a escribir prácticamente de una forma anárquica, sin cuidar las formas o el contenido del párrafo más allá de ceñirse al tema en

concreto. Lo que se trata es volcar toda la información que se tenga sobre un tema determinado de la forma más pródiga posible. Así mismo y a medida que se escribe irán apareciendo nuevos temas o apartados considerados como interesantes o incluso cabe la posibilidad de subdividir los apartados si así se considera. En ese caso bastará con escribir el título del párrafo y continuar con lo que se estaba escribiendo anteriormente y así con todos los temas y apartados que vayan surgiendo.

En una segunda fase, o ciclo, se releerá toda esa información anteriormente escrita, se corregirán frases, errores ortográficos y faltas ortográficas y se irá añadiendo información de una forma más refinada. En esta segunda lectura podrán aparecer nuevos subapartados, reorganizaciones del texto o incluso el borrado de información por considerarla poco atractiva o interesante.

Una vez finalizada esta fase, el trabajo no termina ahí sino que se producirá de nuevo otra iteración en la que se repetirán los pasos descritos en el segundo punto. Este proceso se repetirá hasta llegar a la versión final.

Diagrama de sprints adaptado a documentación.

Gracias a este modelo de desarrollo, la documentación se produce prácticamente en tiempo real, evitando así el dejarla para el último momento. Cada nuevo cambio en el proyecto es reflejado en la documentación sin importar su formato. Poco a poco la documentación irá cogiendo la forma deseada a medida que las distintas iteraciones vayan pasando.

Como nota adicional, añadir que el contenido íntegro de este documento ha sido desarrollado siguiendo este proceso, incluido este mismo apartado.

OpenSource en este proyecto

Otro de los retos accesorios que me presentaba este proyecto era la posibilidad de realizarlo íntegramente con herramientas y programas de código abierto. Personalmente, desde hace ya algún tiempo, tenía ganas de acometer una tarea como esta utilizando sólo herramientas licenciadas como software libre y poder demostrar – y de paso demostrarme a mí mismo también, porque no- que éstas pueden ofrecer una calidad y resultados similares al que proporcionan sus homólogas privativas.

Una de las razones principales que me ha impulsado a embarcarme en este reto es el hecho de que desde hace ya algunos años he ido mudándome paulatinamente a alternativas *opensource*.

Este cambio, comenzó con pequeños detalles en mi día a día informático como puede ser cambiando el navegador web del sistema o el reproductor de vídeo/música predeterminado. Aunque a simple vista no pueda parecerlo, este proceso de migración, lleva bastante trabajo, auto-formación, así como búsqueda de alternativas a las herramientas ya conocidas. Aunque después de todo considero que he conseguido realizarlo con bastante éxito y en el día a día no hay tarea que no pueda realizar usando software libre de las que realizaba desde su equivalente en funcionalidad cerrado.

Por tanto intentaré seguir con esta filosofía de apostar por desarrollos y herramientas libres en el desarrollo de este proyecto y así verificar si es posible alcanzar unos objetivos y resultados que cumplan unos requisitos de calidad similares a los que podríamos conseguir con herramientas cerradas. La teoría nos dice que sí que es posible, pero muchas veces llevar esto a la practica no resulta tan sencillo.

A pesar de que hoy en día podríamos decir que los productos opensource están cada vez más en boga y existe una amplia variedad de herramientas que nos proporcionan alternativas de gran calidad y fiabilidad, en numerosas ocasiones su homólogo cerrado ofrece mayor sencillez de uso, fiabilidad ante errores, soporte y calidad del producto generado.

Por tanto, siendo inicialmente consciente de esta limitación, durante el desarrollo de el proyecto se intentarán utilizar en la medida de lo posible herramientas gratuitas, opensource y de código abierto.

Así mismo se mantendrá una pequeña máquina virtual con un sistema Microsoft Windows de escritorio para realizar algunas de las configuraciones iniciales, que por circunstancias ajenas a este proyecto, sólo se podrán realizar desde Windows. Esta máquina me servirá de apoyo durante todo el desarrollo y me brindará, en el caso de ser necesitada, la oportunidad de utilizar alguna herramienta sólo disponible en entornos Microsoft Windows.

Está documentación, también seguirá la premisa anteriormente explicada y como prueba de ello está siendo escrita íntegramente utilizando la suite ofimática LibreOffice. Las fuentes utilizadas para la realización de textos también serán opensource.

Una vez explicado lo anterior, me gustaría remarcar que en ningún momento el cumplimiento de esta premisa se convertirá en un dogma inquebrantable. Es decir, el uso de software y herramientas libres es un -deseable- posible camino y no un fin en sí mismo, por lo que en el caso de que no poder cumplir con el principal objetivo del proyecto, romperlo no supondrá un problema. En cualquier caso quiero dejar claro que en ningún momento se comprometerá la calidad final del trabajo o los resultados por mantenerlo vigente aunque sí haré un esfuerzo adicional para intentar mantenerlo.

Licencia y disponibilidad de este proyecto.

Una vez finalizado este proyecto toda la documentación, ficheros de configuración, scripts utilizados, lista de software disponible, así como las máquinas virtuales generadas durante este proyecto, se mantendrán en un repositorio público para su descarga, revisión y modificación por parte de la comunidad de usuarios.

Aplicaciones y herramientas clave utilizadas en este proyecto

Además una distribución GNU/Linux con OpenStack y Microsoft Windows Server con Directorio Activo, para el desempeño de este proyecto se han utilizado una serie de aplicaciones/herramientas que pasaré a introducir a continuación de forma breve:

VirtualBox

Es un Hypervisor de tipo dos gratuito y de código abierto desarrollado originalmente por la desaparecida Sun Microsystems. Tras la compra de Sun por parte de Oracle a principios de 2010 (nota <http://www.oracle.com/us/sun/index.html>), el gigante de americano de las bases de datos ha sido el encargado de mantener el desarrollo de VirtualBox. Sencillo, intuitivo y multiplataforma es una buena alternativa gratuita y opensource a la presentadas por VMware o Parallels entre otras.

Durante este proyecto VirtualBox ha sido utilizado para mantener una modesta máquina virtual con una instalación de Microsoft Windows de escritorio en la que poder realizar todas aquellas configuraciones que requerían inevitablemente el uso de herramientas Windows.

Libreoffice

LibreOffice es la suite ofimática de código abierto por antonomasia. Desarrollada inicialmente como OpenOffice bajo el paraguas de Sun Microsystems, tras la compra de esta última por parte de Oracle, se realizó un fork del proyecto que fue bautizado como LibreOffice. Prácticamente condenada al olvido sin una compañía que lo sustente, contra todo pronóstico, Libreoffice ha mejorado enormemente durante sus últimas versiones y la comunidad de desarrolladores del proyecto está haciendo un trabajo encomiable.

Aún existe una versión de OpenOffice mantenida por la Apache Foundation con un funcionamiento óptimo, aunque si bien es cierto que la gran mayoría de las distribuciones GNU/Linux han optado por migrar a LibreOffice.

LibreOffice ha sido utilizado para escribir toda la documentación de este proyecto.

Firefox

Es difícil que Mozilla Firefox necesite presentación hoy en día. Firefox, es uno de los navegadores web mas famosos del mundo. Desarrollado por Mozilla y basado en el mítico NetScape. De código abierto y multiplataforma, en la actualizad solo se ve superador en numero de usuarios por el navegador web de Google, Chrome.

El navegador del panda rojo, es ampliamente utilizado en sistemas Windows, Machintosh y GNU/Linux cumple con los últimos estándares referentes a

A lo largo de este desarrollo fue utilizado para todo lo referente a basqueadas, configuraciones y acceso a través de la web UI de los distintos servicios implementados.

Gedit

Gedit es el editor de texto por defecto del entorno de escritorio de gnome. Sencillo, rápido y con características tan útiles como sintaxis resaltada, búsqueda y reemplazo y pestañas.

Vim

Una de los editores de texto en modo consola más famosos. Tremendamente potente y configurable. Para los usuarios más novatos puede parecer algo complicado en su manejo, pero con algo de tiempo de uso es sencillo llegar a entender el porqué en es uno de los editores de consola para el mundo Linux que mas alabanzas y usuarios recoge.

Git

Draw.io

Draw.io es un editor de diagramas en línea completo y muy sencillo de usar. Está disponible a través de la dirección con su propio nombre <http://www.draw.io> y para su utilización no es necesario ningún tipo de registro o instalación.

Además Draw.io ofrece la opción de ser enlazado con diversos proveedores de almacenamiento, como Dropbox o Google Drive para poder guardar los diagramas automáticamente en dichas localizaciones. También dispone de integración con herramientas como Confluence y JIRA. Todo diagrama generado en draw.io es un fichero XML por lo que favorece la interoperabilidad entre plataformas y su exportación.

Todos los diagramas presentes en esta documentación han sido generados utilizando dicha herramienta. A pesar de que su uso es completamente gratuito para el usuario final, su licencia no es de código abierto.

Hypervisores que son para que sirven distintos tipos 1 y dos
KVM

esxi

Xen

Hyper-V

Xen Server 6.2

origen y desarrollador

empresas que lo usan

partners

Xen Center

(info oficial <http://www.xenserver.org/partners/developing-products-for-xenserver/21-xencenter-development/88-xc-dev-home.html>)

Xen Center es una aplicación exclusiva para sistemas Microsoft Windows, que proporciona una interfaz gráfica de usuario para administrar remotamente servidores instalados mediante el hipervisor XenServer. Gracias a esta herramienta, es posible realizar la mayoría de sus configuraciones remotas de una forma más amigable e intuitiva, de lo que se podría

hacer utilizando una ventana de terminal de comandos. XenCenter es posiblemente una de las mejores opciones iniciales para un usuario no experimentado. Desde mediados del año 2013, XenServer es completamente opensource y su código está licenciado como BSD 2-Clause. XenCenter está desarrollado utilizando el lenguaje de programación C#.

Actualizaciones del programa

A día de hoy, a pesar de ser considerado como un proyecto maduro y ya no recibir grandes cambios en su código, XenServer sigue actualizándose periódicamente. Estas actualizaciones, arreglan fallos conocidos aumentan la estabilidad y fiabilidad.

Puedes echarle un vistazo al código del proyecto es incluso hacer una copia del mismo desde su página en github: <https://github.com/xenserver/xenadmin>

Xen Orchestra

Xen Orchestra es una herramienta que nos proporciona una completa y moderna interfaz Web para administrar y configurar XenServer. Xen Orchestra es una aplicación *opensource* desarrollada por [Vates](#), una empresa francesa especializada en el desarrollo de productos de código abierto. Una de las ventajas que ofrece frente a otras opciones de funcionalidad similar, es que al tratarse de una aplicación web es posible acceder a ella desde cualquier navegador web moderno, independientemente del Sistema Operativo instalado. Para conectarse, bastará con una conexión a Internet y un navegador capaz de interpretar código HTML5 y *javascript*, que son los principales lenguajes utilizados en el desarrollo del Xen Orchestra. Orchestra nos proporciona una interfaz limpia y amigable para administrar nuestro servidor Xen de forma remota.

Desde la página web del proyecto es posible descargarse una imagen Xen en formato xva, preparada para ser importada en nuestro entorno Xenserver. Esta imagen que distribuyen directamente los desarrolladores de la aplicación, no es más que una Debian Wheezy 7 con los servicios de Xen Orchestra corriendo en el puerto 80. Una vez desplegada la imagen en nuestro servidor sólo tendremos que abrir un navegador e introducir la IP de la máquina XOA.

Al tratarse de un desarrollo joven y *opensource* el proyecto tiene un ritmo de actualizaciones bastante elevado, incorporando mejoras, nuevas características y correcciones muy rápido.

Como la misión de XenOrchestra es la de brindarnos un frontend web de configuración para nuestro XenServer, y esto no afecta de manera notoria a la parte intrínseca del proyecto, se ha decidido incorporar las actualizaciones al proyecto a medida que estas fueran siendo liberadas a pesar de que el producto ya estaba siendo utilizado en "producción". Al no disponer de un entorno de desarrollo a pruebas definido como tal, como medida de cautela adicional, cada nueva actualización ha sido probada en una máquina virtual antes de ser instalada. Una vez que la actualización en el entorno de pruebas, es considerada como satisfactoria entonces se procedía con la instalación en el servidor.

La primera versión utilizada en este proyecto en lo que podría considerarse como

“producción” fue la 3.1 liberada el 14 de febrero de 2014. A lo largo del todo el desarrollo han salido nuevas versiones que han sido incorporadas satisfactoriamente a nuestro sistema en producción.

Estado del arte

Introducción a la nube, nube vs modelo tradicional.

OpenStack

Eucalyptus

CloudStack

VmWARE vCloud, sólo nube privada.

Openstack

OpenNebula

Amazon AWS EC2

Google Compute Engine

Microsoft Azure

Joyent

Configuraciones LDAP no Windows. OpenLDAP etc. ApacheDS

OpenDJ...

¿Qué se pretende?

Este proyecto cubrirá la documentación e instalación desde cero de la infraestructura necesaria para montar un sistema de nube propia en el que los usuarios estarán gestionados desde una herramienta ajena al sistema. Para poder utilizar este sistema se configurará un sistema de autenticación basado en la implementación de Microsoft del protocolo LDAP llamado Active Directory. Los usuarios, grupos y permisos serán configurados en Windows Server y se implementará una integración con el sistema de autenticación de la nube desplegada, para que dichos usuarios de estos puedan acceder y utilizar sus recursos.

¿que es la nube?

La “nube” es el término coloquial que se le suele dar al modelo de entrega de recursos informáticos, aplicaciones y contenido software bajo demanda a través de Internet. Desde un punto de vista no técnico podríamos considerarla como un conjunto de recursos disponibles en alguna parte de la que desconocemos su ubicación, pero de los cuales podemos utilizar aquello que necesitamos, en el preciso momento que lo necesitamos. Además una vez que dejemos de utilizar dicho recurso, el sistema de “nube” nos permitirá devolverlo para que otros clientes del servicio puedan reutilizarlo a su vez.

Para entender este concepto de forma más clara, pongamos un ejemplo sencillo con el que posiblemente todos nosotros estemos familiarizados. A día de hoy prácticamente todos los usuarios de Internet disponen de una cuenta de correo personal con la que poder recibir y enviar correos electrónicos. Cada una de estas cuentas, dispone de un espacio asignado en

la nube para todos nuestros archivos relacionados con el correo: bandeja de entrada, mensajes enviados, borradores, etc. En este caso, cuando recibimos un correo que contiene algún fichero adjunto, como fotografías, vídeo o incluso documentos de texto, podemos acceder a ese correo desde diversas localizaciones y seguir teniendo acceso al contenido del mensaje y sus archivos adjuntos. No importa si ingresamos desde nuestro teléfono, tableta, ordenador personal o desde casa de un amigo, siempre y cuando entremos a nuestra cuenta de correo, será posible acceder a los archivos. Pero entonces, ¿donde es que se almacenan estos archivos? ¿En nuestros dispositivos? ¿En el propio navegador? No. La respuesta es: en 'la nube'. Nuestros archivos se encuentran alojados en los servidores físicos de nuestro proveedor de servicio, esperando a ser accedidos por los usuarios. De esta forma no ocupan lugar en nuestros dispositivos y estarán siempre accesibles desde cualquier dispositivo siempre y cuando dispongamos de una conexión a Internet con la que poder conectarnos.

Una vez entendido el ejemplo anterior, amplíemos este concepto no sólo a almacenamiento si no también a capacidad de cálculo o memoria RAM de la que disponemos. Es decir, podríamos adquirir cierta capacidad de proceso de manera puntual para desempeñar un trabajo concreto y una vez finalizado, simplemente devolverlo. Lo más importante sin preocuparnos de tiempos de envío, retrasos, montajes, instalación...etc, los recursos de la nube están siempre disponibles y a disposición de los usuarios para ser utilizados.

Esto genera un modelo de negocio donde hay empresas que se dedican a ceder su infraestructura de en la nube a organizaciones, otras empresas o particulares por periodos limitados de tiempo.

Pros y contras del modelo de nube.

Como todo, el modelo en la nube tiene aspectos positivos y otros negativos. En este apartado veremos algunos de ellos.

Es difícil negar que la nube nos proporciona algunas características que nos hacen la vida mas cómoda y fácil. Algunos ejemplos como el simple hecho de poder acceder a nuestros datos desde cualquier dispositivo con una conexión a Internet es posiblemente el más destacado de todos ellos. Gracias a la nube, esa tediosa tarea de compartir contenido en formato físico entre tus conocidos en CD/DVS o lápices USB ha terminado. Hoy en día bastará con guardar dichos datos en tu cuenta en la nube, enviar un enlace al interesado e inmediatamente ya dispondrá acceso a los archivos para descargarlos o simplemente revisarlos en línea. Así de fácil. Además, debido a que nuestros datos no se encuentran almacenados directamente en nuestros dispositivos, podemos dedicar ese espacio sobrante para otras cosas. Es más, dependiendo de nuestro proveedor de servicios de nube es muy posible que incluso se realice una copia de seguridad automática de nuestros datos, por lo que además podemos despreocuparnos totalmente de tener que realizar duplicados, etiquetarlos y almacenarlos...

Por último, el precio, con el paso de los años el precio de los servicios de nube están haciéndose cada más y más populares y en consecuencia los precios bajan haciendo aún más atractivos si cabe este tipo de servicios.

Como no todo podía ser perfecto, este modelo también conlleva una serie de contras que debemos tener en cuenta antes de lanzarnos a la nube. En primer lugar hemos de ser conscientes de que existe un pérdida del control sobre nuestros datos. Resulta bastante obvio que guardando nuestra vida digital en la nube, perdemos conocimiento de donde se encuentran, quien los tiene alojados o incluso de quien tiene acceso a ellos. Esto se hace más acuciante con los escándalos de los últimos tiempos relacionados con espionaje y filtraciones. Parece que todo lo almacenado en la nube es propenso a ser espiado por los servicios de inteligencia de algunos países alegando motivos de seguridad. Esta preocupación surge a raíz de la restricción del propio modelo de tener que confiar en una tercera parte para almacenar tus datos. En este nuevo modelo, ya no tenemos completo control sobre dónde y cómo se almacenará nuestros datos. También la necesidad de estar siempre conectados a Internet puede ser un problema en determinados entornos. Siempre y cuando dispongamos de una conexión estable y con velocidades decentes no habrá problema, pero en caso contrario, acceder a nuestro datos pueden convertirse en una tarea tediosa y en el peor de los casos, imposible.

Acceso remoto al servidor físico (xenserver05)

Para llevar a cabo este proyecto la Facultad de Ingeniería Informática de Oviedo me cedió uno de sus servidores en el que poder implementar, desplegar así como probar mi entorno de desarrollo al completo.

Foto del servidor

El grupo de Cloud Computing me proporcionó acceso a un servidor físico alojado en uno de los racks de la sala de máquinas de la Facultad. El servidor se corresponde a un modelo Dell PowerEdge R710 con 2Tb espacio en disco y 128Gb de RAM. Desde el grupo de Cloud me también me habilitaron las credenciales del usuario administrador 'root' así como acceso remoto SSH para su configuración a distancia.

Este servidor, está gobernado por XenServer en su versión 6.2. Xen es una distribución GNU/Linux derivada de RedHat Enterprise Linux (RHEL) y es sobre la cual se sustenta toda la infraestructura necesaria para el funcionamiento de este proyecto. Tanto la instalación de XenServer 6.2 en el servidor físico, como la configuración de acceso de mi usuario por SSH, fue realizada directamente por los miembros de Cloud Computing de la Escuela.

A continuación una lista a modo de resumen de la configuración del servidor:

Nombre de la máquina: **xenserver-05**

Hardware

Fabricante: Dell Inc.

Modelo: de servidor: PowerEdge R710

Información de BIOS:

- Fabricante: Dell Inc
- Versión: 3.0.0

Procesador:

- Modelo: Intel(R) Xeon(R) CPU E5504

- Velocidad de reloj: 2.00GHz
- CPUs físicas: 2
- CPUs lógicas : 8
- CPUs Sockets: 2

Memoria:

- Disponible : 122880 MB
- Sockets de memoria: 18

Controladores de almacenamiento

- Interfaz IDE: Dell PowerEdge R710 SATA IDE
- Interfaz RAID: Controlador de BUS RAID: Dell PERC 6/i Integrated RAID

Interfaces de Red disponibles

- Interfaz de red principal:
 - Nombre: eth0
 - MAC: 00:26:b9:85:5a:26
 - Velocidad: Gigabit
- Interfaz de red secundarias:
 - Nombre: eth1
 - MAC: 00:26:b9:85:5a:28
 - Velocidad: Gigabit
 - Nombre eth2
 - MAC: 00:26:b9:85:5a:2a
 - Nombre: eth3
 - MAC: 00:26:b9:85:5a:26
 - Velocidad: Gigabit

Configuración de red

- MAC: 00:26:b9:85:5a:26
- IP: 156.35.94.186
- Máscara: 255.255.254.0
- Puerta de enlace: 156.35.94.201

Sistema Operativo:

- XenServer versión 6.2.0-70446c (xenenterprise)

Así mismo el grupo de Cloud de la Facultad de Informática me proporcionó un rango de direcciones IP con salida directa a Internet (sin proxys ni saltos intermedios) con las que poder configurar las maquinas alojadas sobre el xenserver. Estas IPs estarían dentro del rango 156.35.95.20-156.35.95.24 con mascara de red de 24 bits.

Lista de IPs disponibles:

- 156.35.95.20, 156.35.95.21, 156.35.95.22, 156.35.95.23, 156.35.95.24

Mascara de red:

- 255.255.255.0

Montaje de la infraestructura necesaria

A continuación veremos como fue instalado el sistema sobre el que montaría este proyecto, así como la descripción de algunos de problemas que he ido encontrando mientras realizaba el proceso. En uno de los capítulos finales de esta documentación, se dedica un apartado completo a repasar algunos de los problemas encontrados considerados como más importantes.

Comenzando

El primer problema importante que tuve que afrontar fue la instalación remota del entorno de trabajo sobre el que se desplegaría todo proyecto.

La teoría dice que a día de hoy y gracias a la extraordinaria ubicuidad que nos proporciona Internet, no disponer de acceso físico a una máquina no supone un problema serio. Esto es así en *teoría* y remarco lo de 'en teoría' porque como veremos a continuación, durante el desarrollo de este proyecto me encontré con algunas dificultades, fruto de la falta de acceso físico a la máquina y mi inexperiencia con Xen y entornos de virtualización (Hipervisores).

¿Qué se pretende realizar?

Esta es posiblemente la pregunta más importante que uno debe realizarse antes de iniciar un proyecto como este. No se trata de ponerse a desplegar máquinas sin ton ni son, duplicando funcionalidades, configuraciones y muy probablemente, futuros dolores de cabeza...

En primer lugar antes de meterse manos a la obra es necesario realizar un análisis de qué es lo que se pretende montar, objetivos que se pretende alcanzar y qué necesitamos para llevarlos a cabo: herramientas necesarias, requisitos funcionales, requisitos hardware...

Este trabajo está titulado "**Integración de OpenStack con un Active Directory**" por lo que resulta bastante obvio que como mínimo, el proyecto consistirá en al menos dos instancias bien diferenciadas.

La primera de ellas será un Windows Server que adoptará el rol de Directorio Activo y la segunda se corresponderá con una máquina con los componentes básicos de OpenStack desplegados y configurados para su correcto funcionamiento.

A continuación veremos la justificación de algunas de las decisiones tomadas.

Elección de Windows

En el caso de la decisión en la parte Windows no se presentaban muchas dudas mas allá de la elección del tipo de licencia.

Desde un primer momento se tuvo muy claro que utilizaríamos Windows en su versión para servidores, llamada comercialmente Microsoft Windows Server. En relación a esto, en el momento de escribir líneas, la última versión disponible en el mercado de este producto se trata de Windows Server 2012 R2. Desafortunadamente, para instalarlo disponía de una licencia de estudiante que sólo cubría la instalación de la versión anterior, Windows Server 2012. Microsoft siempre añade nuevas funcionalidades y configuraciones en cada una de

sus versiones, por lo que a veces, utilizar una versión antigua puede suponer quebraderos de cabeza adicionales. De todas formas, a pesar de no ser la última versión disponible en el mercado, para el cometido de este proyecto (rol de Active Directory) cualquiera de las versiones actuales de Windows Server cumple de forma más que suficiente.

Desde hace ya varias décadas, los productos de Microsoft destinados a servidores han mejorado notablemente, afianzándose con el paso de los años como solución más extendida entre las pequeñas y medianas empresas (PYMES) e incluso entre las más grandes. La integración con las otras herramientas propias de la empresa, su sistema de formación y certificaciones, el amplio soporte de los fabricantes hardware y comunicaciones, posicionan a Microsoft Windows Server como una de las opciones más consolidadas para su uso empresarial, así como en proyectos de prácticamente cualquier tipo. Por todo esto, considero que conocer y estar familiarizado algunas de las configuraciones básicas así como conceptos de administración relacionados con sistemas Windows es algo muy importante que complementa y enriquece el perfil de un profesional dedicado a la informática.

Licencias.

En cuanto al tipo de licencia elegida, la compañía de Redmon suelen optar por crear distintas licencias para un mismo producto. Dependiendo de la licencia elegida unas u otras opciones estarán disponibles. A juicio de Microsoft, este modelo hace sus productos más atractivos de cara a los distintos perfiles de cliente final. Ellos mismos son conscientes de lo complicado que puede llegar a ser entender los diferentes tipos de licencia y dejan a disposición de los usuarios un documento con la descripción detallada de cada una de las versiones (contenido PDF anexo, disclaimer logos imágenes, pertenecen a sus respectivos autores)

Por razones obvias relacionadas con la funcionalidad final del proyecto, en este caso sólo me centraré en aquellas versiones que ofrecen como mínimo la característica de Directorio Activo. El resto de ellas, aunque igualmente válidas para otros usos, no cumplen el cometido mínimo de este proyecto por lo que no serán tenidas en cuenta.

De todas las distintas versiones ofrecidas en el catalogo de Microsoft evaluaré únicamente las siguientes:

- Windows Server 2012 Datacenter
- Windows Server 2012 Standard
- Windows Server 2012 Essentials
- Windows Server 2012 Foundation

	Versiones de Windows Server 2012			
	Datacenter	Standard	Essentials	Foundation
Número de usuario				

s mximos				
Maxim o número de conexio nes SMB				

Siendo realista y teniendo siempre presente el alcance final del proyecto, todas las versiones anteriormente descritas nos proporcionan una experiencia parecida por su similitud de características ofrecidas, por lo que es más que probable que todas cumplan con creces los requisitos de este proyecto.

Además como ya había comentado de forma breve anteriormente, el precio de la licencia, muchas veces determinante, no supone un problema puesto que en este proyecto se utilizará las licencias gratuitas de estudiante proporcionadas por DreamSpark.

Teniendo en cuenta todo lo anteriormente comentado, mi decisión final se inclinó por Windows Server 2012 en su versión Standard, simplemente por cuestión de simplicidad. La versión Standard incluye prácticamente todas las funcionalidades que se requieran hoy en día en un servidor de uso general. En un entorno de producción real, con costes reales, quizá podría haberse optado por una licencia no tan genérica con el objetivo de reducir el gasto del presupuesto en licencias del proyecto. Uno de los anexo de este documento dejo información adicional sobre licencias, así como precios oficiales de las licencias de Microsoft Server.

DreamSpark

DreamSpark (https://www.dreamspark.com/) es una iniciativa surgida en el seno de Microsoft, que tiene como objetivo dotar con licencias software a estudiantes para que así puedan utilizar gratuita y legalmente algunos de sus productos. El programa fue anunciado públicamente por Bill Gates en el año 2008. Desde su creación la lista de instituciones y países elegibles no ha parado de crecer. Para beneficiarse del programa sólo será necesario darse de alta en el mismo utilizando una cuenta de correo de una institución educadora válida.

Ésta es una gran iniciativa por parte de Microsoft que proporciona a estudiantes de todo el mundo la oportunidad de acceder a software al que normalmente no tendría acceso debido al elevado coste de licencias.

La lista de software ofertado es muy amplia y es posible encontrar desde sistemas

operativos de escritorio, servidores, entornos de desarrollo, software de virtualización, etc.

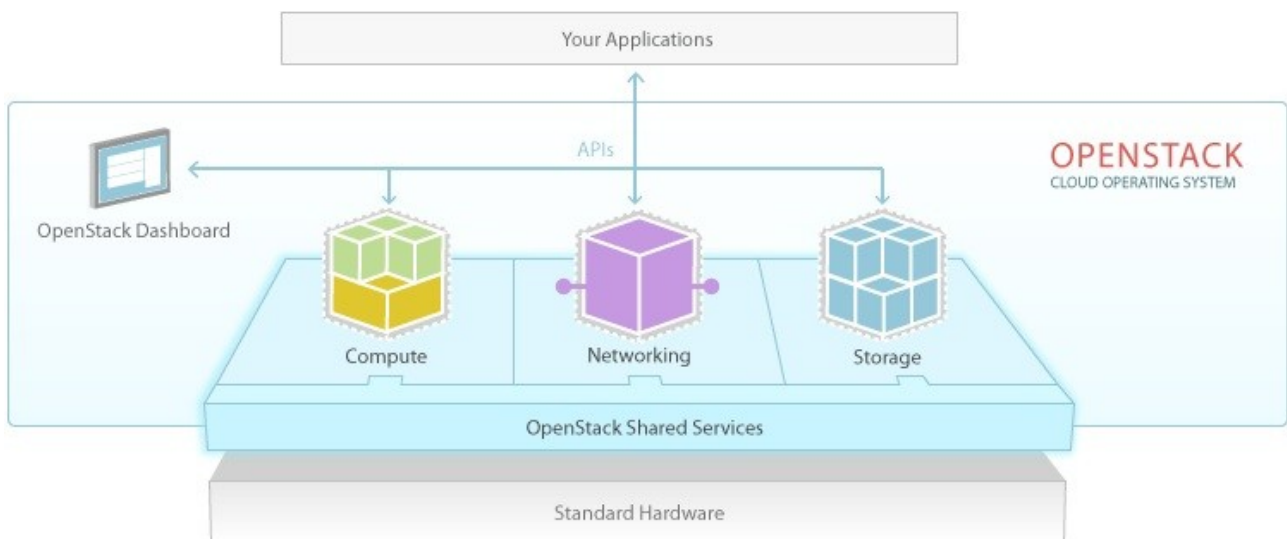
Todas las herramientas de Microsoft que requerían licencia comercial utilizadas en este proyecto, han sido activadas utilizando el programa DreamSpark.

Elección de OpenStack, sistema de computación en la nube

Como en el caso anterior relacionado con Windows Server, la elección de OpenStack como el sistema de nube a utilizar en este proyecto tampoco fue demasiado compleja. En la actualidad existen en el mercado varias alternativas aunque desde un primer momento estaba inclinado hacia aprender y enfrentarme por mi mismo al que es considerado por muchos uno de los desarrollos de software libre más prometedores del mercado en la actualidad, OpenStack.

En la sección de Estado de Arte, vimos las características y pormenores de las distintas soluciones disponibles en el mercado actualmente. OpenStack es un desarrollo joven con una comunidad de usuarios y desarrolladores en expansión y que ya está siendo utilizado en grandes organizaciones como HP, Yahoo!, Ebay, el CERN o la NASA, entre otras.

OpenStack es instalado como una capa software adicional sobre el sistema operativo en la que se despliegan sus distintos módulos y servicios.



Por tanto para poder instalar OpenStack, previamente debemos elegir sobre qué sistema operativo lo desplegaremos.

En este apartado veremos cuales han sido las distintas opciones barajadas así como la escogida finalmente, dado que en contraposición al caso de Windows sí que ha habido varios firmes candidatos.

Antes de comenzar con este apartado me gustaría remarcar que prácticamente todas las opciones aquí barajadas podrían haber sido completamente válidas, si bien es cierto que la experiencia OpenStack que nos hubiera brindado cada una de ellas diferiría en algunos

aspectos de las de las otras.

Elección de la distribución GNU/Linux

Ubuntu 14.04 LTS

Ubuntu es posiblemente una de las tres distribuciones GNU/Linux más conocidas y comentadas a nivel mundial. Basada en la robusta Debian esta distribución fue creada en el año 2004 por Canonical, una empresa con capital íntegramente privado que es la que se encarga al completo de su desarrollo y mantenimiento. Además, debido a al apoyo por parte de Canonical, Ubuntu es una de las distribuciones soportadas oficialmente por el proyecto OpenStack.

Gracias a su sistema de repositorios (llamados PPAs) es fácil disponer de una versión actualizada y adaptada a nuestro sistema, sin requerir demasiado esfuerzo por parte del usuario final. Además Ubuntu dispone de una amplia documentación disponible en la red, así como de una gran comunidad de usuarios.

Otra característica a tener muy en cuenta es que Canonical brinda a disposición de los usuarios algunas versiones especiales denominadas LTS (o Long Term Support) con soporte completo oficial extendido.

Numerosos parches y mejoras han sido lanzadas desde su lanzamiento en abril 2014, haciendo de esta distribución una opción estable y madura.

Por otra parte, no menos importante, desde mediados de 2008 Ubuntu -en sus distintas versiones- ha sido la distribución que he utilizado en mi ordenador personal. Por lo que a pesar de que mi conocimiento sobre ella no va mas allá del simple uso en el día, sí que parto de una base que con otras distribuciones no tendría.

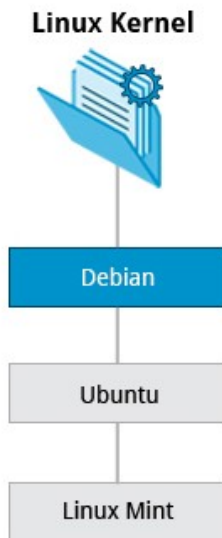
Otras opciones similares a las ofrecidas por Ubuntu:

Debian es la distribución 'madre' de muchas otras grandes conocidas del mundo GNU/Linux como la propia Ubuntu o Linux Mint. Esta distribución es ampliamente utilizada en servidores y ordenadores de uso general. El proyecto Debian es un proyecto completamente de código abierto centrado primordialmente en brindar al usuario una experiencia lo más estable y robusta posible. Es decir, apostando por Debian sin ningún tipo de añadidos o configuración de repositorios adicionales, sólo encontremos aquellas versiones de paquetes con una larga trayectoria de uso, pruebas y verificaciones. Este empeño hace que muchos de los paquetes que la conforman no estén muy actualizados y se encuentren disponibles en versiones antiguas.

Como en todo proyecto software la estabilidad del sistema base es algo muy importante, aun así en este caso no se trata de una opción por que nuestro objetivo es desplegar OpenStack con sus correspondientes dependencias. En el caso de hacerlo en Debian tendríamos que contar con repositorios denominados como 'testing' y actualizar una gran cantidad de paquetes del sistema debido las numerosas dependencias.

Además, a pesar de que Debian es una excelente distribución, no hay que olvidar que se trata de una versión comunitaria sin una empresa u organización detrás dándole soporte, por lo que quizá para un desarrollo como este no sea la mejor opción.

....



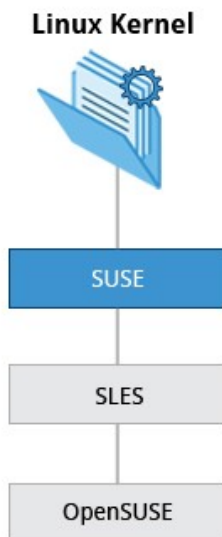
SuSe

Otra de las opciones aquí barajadas fue la proporcionada por la empresa alemana Novel con su sistema operativo SUSE y las diversas variantes de éste. Suse es la distribución madre y sobre ella está basada la versión empresarial denominada SUSE Linux Enterprise Server, llamada comúnmente, SLES.

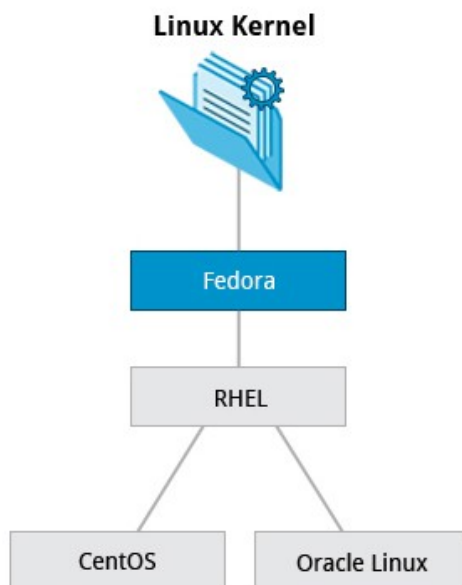
SLES es un SUSE con añadidos muy interesantes dirigidos al sector profesional y con el soporte y ayuda por parte de Novel. A su vez, partiendo de la base de SLES surge OpenSuSe, proporcionando todas las ventajas de SLES pero sin coste para el usuario final y obviamente sin el soporte empresarial de Novel.

Relacionado con OpenStack, en el año 2013 Novel lanzaba públicamente su versión comercial de la implementación de Openstack en Suse Linux bajo el nombre de SuSe Cloud versión 3.0 y basada en Openstack Havana. Con cada nueva versión de OpenStack novel lanza una nueva versión actualizándolo y dotándolo de las nuevas características. SuSe Cloud es un producto muy sólido que goza de unas excelentes críticas entre la comunidad, pero su elevado precio y su orientación hacia un perfil más profesional, la descartan auténticamente para el uso en este proyecto.

Aún así, esta solución es ideal para un entorno empresarial puro en el que se necesite desplegar y mantener un sistema en la nube de forma rápida y fiable a la vez que cuenta con un amplio soporte y experiencia por parte de la gente de SuSe.



RHEL y derivadas (Scientific Linux, CentOS y Fedora)



Posiblemente el rival mas duro respecto a Ubuntu Linux. Como en el caso de Ubuntu existe una ingente cantidad de documentación en línea así como una amplia comunidad de usuarios.

En las primeras fases del desarrollo de este proyecto, a principios de 2014 CentOS la edición comunitaria de RedHat Enterprise Linux (RHEL) fue integrada oficialmente bajo el paraguas protector de la empresa de sombrero rojo.

En cuanto a Fedora es una versión que llevo utilizado a diario en mi puesto de trabajo y en líneas generales estoy bastante satisfecho con su rendimiento y desempeño. Fedora es la rama de pruebas de RHEL e incorpora actualizaciones de paquetes y aplicaciones de una forma muy rápida. No en vano, Fedora se suele considerar la rama de pruebas de lo que serán las nuevas versiones de RHEL. Es por este motivo por la que no la he tenido finalmente en cuenta para este proyecto. Para el uso diario en un entorno de trabajo en un

ordenador de escritorio, Fedora cumple las expectativas del día a día, pero no es recomendable para un entorno en el que necesitemos la mayor tasa posible de fiabilidad y estabilidad.

En el caso de RHEL, cumple con todas las premisas en las que las otras fallaban. Es soportada por una gran empresa, RedHat, una gran comunidad de usuarios, orientada al sector profesional, actualizaciones, así como sus derivadas, además RHEL es una de las distribuciones soportadas oficialmente por el proyecto OpenStack... El único problema, es que su uso no es gratuito, por lo que para utilizar una versión en este desarrollo tendría que desembolsar el precio de la licencia (799 dolares americanos al año para una única instalación). A pesar de ser una fantástica candidata, debido a esto, esta versión fue descartada.

Otras - Scientific Linux.

Scientific Linux es una distribución orientada a la comunidad científica auspiciada por Fermilab y CERN. Está fuertemente basada en CentOS por lo que hereda todas sus grandes ventajas, más el buen hacer de Fermilab y CERN. Además su uso es completamente gratuito.

Debido a esto, Scientific Linux ha sido la versión Linux finalmente elegida sobre la que montar Openstack.

Vamos a dedicarle un apartado a esta distribución para entender de donde viene que es lo que busca así como sus principales objetivos.

Scientific Linux

Al tiempo de escribir estas líneas, la última versión disponible en el mercado está basada en CentOS 6.6. Prácticamente se trata de una CentOS ajustada para cumplir los altos estándares de calidad fiabilidad del CERN. Desarrollada por Fermilab y CERN nos proporciona un sistema estable, seguro y claramente destinado a un entorno de producción donde la estabilidad sea una de las bazas más importantes atener en cuenta. En su versión de escritorio, Scientific Linux utiliza el antiguo, probado y ampliamente configurable, Gnome 2.30.

Para que nos hagamos una idea de forma gráfica y bastante inmediata de lo comprometidos que están desde Scientific Linux con la estabilidad podemos echar un vistazo a algunos de sus componentes claves. Por ejemplo en el caso del kernel, SL utiliza una versión Long Term es decir una versión del núcleo con soporte extendido basado en el kernel 2.6.32. Durante el desarrollo de este proyecto se utilizó este: **2.6.32-504.1.3.el6.x86_64** lo que significa que tiene al menos **504** revisiones (correcciones, arreglos y mejoras) de diversa índole. Además del núcleo, prácticamente todos los paquetes siguen este principio por lo que nos puede dar una idea aproximada del grado de madurez del proyecto.

La parte negativa de esta filosofía, como nos pasaba en el caso de Debian, es que no encontraremos paquetes nuevos, por lo que si se trata de probar nuevas funcionalidades es posible que los oficialmente soportados no nos proporcione la mejor experiencia.

Instalación del entorno de trabajo

Primeras decisiones y configuraciones

Una vez decididos los sistemas operativos que se instalarán nos centraremos en definir aquellas necesidades hardware de cada una de las máquinas a desplegar. A pesar de que todo el sistema está montando sobre un entorno de virtualización y gracias a ello no supondría demasiada dificultad **aprovisionar** nuestras máquinas con nuevos recursos a posteriori. Aun así creo que puede ser muy provechoso el realizar este ejercicio de reflexión y dedicar un momento a pensar cuales son los requisitos hardware que debe cumplir cada una de las maquinas para desempeñar su actividad. Gracias a esto, procuraremos no vernos obligados a realizar modificaciones importantes en la configuración hardware de nuestras maquinas durante su ciclo de vida.

Recordemos el Hardware físico que tenemos disponible

Xenserver	
Procesador (núcleos)	8
RAM	128Gb
Disco	1000Gb

Ahora la configuración de las máquina virtuales

Windows Server

Requisitos mínimos (<http://technet.microsoft.com/es-es/library/jj134246.aspx>)

Procesador de 64 bits o compatible

512 RAM

32Gb disco

Unidad de DVD

Pantalla Super VGA (800 x 600) o de mayor resolución, Teclado y ratón de Microsoft® (u otro dispositivo apuntador compatible)

Acceso a Internet

Con toda esta información y sabiendo que el uso que se le dará a lo largo de proyecto será sólo el de alojar un Directorio Activo con usuarios y grupos de prueba se ha decidido dotarla con lo siguiente:

Windows Server 2012 Standard	
Procesador (núcleos)	1
RAM	16Gb
Disco	200Gb

Veamos ahora la configuración elegida frente a los requisitos mínimos.

. Minimos Asigandos Cometarios

Cumple todo.

Scientific Linux

En el caso de Scientific Linux al ser una versión basada en CentOS Linux los requisitos mínimos serán los mismos que para esta. En este caso los podemos encontrar aquí:

<http://wiki.centos.org/About/Product>

Tabla resumen

Scientific Linux 6.6	
Procesador (núcleos)	1
RAM	392Mb si usamos CLI / 512M entorno gráfico
Disco	2Gb

La información anterior es sólo aplicable si quisiéramos realizar una instalación de SCL6 y nada más. En este caso SLC6 será la base para nuestra configuración OpenStack tenemos que tener en cuenta los requisitos mínimos de este:

<http://openstack.redhat.com/Quickstart>

Por lo que en este caso de mínimo tendríamos:

Openstack	
Procesador	1 (aunque es imprescindible que este sea compatible con virtualización)
RAM	2Gb
Disco	5Gb

Como en el caso anterior, sabiendo esta información eligiéremos una configuración inicial capaz de desempeñar con éxito nuestras exigencias sin tener que modificarla en el corto plazo.

También hay que tener en cuenta que este sistema incluirá en sí mismo las nuevas instancias virtuales creadas en el futuro, por lo que debemos ser generosos en la configuración si queremos poder montar a su vez varias de ellas en nuestra entorno de nube.

2 procesadores

16 gb de ram

500GB de disco

Como en el caso de Windows compararemos con los requisitos mínimos:
Cumple todo.

Instalación

Configuraciones de RED

Como ya se comentó anteriormente en este documento, inicialmente el grupo de Cloud Computing de la Facultad de Informática de Oviedo proporcionó a este proyecto cinco IPs públicas que poder asignar a las máquinas virtuales, mediante la cual acceder posteriormente.

Debido a esto, mi intención inicial era la de repartir cada uno de los servicios en hasta cinco máquinas distintas aunque como se explica más adelante en el apartado de Problemas encontrados, tras varias pruebas con el sistema funcionando en producción, me vi obligado a desechar la idea de tener cada uno de los servicios dispersados en distintas máquinas como inicialmente había diseñado. Para solucionar el problema se optó por una solución mas pragmática de agrupar aquellos servicios accesorios al proyecto en una única máquina. De esta forma ahorramos IPs, esfuerzos y como se verá mas adelante, recursos, sobretodo núcleos de procesador, muy necesarios en este tipo de entornos.

Con todo la tabla de asignación de IPs queda de la siguiente forma.

Nombre	IP	Descripción
alexandria	156.35.95.20	Debian. Biblioteca NFS con las ISO de instalación del resto de sistemas.
Lesoleil (el Sol)	156.35.95.21	Windows Server 2012 con funciones de controlador de dominio y directorio activo.
Lenuage (La nube)	156.35.95.22	Scientific Linux 6.6 con RDO OpenStack
Lemiroir (el espejo)	156.35.95.20	Debian con ownCloud para alojar las copias de seguridad
Xoa	156.35.95.20	Debian con XenOrchestra para configurar y nuestras máquinas virtuales Xen.

Instalación de Windows Server

Cubrir el proceso de instalación. Este apartado necesita ser revisado y reordenado

Las instalaciones de Microsoft Windows suelen bastante sencillas y gracias a los asistentes en forma de diálogos que nos proporciona el sistema, cualquier persona con unos mínimos conocimientos de informática puede ser capaz completarlas con prácticamente éxito garantizado.

Es este caso, el principal problema residía en la dificultad que entraña instalar un Windows,

sistema operativo que carece de instalador en red, sin tener acceso físico a la máquina. En Internet pueden encontrarse imágenes no oficiales modificadas por usuarios con las que se puede llegar a suplir esta característica. Pero en mi caso no disponía de una de estas imágenes y descargarla de una fuente no verificada en Internet, no parecía la mejor opción. Para solucionar este inesperado problema se tuvieron en cuenta las siguientes alternativas.

Posibles soluciones barajadas y pros y contras:

La primera es posiblemente la más obvia, no se disponía de acceso físico, pero podía ponerme en contacto con gente que sí tenía y que podría ayudarme: los becarios de la Facultad de Informática.

Con su ayuda, podrían dejarme conectado un CD/DVD o un lápiz USB con la imagen de instalación. Remotamente podría conectarme y completar la instalación de manera más o menos normal. Posiblemente fuera la opción más directa y fácil para mí, pero por otro lado, implicaba movilizar a terceras personas y creaba cierta dependencia en el caso de encontrarme con una, más que probable, situación similar en el futuro.

Descartada la primera, la segunda opción que se me ocurrió fue la de preparar una imagen de Windows Server que admitiera instalación en red. Si bien no me terminaba de gustar la idea de utilizar una de las de las disponibles extra oficialmente en Internet, fabricar yo mismo la imagen no me producía tanto reparo. El problema de esto es que el Acuerdo de Licencia de Usuario Final (EULA por sus siglas en inglés) no permite ningún tipo de modificación de las imágenes de Windows, por lo que prácticamente antes de empezar ya estaría incurriendo en una violación de la licencia del producto. Además este tipo de modificaciones requieren a su vez de software especial, en muchos casos de pago. Tras investigar esta vía, decidí rechazarla por todas las razones comentadas anteriormente.

Almacén en Red

He de reconocer que esta opción era posiblemente la que menos me gustaba de primeras. La idea era sencilla sobre el papel, en algún lugar accesible montar una carpeta compartida en la que almacenar las imágenes ISO de los sistemas operativos. Esto funcionaría como especie de almacén en red desde el cual podría acceder y montar los archivos de instalación de los sistemas operativos. Sería como un pendrive usb de gran tamaño conectado por red con el servidor.

Tras leer la documentación de XenServer, soportaba dos tipos de almacén en red, SAMBA/CIFS y NFS. De cara al funcionamiento posterior, ambas opciones resultaban bastante parecidas y decantarse por una en favor de la otra respondía más que nada a una cuestión de preferencia personal. Debido a mi completa inexperiencia con este tipo de tecnologías, pregunté a los becarios de la Facultad de Informática, en busca de respuestas sobre como lo tenían ellos implementado. En su contestación, me comentaron que ellos utilizaban un NFS, así que finalmente me decidí por esta opción.

Biblioteca NFS ISO

Una vez decidido que utilizaría esta opción para almacenar las imágenes con las que comenzaría a instalar los sistemas operativos, tenía a su vez dos variantes:

1. Que los becarios me proporcionase acceso a sus servidores NFS ya correctamente configurados y funcionando.
2. Instalar mi propio servidor NFS, partiendo de un sistema operativo que sí tuviera instalador en red.

Opción 1: El problema de esta opción es que en esos servidores se guardaban licencias y otro software al que por un motivo u otro yo no podía tener acceso. En los correos electrónicos que intercambié tampoco parecían muy seguros de querer configurar un acceso especial a sólo aquellos productos software en los que estaba interesado. Así que finalmente tuve que descartar también esta opción.

Opción 2:

Al final y tras valorar los pros y contras de cada una de las opciones anteriormente detalladas se ha optado por a montar mi propio almacenamiento en red NFS en donde almacenar las imágenes ISO de los sistemas operativos.

Biblioteca NFS ISO

¿Qué es NFS?

NFS es un protocolo de sistema distribuido de ficheros desarrollado originariamente en 1984 por la desaparecida Sun Microsystems. NFS permite el acceso remoto de ficheros desde un ordenador cliente a un servidor a través de la red. A lo largo de los años el protocolo ha sufrido varias revisiones dotándolo de mayor seguridad y compatibilidad con un amplio número de sistemas operativos.

La versión actual es la 4.1 lanzada en enero del año 2010. La versión 4.2 está actualmente siendo desarrollada, aunque no se espera su lanzamiento en un futuro cercano.

(<https://tools.ietf.org/html/draft-ietf-nfsv4-minorversion2-29>).

¿Que sistemas operativos los soportan?

Actualmente NFS es soportado por la gran mayoría de los sistemas operativos modernos su uso está bastante extendido. En relación con este proyecto, se ha utilizado una NFS para almacenar imágenes ISO y así poder crear una biblioteca de imágenes con la que poder crear nuestras maquinas en Xen.

En este caso y dado que sólo podía montar mi sistema a a partir de un sistema operativo que tuviera instalador en red, he optado por Debian. Como vimos anteriormente, Debian ofrece toda la flexibilidad de las distribuciones GNU/Linux y además una fiabilidad, estabilidad y seguridad digna de los entornos de producción de carácter más empresarial. A pesar de mis experiencias previas con Ubuntu, Debian se trataba de una distribución completamente nueva para mí y me parecía interesante afrontar el aprendizaje de la instalación y configuración de un servidor de estas características basado en Debian.

Instalación de Debian + configuración de red SSH

Creamos un máquina con una configuración modesta para dicho fin.

1 núcleo

1gb de RAM

150 de disco (mas que suficiente para almacenar algunas imágenes ISO de los sistemas operativos que queremos instalar en nuestro entorno)

Creamos la máquina con el asistente de XenCenter para Windows y seleccionamos instalación desde URL, para esta máquina sólo necesitaremos una configuración básica con una configuración de red.

En las opciones de instalación seleccionamos instalación por URL.

Arrancamos la máquina y realizamos la instalación normal de Debian a través del asistente en modo consola. (detallar los pasos de la instalación e incluso alguna captura...). Como nombre de máquina y dado que su principal función será de tratará de una biblioteca de archivos, alexandria.

Una vez instalado el sistema base y configurada la contraseña del usuario root y habilitadas las conexiones remotas por SSH, el siguiente paso es conectarse a la máquina recién creada mediante el comando.

```
dafero@archenvy ~-> ssh root@156.35.95.20
```

Una vez conectado (captura de conexión y prompt) comenzaremos con la creación de la biblioteca NFS ISO.

Instalación y configuración de nuestra biblioteca NFS

En primer lugar comprobaremos que todos los paquetes del sistema se encuentran correctamente actualizados. Para este caso concreto dado que se acaba de realizar la instalación del sistema, este paso no sería estrictamente necesario. Normalmente, siempre y cuando durante el proceso de instalación haya acceso a Internet, el propio instalador se encargará de descargar e instalar las ultimas versiones disponibles de los paquetes. Aun así nunca está de más verificar que todas las dependencias están satisfechas y que el sistema está correctamente actualizado.

Para asegurarnos de esto en Debian y derivadas podemos ejecutar el siguiente comando:

```
root@alexandria:~# sudo apt-get update && sudo apt-get upgrade
```

Una vez terminado el proceso, procedemos con la instalación de los paquetes necesarios para crear nuestra biblioteca NFS

```
root@alexandria:~# apt-get install nfs-kernel-server nfs-common
```

Configuración

En primer lugar crearemos el directorio en donde se alojarán los archivos ISO de nuestra biblioteca. En este caso y por simplicidad, se creará un nuevo directorio llamado *myshare* en /home, aunque tanto la ruta como el nombre del directorio pueden ser modificadas libremente para cada caso concreto.

Comencemos con la creación del directorio. Para realizarlo teclearemos el siguiente comando:

```
root@alexandria:~# mkdir /home/myshare
```

Para configurar el acceso a nuestra almacén NFS necesitamos añadir una línea adicional al fichero *exports* en la que indicaremos desde qué redes será posible acceder a nuestra biblioteca. Este paso es importante puesto que no queremos que servicios externos a este proyecto se conecten y puedan ver el contenido de nuestra biblioteca.

El formato de cada una de estas líneas del fichero *exports* es siempre el mismo, en primer lugar tecleamos la ruta absoluta al directorio que será exportado y a continuación y separado por un espacio, la lista de direcciones o sub-redes de los clientes a los que les será permitido montar dicho directorio. Finalmente, y siempre en la misma línea, añadimos de una serie de opciones o *flags* que determinarán los permisos aplicados para los clientes que se conecten.

```
#/ruta/al_directorio_a/exportar IP/MASCARA(opcion1, opcion2...)
IP2/MASCARA2(opcion1, opcion2...)
```

```
/home/example 192.168.0.0/255.255.255.0(ro,no_subtree_check)
```

Una vez explicado esto, procedemos a editar el fichero *exports* que se encuentra en: */etc/exports*. En esta documentación utilizaremos *vim* para editar el archivo, pero queda a disposición del lector utilizar el editor que considere conveniente.

```
root@alexandria:~# vim /etc/exports
```

Lo abrimos y añadimos la siguiente línea.

```
/home/myshare 156.35.94.186(ro,no_subtree_check)
```

En este caso y tal y como había comentado anteriormente, por motivos de seguridad exportaré el directorio *myshare* contenido en */home* únicamente para la IP 156.35.94.186 que es la que corresponde al servidor *xenserver05*. Dentro de las opciones habilitaremos sólo permisos de lectura (*ro*= read only) y con la opción desactivada para que no compruebe los sub-árboles de directorios (*no_subtree_check*).

Permitiendo conexiones entrantes.

Una vez configurado lo anterior editamos el archivo */etc/hosts.allow* con aquellas IP o rangos de IPs para las que queremos permitir la conexión.

La sintaxis de *host.allow* es la siguiente, *nombre_servicio*: *IP/Mascara*. En mi caso y dado que sólo pretendo que el servidor NFS sea accesible por *xenserver-05* (156.35.94.186), únicamente configuraré el acceso para dicha dirección IP sin necesidad de aplicar ningún tipo de máscaras o rangos.

De esta forma nos aseguramos que sólo tendrán acceso desde esa IP concreta y limitaremos la entrada a un posible atacante.

Con todo lo anterior, la información relevante del fichero */etc/hosts.allow* nos quedaría así:

```
root@alexandria:~# vim /etc/hosts.allow
```

```
#
```

```
portmap: 156.35.94.186
lockd: 156.35.94.186
rquotad: 156.35.94.186
mountd: 156.35.94.186
statd: 156.35.94.186
```

Firewall

En el caso de tener un firewall activado en el sistema (una opción altamente recomendable debido que la maquina estará expuesta directamente a Internet) será necesario abrir en éste ciertos puertos para garantizar la correcta comunicación entre el cliente y el servidor NFS.

Para ello editamos el archivo **/etc/default/nfs-common**:

```
root@alexandria:~# vim /etc/default/nfs-common
```

Y añadimos la siguiente información:

```
# Options for rpc.statd.
STATDOPTS="-p 32765 -o 32766"

# vi /etc/default/nfs-kernel-server
# Options for rpc.mountd.

RPCMOUNTDOPTS="--manage-gids -p 32767"

#
```

Por último sólo nos queda crear la reglas necesarias en nuestro cortafuegos. En el caso de Debian el firewall instalado en el sistema es iptables. Su sintaxis puede resultar algo complicada para los usuarios nóveles, por lo que para manejarlo utilizaremos la cómoda interfaz que nos proporciona *Uncomplicated Firewall* (ufw).

En el caso de no tener *ufw* instalado en el sistema tecleamos lo siguiente:

```
root@alexandria:~# sudo apt-get install ufw -y
```

Una vez instalado veremos que tanto añadir modificar eliminar regla con ufw es una tarea sumamente sencilla.

```
ufw allow from IP to IP 2 port XXXX proto protocolo
```

Como en los ejemplos anteriores, por una cuestión de seguridad sólo abriremos los puertos estrictamente necesarios y únicamente para la IP del servidor Xen Server.

El resto los mantendremos cerrados a la espera de necesitarlos.

```
root@alexandria:~# ufw allow from 156.35.94.186 to any port 111 proto tcp
root@alexandria:~# ufw allow from 156.35.94.186 to any port 111 proto udp
root@alexandria:~# ufw allow from 156.35.94.186 to any port 2049 proto tcp
root@alexandria:~# ufw allow from 156.35.94.186 to any port 2049 proto udp
root@alexandria:~# ufw allow from 156.35.94.186 to any port 32765:32767 proto tcp
root@alexandria:~# ufw allow from 156.35.94.186 to any port 32765:32767 proto udp
```

Una vez añadidas todas estas reglas (podemos usar el script de configuración automático disponible en el apartado de anexos) verificamos que las reglas han sido añadidas correctamente en el firewall tecleando

```
root@alexandria:~# ufw status
Status: active

To Action From
--
22 ALLOW Anywhere
111/tcp ALLOW 156.35.94.186
111/udp ALLOW 156.35.94.186
2049/tcp ALLOW 156.35.94.186
32765:32767/tcp ALLOW 156.35.94.186
32765:32767/udp ALLOW 156.35.94.186
2049/udp ALLOW 156.35.94.186
22 ALLOW Anywhere (v6)
root@alexandria:~#
```

Para comprobar que todo está funcionando correctamente utilizaremos el comando `exportfs`. La salida de este comando mostrará aquellos directorios que han sido exportados junto con la lista de IPs habilitadas para su acceso.

La salida del comando debería ser algo similar a la siguiente:

```
root@alexandria:~# exportfs
/home/myshare 156.35.94.186
root@alexandria:~#
```

Sólo nos queda reiniciar los servicios para que se apliquen las nuevas configuraciones

```
root@alexandria:~# /etc/init.d/nfs-common restart
[ ok ] Stopping NFS common utilities: idmapd statd.
[ ok ] Starting NFS common utilities: statd idmapd.
root@alexandria:~# /etc/init.d/nfs-kernel-server reload
[ ok ] Re-exporting directories for NFS kernel daemon....
# /etc/init.d/nfs-kernel-server restart
[ ok ] Stopping NFS kernel daemon: mountd nfsd.
[ ok ] Unexporting directories for NFS kernel daemon....
[ ok ] Exporting directories for NFS kernel daemon....
[ ok ] Starting NFS kernel daemon: nfsd mountd.
root@alexandria:~#
```

Una vez completado todo este proceso satisfactoriamente ya deberíamos tener nuestro servidor correctamente NFS configurado y listo para recibir conexiones.

Para probarlo, podemos copiar o descargar algunas imágenes de prueba en formato ISO en nuestro recién directorio exportado, en este caso: `/home/myshare` y comprobar posteriormente si el servidor Xen tiene acceso a ellos y puede cargar su contenido.

En este caso probaremos a descargar una de las imágenes públicas de Ubuntu.

```
root@alexandria:~# cd /home/myshare/
root@alexandria:/home/myshare# wget http://cloud-
```

```
images.ubuntu.com/trusty/current/trusty-server-cloudimg-amd64-disk1.img
root@alexandria:/home/myshare#
```

Uso de NFS xenserver

En este punto, si todo ha ido bien, ya tendremos nuestro servidor NFS configurado y listo para su uso. El siguiente paso será añadirlo como un nuevo volumen a nuestro xenserver para que podamos arrancar fácilmente máquinas desde nuestros propios archivos ISO.

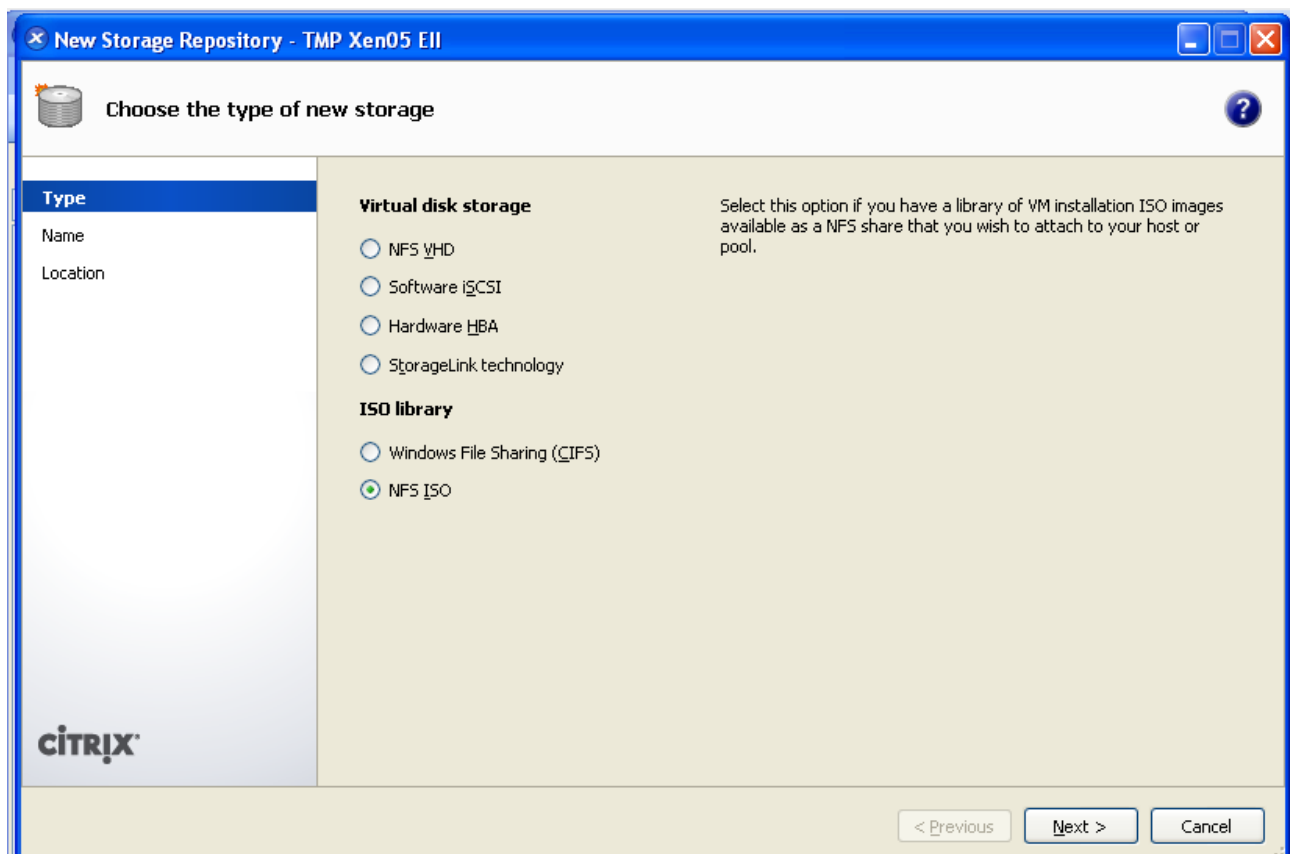
Añadir servidor NFS ISO a nuestro servidor XenCenter

En este caso tenemos dos opciones para poder realizar esto, la primera desde la interfaz gráfica y la segunda con la interfaz de comandos.

Comandos:

```
`showmount -e host`
mount the share from NFS remote Storage to the domU as a client
xe-mount-iso-sr 192.168.0.20:/nfs
Now it should be visible in xenCenter and have its own object ID.
`xe sr-list | grep -B 1 Remote`
```

O de forma gráfica



Instalación XenOrchestra

Como vimos anteriormente Xen Orchestra (XOA) es un software con el que podemos administrar un servidor XenServer cómodamente a través de una interfaz Web. En este apartado veremos los pasos necesarios para su instalación y configuración inicial. Desde la página oficial de XOA se puede encontrar dos tipos de instalación; Recomendada y manual.

Ambas son completamente válidas, aunque dado que cada una tiene como objetivo ser aplicada a un escenario diferente, en este apartado veremos **ambas** formas de instalación.

Instalación recomendada

El archivo de imagen consistía en un fichero de casi 700 megas que tenía que importar a nuestra máquina xenserver. **imagen .xva, disponible desde la página oficial del proyecto.**

Dos formas de hacerlo, interfaz gráfica o comandos.

La primera de ellas, a través de la interfaz gráfica proporcionada por el programa, tenía una importante desventaja para este entorno. Para importar la imagen debía, en primer lugar descargarla a mi ordenador, pasarla a mi máquina virtual y desde ahí utilizar la interfaz gráfica de usuario de administración del XenServer para importar la imagen. Lo que intentará subir la imagen de nuevo al servidor. En mi caso esto a pesar de que contaba con una conexión a Internet decente, este proceso no fue viable debido a que los tiempos de espera suponían varias horas.

Por tanto y a pesar la facilidad aparente del proceso, desgraciadamente éste no era aplicable en este entorno.

Interfaz de comandos: En este caso lo que se pretende es realizar el mismo proceso de importación de imagen pero desde la interfaz de línea de comandos del Xensever.

Utilizaremos el almacén NFS anteriormente configurado para almacenar la imagen y posteriormente importarla directamente desde allí.

(<https://sites.google.com/site/norandatechnology/xenserver/export-import-template-nfs>)

Nos conectamos por SSH

```
ssh root@156.35.94.186
```

```
[root@xenserver-05 scp]# xe sr-list
uuid ( R0)                : 418392b2-1068-b6a8-bb04-f02cfc3211fe
  name-label ( RW): Removable storage
  name-description ( RW):
    host ( R0): xenserver-05
    type ( R0): udev
  content-type ( R0): disk

uuid ( R0)                : f3171d4c-968f-5dfd-e084-aa4da645e813
  name-label ( RW): DVD drives
  name-description ( RW): Physical DVD drives
    host ( R0): xenserver-05
    type ( R0): udev
  content-type ( R0): iso

uuid ( R0)                : eb8b5f62-2456-5500-442a-75bcfe9df35d
  name-label ( RW): NFS ISO library
  name-description ( RW):
    host ( R0): xenserver-05
    type ( R0): iso
```

```

        content-type ( R0): iso

uuid ( R0)          : 673d83e2-5d1b-f308-a1c0-d273a5aea5cd
    name-label ( RW): Local Storage
    name-description ( RW):
        host ( R0): xenserver-05
        type ( R0): lvm
        content-type ( R0): user

uuid ( R0)          : cd145f36-a909-f8f6-bd2d-5362d9282107
    name-label ( RW): XenServer Tools
    name-description ( RW): XenServer Tools ISOs
        host ( R0): xenserver-05
        type ( R0): iso
        content-type ( R0): iso

```

Importar una máquina .xva en nuestro entorno Xen

Desde entorno gráfico: <http://support.citrix.com/proddocs/topic/xencenter-61/xs-xc-vms-import.html>

Instalación manual

En este caso y dado que no queremos destinar una máquina completa para esta funcionalidad, aprovecharemos la ya existente utilizada como servidor de imágenes ISO.

https://github.com/vatesfr/xo/blob/master/doc/installation/manual_installation.md

Realizaremos una instalación manual

Prerequisitos y paquetes

Para comenzar con la instalación será necesario disponer de las librerías de Node.js así como algunos paquetes estándar de desarrolladores como pueden ser las *build essentials*, *curl*, *wget*, *git* o *python*.

NodeJS

Tal y como recomiendan en la página oficial del desarrollo utilizaremos el gestor de binarios de Node.js llamado 'n' para instalar Node.js.

En primer lugar nos aseguramos de que curl y wget están instalados en nuestro sistema. Utilizaremos la opción '-y' para contestar afirmativamente de forma automática al dialogo de confirmación instalación por parte del usuario.

```
root@alexandria:~# apt-get install wget curl -y
```

Una vez confirmado, precedemos con la descarga del script de instalación de 'n' que guardaremos en /usr/local/bin/n.

```
root@alexandria:~# curl -o /usr/local/bin/n
https://raw.githubusercontent.com/visionmedia/n/master/bin/n
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
100	8393	100	8393	0	0	12991	0
--:--:--	--:--:--	--:--:--	--:--:--	--:--:--	--:--:--	--:--:--	15956

Una vez descargado, procedemos a darle permisos de ejecución mediante el comando `chmod +x` e instalamos la versión denominada como 'stable'.

```
root@alexandria:~# chmod +x /usr/local/bin/n
root@alexandria:~# n stable

install : v0.10.33
mkdir   : /usr/local/n/versions/0.10.33
fetch   : http://nodejs.org/dist/v0.10.33/node-v0.10.33-linux-x64.tar.gz
installed : v0.10.33

root@alexandria:~#
```

Para verificar que todo ha ido bien podemos ejecutar el siguiente comando que, en el caso de estar todo bien configurado, nos devolverá la versión instalada:

```
root@alexandria:~# node -v
v0.10.33
root@alexandria:~#
```

Procederemos ahora con la instalación de los paquetes requeridos:

```
root@alexandria:~# apt-get install build-essential redis-server libpng-dev git python-minimal -y
```

Una vez finalizado el proceso de actualización estamos listo para descargar el código del proyecto desde los repositorios de GitHub.

```
root@alexandria:~/xo# git clone http://github.com/vatesfr/xo-server
root@alexandria:~/xo# git clone http://github.com/vatesfr/xo-web
```

XO-Server

Pasaremos ahora a la instalación de dependencias del proyecto. Para ello lanzaremos el comando `npm install` desde el directorio en el que se encuentra `xo-server`.

```
root@alexandria:~/xo# cd xo-server/
root@alexandria:~/xo/xo-server# npm install
```

Una vez terminado el proceso de instalación de la parte del servidor copiaremos la plantilla de configuración contenida en el directorio `xo-server` a un archivo oculto en el mismo directorio de nombre `xo-server.xml`

```
root@alexandria:~/xo/xo-server# cp sample.config.yaml .xo-server.yaml
```

Y editamos la plantilla recién copiada para configurarla a nuestro gusto. Para este caso concreto, modificaremos el puerto donde escuchará la aplicación y el uso de `https` en detrimento de `http`.

```
# Basic HTTPS.
```



```
# -  
# # The only difference is the presence of the certificate and the  
# # key.
```

```
host: '156.35.95.20'
```

```
port: 443
```

```
# # File containing the certificate (PEM format).
```

```
# #
```

```
# # Default: undefined
```

```
certificate: './certificate.pem'
```

```
# # File containing the private key (PEM format).
```

```
# #
```

```
# # If the key is encrypted, the passphrase will be asked at
```

```
# # server startup.
```

```
# #
```

```
# # Default: undefined
```

```
key: './key.pem'
```

```
# List of files/directories which will be served.
```

```
mounts:
```

```
['/': './xo-web/dist/']
```

Como se puede apreciar en las líneas remarcadas, para que esto funcione será necesario crear los archivos correspondientes al certificado y a la clave privada. En este caso y por motivos de simplicidad optaremos por generar nuestro propio certificado auto-firmado que una vez que accedamos desde el navegador tendremos que aceptar.

```
root@alexandria:~/xo/xo-server# openssl req -x509 -newkey rsa:2048 -keyout  
key.pem -out certificate.pem -days 365
```

```
Generating a 2048 bit RSA private key
```

```
.....+++
```

```
.....
```

```
writing new private key to 'key.pem'
```

```
Enter PEM pass phrase:
```

```
Verifying - Enter PEM pass phrase:
```

```
-----
```

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
```

```
-----
```

```
Country Name (2 letter code) [AU]: ES
```

```
State or Province Name (full name) [Some-State]: Asturias
```

```
Locality Name (eg, city) []: Oviedo
```

```
Organization Name (eg, company) [Internet Widgits Pty Ltd]: EII
```

```
Organizational Unit Name (eg, section) []: IT
```

```
Common Name (e.g. server FQDN or YOUR name) []: alexandria.tfg
```

```
Email Address []: daferoes@gmail.com
```

Una vez terminado el proceso de generación verificamos que los archivos han sido creador correctamente.

```
root@alexandria:~/xoa/xo-server# ls -la
total 48
4 drwxr-xr-x 2 root root 4096 Dec 4 15:30 bin
4 -rw-r--r-- 1 root root 1411 Dec 4 17:21 certificate.pem
4 -rw-r--r-- 1 root root 1844 Dec 4 15:30 coffeelint.json
4 -rw-r--r-- 1 root root 162 Dec 4 15:30 index.js
4 -rw-r--r-- 1 root root 1834 Dec 4 17:21 key.pem
4 drwxr-xr-x 56 root root 4096 Dec 4 15:31 node_modules
4 -rw-r--r-- 1 root root 2224 Dec 4 15:30 package.json
4 -rw-r--r-- 1 root root 1164 Dec 4 15:30 README.md
4 -rwxr-xr-x 1 root root 561 Dec 4 15:30 run-tests
4 -rw-r--r-- 1 root root 2693 Dec 4 15:30 sample.config.yaml
4 drwxr-xr-x 4 root root 4096 Dec 4 15:30 src
4 -rw-r--r-- 1 root root 218 Dec 4 15:30 xo-server.service
```

XO-Web

Parecido al caso anterior, aunque esta vez instalaremos aquellas dependencias necesarias para poder desplegar la interfaz Web de la aplicación. Situados en el directorio xo-web ejecutamos lo siguiente.

```
root@alexandria:~/xoa/xo-web# npm install
```

Debido a un bug en el código, antes de construir la aplicación será necesario modificar el archivo bower y actualizar la versión de angular. Este problema está documentado en la sección de Problemas encontrados al final de este documento.

Finalmente ejecutamos

```
root@alexandria:~/xoa/xo-server# ./gulp --production
```

Y si todo va como es esperado podremos ejecutar el archivo que despliega el servidor.

```
./bin/xo-server
```

Obviamente para que tengamos acceso desde el navegador será necesario abrir el puerto en el que se está ejecutando la aplicación y que hemos configurado en nuestro archivo de propiedades .xo-server.yaml. En mi caso concreto he configurado el servicio el 443 así que procederemos a abrir este puerto para todas las conexiones entrantes.

```
root@alexandria:~# ufw allow from any to any port 443 proto tcp
```

Documentar primeros pasos con XOA
crear usuario y cambiar contraseña

Actualizando XenOrchestra

Una vez instalado

service xo update

Y despues reiniciamos el servicio

service xo restart

Instalación OpenStack

Instalación de Scientific Linux, con estos parámetros (tabla resumen, ip hardware y demás)

La instalación de Scientific Linux no tiene demasiadas complicaciones si seguimos los pasos del asistente, elegimos un contraseña de administrador y nos conectamos mediante la consola.

Una vez conectados lo primero que deberemos hacer es dotar nuestra nueva máquina de su IP estática para que de esta forma pueda comenzar a comunicarse con el resto de maquinas de su entorno.

Al tratarse de una distribución basada en CentOS que a su vez está basada en RedHat Enterprise Linux el procedimiento de configuración será similar al requerido para cualquiera de estas.

<http://www.putorius.net/2012/10/how-to-configure-static-ip-address-in.html>

Reiniciamos el servicio de red y probamos que tenemos conectividad con el exterior (ping a uno de los servidores públicos DNS de Google).

Ping 8.8.8.8

```
[root@lenuage ~]# ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=45 time=45.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=45 time=45.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=45 time=45.4 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=45 time=45.3 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3049ms
rtt min/avg/max/mdev = 45.214/45.430/45.798/0.268 ms
[root@lenuage ~]#
```

Una verificado que tenemos conectividad con el exterior probaremos que nuestro servidor DNS resuelve correctamente.

```
[root@lenuage ~]# ping -c 4 www.google.es
PING www.google.es (74.125.230.56) 56(84) bytes of data.
64 bytes from mad01s25-in-f24.1e100.net (74.125.230.56): icmp_seq=1 ttl=53
time=7.85 ms
64 bytes from mad01s25-in-f24.1e100.net (74.125.230.56): icmp_seq=2 ttl=53
time=7.64 ms
64 bytes from mad01s25-in-f24.1e100.net (74.125.230.56): icmp_seq=3 ttl=53
time=7.65 ms
64 bytes from mad01s25-in-f24.1e100.net (74.125.230.56): icmp_seq=4 ttl=53
time=7.66 ms

--- www.google.es ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
```

```
rtt min/avg/max/mdev = 7.644/7.704/7.858/0.124 ms
[root@lenuage ~]#
```

Una vez finalizada la configuración la conectividad con el exterior, revisaremos que SSH está activado para dejar de utilizar la limitada consola web que nos proporciona XenCenter. Service sshd status

El servicio está corriendo. Ahora comprobaremos si el firewall está configurado para recibir peticiones a través de puerto 22 (por defecto el puerto que usa SSH)

```
[root@lenuage ~]# iptables -S | grep '\-\-dport 22'
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
[root@lenuage ~]#
```

*en el caso de arriba utilizamos el comando grep para filtrar la salida de iptables y que así sólo nos muestre la información que nos interesa

Vemos que el puerto 22 está abierto y escuchando.

Instalación de OpenStack usando RDO Packstack

Copias de seguridad

En este apartado trataremos uno de los temas más importantes que hay que tener en cuenta al llevar a producción un proyecto como este. Así mismo se identificaran algunos de los errores que se han cometido y que no deberían cometerse en un entorno de producción real.

A continuación algunas recomendaciones a tener en cuenta, la mayoría son de sentido común, pero nunca esta de más releerlas, reflexionar sobre ellas e intentar aplicarlas en la medida de lo posible.

Las copias de seguridad deberían estar en sitios separados

Para las copias de seguridad de los archivos accesorios al proyecto, documentación, bibliografía, imágenes etc. se ha utilizado un servidor de copias en nube.

Llevar a cabo un proyecto como este no sería posible sin un sistema de copias de seguridad apropiado. Todos los archivos accesorios a este proyecto (documentación, ficheros de configuración, bibliografía, etc) se guardarán de forma local en mi ordenador personal (desde el cual se realizara el proyecto) y de manera simultánea, se mantendrá una copia de dichos archivos en una ubicación distinta.

Como no podía ser de otro modo en este proyecto, las copias de seguridad serán almacenadas en la nube.

En el caso de las máquinas virtuales sobre las que se sustenta el proyecto, no se mantendrán copias de seguridad en ubicaciones distintas. Se ha tenido que llegar a este compromiso, no sólo por una cuestión espacio si no también por una limitación de red. Desgraciadamente, mover decenas de GB de información entre dos ordenadores conectados por WAN no es algo temporalmente asumible con las conexiones de datos actuales. Llegados a este caso es más que probable que se invirtiese menor tiempo en reinstalar todo el sistema de nuevo, que en copiar y restaurar la imagen desde otra ubicación.

Donde sí se mantendrán copias de las máquinas virtuales será localmente en el servidor

xenserver05. Así mismo se mantendrán una serie de instantáneas o *snapshots* de cada una de las máquinas, creadas antes de realizar cambios considerados como relevantes en la configuración.

Así en el caso de catástrofe, será posible devolver la máquina a un estado de configuración anterior en apenas segundos a partir de su instantánea previa. Si este proceso de restauración no funcionase (por desgracia proceso más frecuente de lo que parece en Xen), siempre sería posible volver a un estado temprano, en el cual ya se hubieran realizado algunas de los pasos más tediosas como pueden ser la instalación del sistema operativo, la configuración de red y acceso. De esta forma en apenas unos minutos podremos volver a disponer de las máquinas en un estado en el que sólo será necesario reconfigurar algunos archivos. Una vez finalizado el proyecto se crearán nuevas copias completas de dichas máquinas virtuales para poder volver al estado final de este proyecto en el futuro.

Como ya comenté anteriormente, soy consciente de que este método de guardado no es el más adecuado, pero debido al peso de la máquina y las conexiones a Internet actuales, mantener una copia completa en otra ubicación fuera del entorno de la Universidad y restaurarla en caso de catástrofe, llevaría muchísimo más tiempo que el hecho de reinstalarla y reconfigurarla una vez que se poseen los conocimientos y los ficheros de configuración pertinentes.

GIT

ownCloud

ownCloud es un software desarrollado por la empresa *ownCloud Inc.* que nos proporciona de una forma sencilla y automatizada disfrutar de nuestro propio servidor en el que poder alojar cualquier tipo de datos. Además de la parte servidor, ownCloud proporciona un cliente de escritorio multiplataforma gracias al cual podemos mantener nuestros archivos sincronizados en nuestros ordenadores y dispositivos móviles de una forma cómoda. Podríamos considerar ownCloud como un Dropbox en el que el propio cliente puede ser el encargado de alojar sus propios archivos. De esta forma desaparecen por completo los posibles problemas relacionados con la privacidad o el acceso por parte de terceros a nuestros datos sin consentimiento.

En cuanto al modelo de negocio de la empresa que está detrás de ownCloud, es muy similar al de otras compañías del sector open-source, ofrecen un producto sin ningún tipo de coste de licencia por uso y posteriormente venden soporte o ediciones "empresariales" de ese mismo, en las que se pulen detalles más orientados al mundo de la empresa.

Instalación y configuración de ownCloud en Debian

A lo largo de este apartado cubriremos el proceso de instalación y configuración de nuestro almacén de copias ownCloud. En cuanto a la justificación del sistema operativo anfitrión, como en otras ocasiones se ha elegido Debian por su conocida estabilidad, fiabilidad y seguridad. Como en el caso del XenOrchestra se ha optado por reutilizar el servidor inicialmente dedicado a proporcionarnos la biblioteca de imágenes ISO

Instalación

En primer lugar, antes incluso de proceder con la instalación en sí misma, descargamos y añadimos la clave pública del repositorio de ownCloud a nuestro almacén de claves del sistema. Este paso es importante dado que en caso contrario, añadiendo el repositorio y actualizando, nos saltaría el siguiente error de clave GPG:

```
http://download.opensuse.org Release: The following signatures couldn't be
verified because the public key is not available: NO_PUBKEY 977C43A8BA684223
```

Por tanto, para añadirla tecleamos como root:

```
root@alexandria:~# cd /tmp/
root@alexandria:/tmp# wget
http://download.opensuse.org/repositories/isv:ownCloud:community/Debian_7.0/Release.key
--2014-12-10 19:42:50--
http://download.opensuse.org/repositories/isv:ownCloud:community/Debian_7.0/Release.key
Resolving download.opensuse.org (download.opensuse.org)... 195.135.221.134,
2001:67c:2178:8::13
Connecting to download.opensuse.org (download.opensuse.org)|
195.135.221.134|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location:
http://download.opensuse.org/repositories/isv:ownCloud:/community/Debian_7.0/Release.key [following]
--2014-12-10 19:42:51--
http://download.opensuse.org/repositories/isv:ownCloud:/community/Debian_7.0/Release.key
Reusing existing connection to download.opensuse.org:80.
HTTP request sent, awaiting response... 301 Moved Permanently
Location:
http://download.opensuse.org/repositories/isv:/ownCloud:/community/Debian_7.0/Release.key [following]
--2014-12-10 19:42:51--
http://download.opensuse.org/repositories/isv:/ownCloud:/community/Debian_7.0/Release.key
Reusing existing connection to download.opensuse.org:80.
HTTP request sent, awaiting response... 200 OK
Length: 1003 [application/pgp-keys]
Saving to: `Release.key'

100%
[=====
=====
=====>] 1,003      --.-K/s   in 0s

2014-12-10 19:42:51 (88.4 MB/s) - `Release.key' saved [1003/1003]

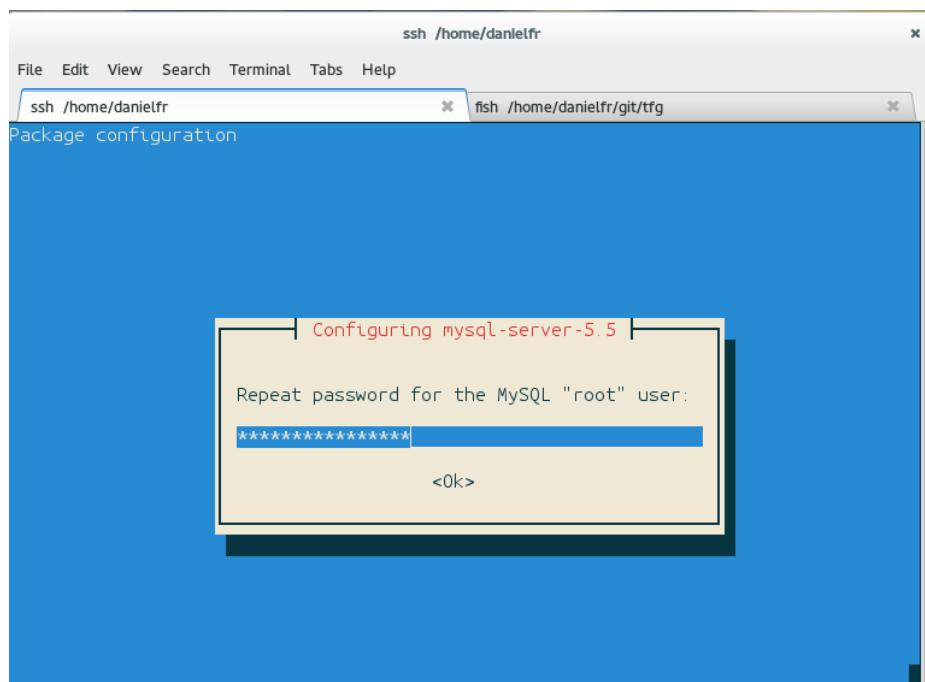
root@alexandria:/tmp# apt-key add - < Release.key
OK
root@alexandria:/tmp# echo 'deb
http://download.opensuse.org/repositories/isv:/ownCloud:/community/Debian_7.0/ /
' >> /etc/apt/sources.list.d/owncloud.list
```

```
root@alexandria:/tmp#
```

Una vez realizado lo anterior actualizamos el sistema y procedemos con la instalación del paquete *owncloud*

```
root@alexandria:/tmp# apt-get update && apt-get install owncloud
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2 apache2-mpm-prefork apache2-utils apache2.2-bin apache2.2-common dbus
  fontconfig-config fonts-droid ghostscript gsfonts imagemagick-common libaiol
  libapache2-mod-php5 libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libavahi-client3 libavahi-common-data
  libavahi-common3 libcups2 libcupsimage2 libdbd-mysql-perl libdbi-perl libdbus-1-
  3 libffi5 libfontconfig1 libgd2-xpm libglib2.0-0
  libglib2.0-data libgs9 libgs9-common libhtml-template-perl libice6 libicu48
  libijs-0.35 libjasper1 libjbig0 libjbig2dec0 libjpeg8 liblcms2-2 liblqr-1-0
  libltdl7 libmagickcore5 libmagickwand5 libmcrypt4
  libmysqlclient18 libonig2 libpaper-utils libpaper1 libpq5 libqdbm14 libsm6
  libsystemd-login0 libtiff4 libxpm4 libxt6 mysql-client-5.5 mysql-common mysql-
  server mysql-server-5.5 mysql-server-core-5.5 php-pear
  php-xml-parser php5 php5-cli php5-common php5-curl php5-gd php5-imagick php5-
  intl php5-mcrypt php5-mysqld php5-pgsql php5-sqlite poppler-data shared-mime-
  info ssl-cert ttf-dejavu-core x11-common
Suggested packages:
  apache2-doc apache2-suexec apache2-suexec-custom dbus-x11 ghostscript-cups
  ghostscript-x hpijs cups-common libgd-tools libipc-sharedcache-perl libjasper-
  runtime liblcms2-utils libmagickcore5-extra
  libmcrypt-dev mcrypt libterm-readkey-perl tinyca clamav clamav-daemon
  smbclient libav-tools ffmpeg libreoffice-writer php5-dev poppler-utils fonts-
  japanese-mincho fonts-ipafont-mincho fonts-japanese-gothic
  fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-unfonts-core
  openssl-blacklist
Recommended packages:
  php5-apc
The following NEW packages will be installed:
  apache2 apache2-mpm-prefork apache2-utils apache2.2-bin apache2.2-common dbus
  fontconfig-config fonts-droid ghostscript gsfonts imagemagick-common libaiol
  libapache2-mod-php5 libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libavahi-client3 libavahi-common-data
  libavahi-common3 libcups2 libcupsimage2 libdbd-mysql-perl libdbi-perl libdbus-1-
  3 libffi5 libfontconfig1 libgd2-xpm libglib2.0-0
  libglib2.0-data libgs9 libgs9-common libhtml-template-perl libice6 libicu48
  libijs-0.35 libjasper1 libjbig0 libjbig2dec0 libjpeg8 liblcms2-2 liblqr-1-0
  libltdl7 libmagickcore5 libmagickwand5 libmcrypt4
  libmysqlclient18 libonig2 libpaper-utils libpaper1 libpq5 libqdbm14 libsm6
  libsystemd-login0 libtiff4 libxpm4 libxt6 mysql-client-5.5 mysql-common mysql-
  server mysql-server-5.5 mysql-server-core-5.5 owncloud
  php-pear php-xml-parser php5 php5-cli php5-common php5-curl php5-gd php5-
  imagick php5-intl php5-mcrypt php5-mysqld php5-pgsql php5-sqlite poppler-data
  shared-mime-info ssl-cert ttf-dejavu-core x11-common
0 upgraded, 81 newly installed, 0 to remove and 11 not upgraded.
Need to get 80.6 MB of archives.
After this operation, 332 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

Durante la instalación se nos abrirá un dialogo para configurar el acceso a la base de datos SQL que servirá de *backend* de la aplicación. Esta base de datos estará basada en una MySQL en su versión 5.5.



Una vez finalizada la instalación ya estaremos listos para utilizar ownCloud <http://156.35.95.20/owncloud/index.php>. Aunque antes de comenzar realizaremos algunos ajustes en la configuración para poder utilizar HTTP/SSL (importantísimo si no queremos que nuestras credenciales viajen en texto plano) y cambiar el puerto donde se ejecuta el servicio dado que es más que probable que en el 80/443 ya estemos ejecutando otros servicios.

Como se puede observar del apartado anterior echar a andar una configuración básica de ownCloud es un proceso bastante directo y sencillo. A pesar de esto, una vez instalado es recomendable al menos configurar el acceso por https/ssl para que así todo acceso a la aplicación web sea cifrado y aquellos datos sensibles como, nombres de usuario, contraseñas o archivos no viajen por la red en texto plano.

A continuación veremos como configurarlo.

En primer lugar y como cuando confirmamos el acceso https de XenServer, crearemos nuestros propios certificados. Aunque si bien es cierto que en este caso seguiremos un proceso similar pero algo distinto. debido al tipo de aplicación. Por simplicidad los almacenaremos el certificado en un directorio que crearemos para este fin en el lugar donde se encuentra la configuración de Apache.

Generación de Certificados

En primer lugar crearemos un directorio en el que almacenaremos el certificado.

```
root@alexandria:~# cd /etc/apache2/
```



```
root@alexandria:~# mkdir CertOwncloud
root@alexandria:~# cd CertOwncloud
```

Una vez dentro del directorio ejecutaremos lo siguiente:

En primer lugar generamos la clave:

```
root@alexandria:/etc/apache2/CertOwncloud# openssl genrsa -out owncloud.key 2048
Generating RSA private key, 2048 bit long modulus
....+++
.....+++
e is 65537 (0x10001)
```

Una vez generada, crearemos nuestros .key y .csr

```
root@alexandria:/etc/apache2/CertOwncloud# openssl req -new -key owncloud.key
-out owncloud.csr
```

Y a partir de ellos estos dos archivos, crearemos finalmente nuestro certificado:

```
root@alexandria:/etc/apache2/CertOwncloud# openssl x509 -req -days 365 -in
owncloud.csr -signkey owncloud.key -out owncloud.crt
```

Una vez creado sólo nos quedará copiarlo en aquellos directorios donde será usado. En nuestro caso

```
root@alexandria:/etc/apache2/CertOwncloud#cp owncloud.crt /etc/ssl/certs
root@alexandria:/etc/apache2/CertOwncloud#cp owncloud.key /etc/ssl/private
```

Configuración Apache

En el caso de la configuración de Apache solo sera necesario añadir un solo archivo, con el contenido que veremos a continuación.

Para ello, creamos un nuevo fichero llamado 'owncloud.https' dentro del siguiente directorio /etc/apache2/sites-available/

```
root@alexandria: vim /etc/apache2/sites-available/owncloud.https
```

En el que copiaremos y pegaremos la siguiente configuración.

```
NameVirtualHost *:4433
<VirtualHost *:4433>
DocumentRoot /var/www/owncloud
# Configuración de SSL
SSLEngine On
SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
# Ruta a los Certificados
SSLCertificateFile /etc/ssl/certs/owncloud.crt
SSLCertificateKeyFile /etc/ssl/private/owncloud.key
</VirtualHost>
```

Nótese la configuración de puertos, en este caso usaremos 4433, así como la ruta especificada en DocumentRoot y los directorios donde previamente guardamos el certificado y la llave.

Dado que estamos utilizando un puerto de los denominados 'no estándar', cambiaremos la

configuración del puerto de escucha de Apache, por omisión en el 80. Para ello abrimos el archivo `ports.conf` dentro de la configuración de Apache y modificamos el valor al que hace referencia la etiqueta *Listen* por el puerto especificado anteriormente, 4433.

```
root@alexandria:~# vim /etc/apache2/ports.conf

# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default
# This is also true if you have upgraded from before 2.2.9-3 (i.e. from
# Debian etch). See /usr/share/doc/apache2.2-common/NEWS.Debian.gz and
# README.Debian.gz

NameVirtualHost *:80
Listen 4433

<IfModule mod_ssl.c>
    # If you add NameVirtualHost *:443 here, you will also have to change
    # the VirtualHost statement in /etc/apache2/sites-available/default-ssl
    # to <VirtualHost *:443>
    # Server Name Indication for SSL named virtual hosts is currently not
    # supported by MSIE on Windows XP.
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

Por último sólo debemos activar algunos módulos en Apache y reiniciar el servicio para que los últimos cambios realizados en los ficheros de configuración sean aplicados.

```
root@alexandria:~# a2enmod ssl
root@alexandria:~# a2ensite owncloud.https
root@alexandria:~# apachectl configtest
root@alexandria:~# service apache2 restart
[ ok ] Restarting web server: apache2 ... waiting .
root@alexandria:~#
```

Con esto ya hemos finalizado con Apache, aunque antes de probarlo será necesario añadir una regla en nuestro corta fuegos con el puerto definido anteriormente. Como en anteriores ocasiones, utilizaremos la interfaz para *iptables*, llamada *ufw*.

```
root@alexandria:~# ufw allow 4433
```

Ahora sí, lo único que nos quedaría es probarlo para verificar su correcto funcionamiento. Abrimos un navegador y tecleamos <https://156.35.95.20:4433>

Y si todo va bien veremos la pantalla de ingreso de ownCloud en la que sólo nos quedará definir un usuario administrador de la aplicación así como la configuración de la base de datos.

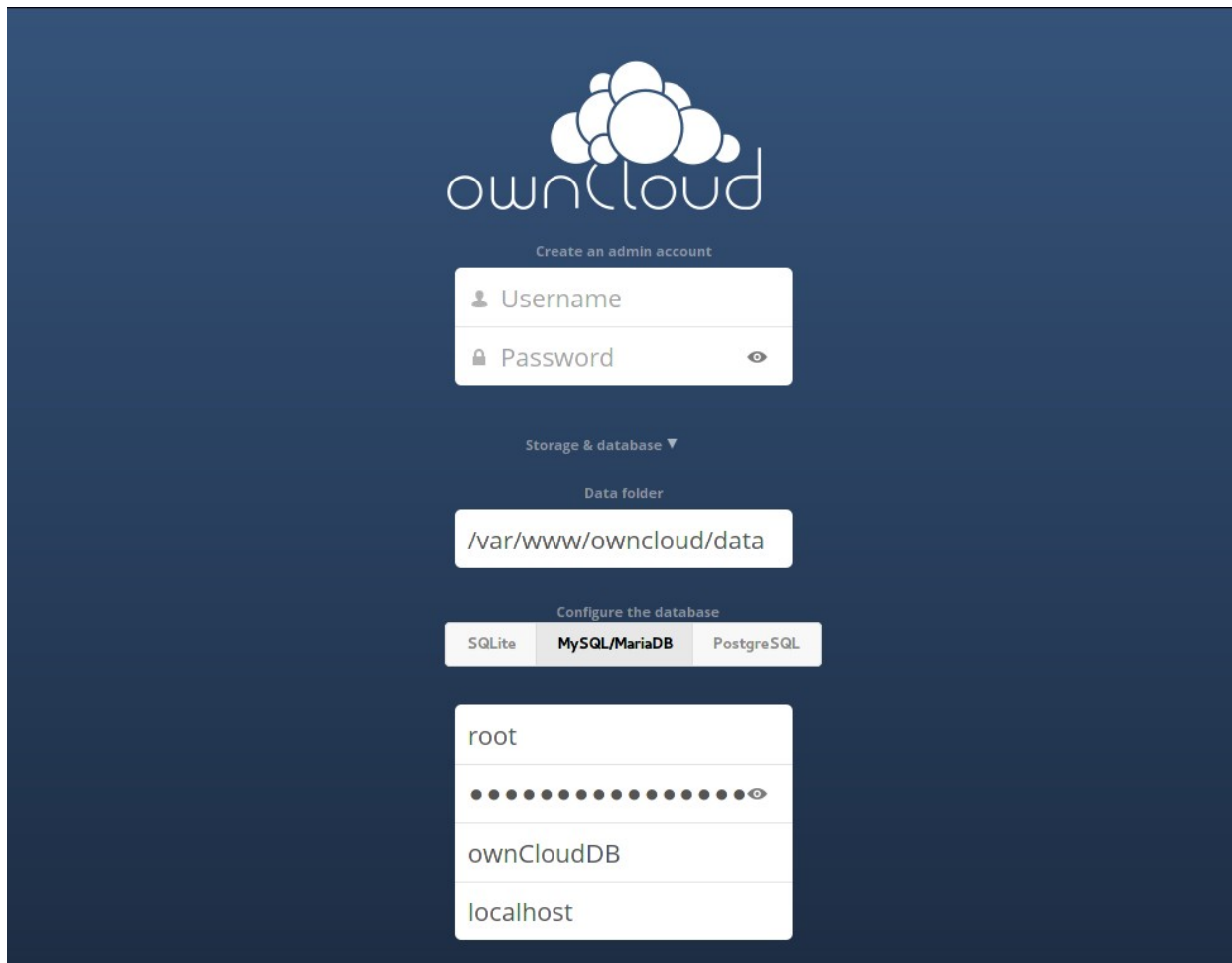


Imagen 1:

Cubrir la instalación del cliente y capturas y demás.

Una vez instalada la versión de servidor, instalaremos el cliente de escritorio que nos permitirá mantener nuestros archivos sincronizados en todo momento.

Desde la pagina oficial del proyecto proporcionan los binarios necesarios para la instalación del cliente de ownCloud para los sistemas operativos más comunes del mercado:

Windows, Mac y Linux. En el caso de Linux nos proporciona los binarios para algunas de las distribuciones mas extendidas CentOS/RHEL, Fedora, openSUSE, Ubuntu y Debian. Así mismo y dado que está licenciado bajo una licencia de código abierto, es posible descargar directamente los archivos fuente del programa, para poder instalarlo desde tu propia compilación.

En mi caso concreto y dado que utilizo Arch Linux en mi ordenador personal, he optado por instalarlo desde los repositorios AUR (Arch User Repository) en lo que se puede encontrar la última versión compatible con este sistema, lista para ser instalada.

Para instalarla abrimos una terminal de órdenes y escribimos.

```
dafero@archenvy ~-> sudo yaourt -S owncloud-client
```

Una vez instalador sólo queda configurar el cliente para que se conecte a nuestro servidor.

Instalación de la primera máquina en Xen Server

OpenStack IceHouse

OpenStack IceHouse es el noveno lanzamiento de OpenStack desde que comenzó oficialmente su desarrollo en la segunda mitad del año 2010. IceHouse fue liberado al público en mayo de 2014 y desde su lanzamiento y hasta la actualidad han salido cuatro revisiones en las que se han arreglado errores y fallos, mejorando su estabilidad y fiabilidad.

Historia del desarrollo

OpenStack comenzó su andadura de la mano de la unión de dos proyectos pertenecientes a dos grandes empresas, la primera Rackspace una empresa americana de computación y alojamiento en la nube y la segunda, NASA (Agencia Espacial Norteamericana), A principios del año 2010, Rackspace liberó bajo licencia de código abierto su conocido sistema operativo basado en tecnologías de nube, Swift. Por otro lado, NASA estaba utilizando un proyecto libre de desarrollo propio llamado OpenNebula, aunque estaba mucho de ser una solución madura.

De la unión de estos dos proyectos surgió OpenStack para dar una respuesta libre a la ya por aquel entonces ampliamente utilizada, Amazon EC2 (Amazon Elastic Cloud Computing).

Desde su creación en el año 2010, diversas versiones de OpenStack han visto la luz.

OpenStack tienen un modelo de desarrollo muy rápido y ágil, con lanzamiento de versiones nuevas cada aproximadamente 6 meses. Como en otros proyectos software, el sistema de nombrado de versiones sigue un orden alfabético, comenzando por la A.

Con esto, tenemos los siguientes nombres clave para las distintas versiones: Austin, Bexar, Cactus, Diablo, Essex, Grizzly, Havana, Icehouse, Juno y la futura Kilo.

Gráfico con los lanzamientos desde 2010. Austin, Bexar, Cactus, Diablo, Essex, Grizzly, Havana, Icehouse, Juno y la futura Kilo.

<http://dcxyc.files.wordpress.com/2014/04/openstack-history.png>

Para facilitar la vida a los desarrolladores el sistema, el nombrado de paquetes es un poco distinto y en lugar de utilizar el nombre en clave se utiliza la nomenclatura: componente/proyecto, año, lanzamiento, revisión.

Nombre del componente/proyecto: Keystone, glance, nova...

Año: año de lanzamiento, 2012, 2013..

Lanzamiento: Se liberan dos versiones anuales por lo que este campo solo puede contener los valores 1 o 2.

Revisión: Distintas revisiones para un mismo paquete.

Con todo esto para un nombre de paquete:

openstack-keystone-2014.1.2

Tenemos el componente keystone, lanzado en la primera mitad del año 2014 (nombre de versión IceHouse) y revisión número 2.

Summits

Otra característica importante dentro de la comunidad OpenStack son los llamados summits. Estos summits son una serie de conferencias de 5 días de duración que se celebran cada 6 meses en distintas ciudades del mundo. Es un punto de encuentro entre desarrolladores, clientes y usuarios finales en los que se discute que decisiones tomará el proyecto durante los próximos ciclos. Este tipo de eventos contribuyen sin duda a publicitar el uso de OpenStack y ayudan a la -cada año creciente- comunidad de usuarios y desarrolladores.

Componentes

OpenStack está dividido en distintos módulos separados por funcionalidad. Con cada nuevo lanzamiento, nuevos módulos aparecen. Este desarrollo modular permite un rápido crecimiento y evolución del proyecto.

Hay unos módulos que pertenecen al núcleo o "core" de Openstack y otros que podríamos considerar como accesorios. La configuración elegida en cada caso dependerá de las características particulares del proyecto.

A continuación hablaremos de algunos de los módulos más importantes que forman parte del proyecto.

Nova

Swift

Keystone

Horizon

Glance

Cinder

Heat

Neutron

Solucionando el problema

Introducción

Conectar el Servicio de Identidad de OpenStack, u OpenStack Identity Service en inglés, para que consiga autenticarse contra un Active Directory no es una tarea difícil en sí misma. De hecho, a día de hoy ya existen varias empresas u organizaciones que tienen una solución funcionando en producción parecida a la aquí propuesta. El problema principal reside en que no todas las soluciones documentadas en Internet para abordar este problema funcionan y/o hacen referencia a un escenario similar al aquí encontrado. En la mayoría de los casos se necesitan unos amplios conocimientos del funcionamiento interno de Active Directory así como del código y forma de funcionar de OpenStack para entender realmente qué puede estar pasado en caso de problemas.

Como ya se explicó en el apartado de objetivos del proyecto, se pretende enlazar un Active Directory de Microsoft con un OpenStack. De esta forma usuarios, proyectos y roles, así como sus respectivas credenciales, vivirán en el Directorio Activo.

Sobre el papel esta funcionalidad vino implementada por primera vez en OpenStack Havana (versión 8), lanzada en la segunda mitad del año 2013. Quiero remarcar el hecho de “sobre el papel” porque a pesar de que oficialmente el código nos proporciona la funcionalidad, para ponerlo a funcionar es necesario ser consciente de muchas de sus limitaciones y estar dispuesto a enfrentarse a funcionalidad incompleta y errores no documentados. Afortunadamente para este desarrollo muchos de estos problemas se solucionaron en la siguiente versión, IceHouse, la cual se está utilizando como base de este proyecto.

En mi caso concreto algunos de estos errores anteriormente comentados me dieron muchos dolores de cabeza dado que en muchos casos arreglar uno únicamente servía para que aparecieran otros derivados o incluso en muchos casos, no derivados.

La parte positiva es que la tratarse de un proyecto abierto con una comunidad muy extensa siempre me resultó muy sencillo reportar los errores, los funcionamientos no esperados o simplemente no implementados. En algunos casos mis pequeños parches al código, comentarios y pruebas realizadas también resultaron útiles a la comunidad.

Manos a la obra

Para llevar a cabo la integración podemos optar por varias formas de abordarlo según como tengamos diseñada nuestra infraestructura. Aunque leyendo casos de uso de empresas y organizaciones que actualmente tiene un sistema parecido en producción podemos encontrar dos.

La primera opción es posiblemente la más obvia, ya que estamos utilizando un Active Directory como backend de identidad, almacenaremos en éste todos los actores necesarios para hacerla posible. Es decir, roles, proyectos y usuarios vivirán en el Directorio Activo y Keystone, el servicio de identidad de OpenStack será el encargado de leer y consultar esa información para llevar a cabo la autenticación.

La segunda de ellas consiste en almacenar los roles y proyectos (tenants) en una base de datos SQL a la que accederá Keystone, mientras que los usuarios se guardarán de el Active Directory. De esta forma la configuración particular de OpenStack se almacenara en OpenStack y la relacionada con usuarios se quedara en el Directorio Activo. La mayoría de las empresas están utilizando este enfoque puesto que es sencillo de manejar y sólo se requiere acceso de lectura en el Directorio Activo. además en un entorno empresarial, normalmente el grupo destinado a administrar todo lo relacionado con OpenStack y maquinas virtuales suele ser distinto al que se encarga del Directorio Activo, por lo que mantenerlo separado es una buena opción.

Al tratarse de una prueba de concepto, en este proyecto se han optado por poner en marcha ambas formas anteriormente explicadas. Aunque evidentemente sólo una de ellas podrá estar activa al mismo tiempo.

Autenticación en OpenStack

En el proceso de autenticación participan la terna, usuario, rol y proyecto (tenant). Es decir, un usuario, pertenecerá a un proyecto en el que tendrá un rol asignado. Para que OpenStack funcione correctamente, dicha terna deberá estar bien configurada.

El encargado de verificar dicha configuración es el módulo Keystone, el servicio de Identidad de Openstack. Toda esta configuración reside en el archivo de configuración de éste, llamado keystone.conf en el que para hacerlo funcionar habrá que editar varias de sus líneas.

Creación de usuarios y tenants (proyectos)

En OpenStack cada usuario puede estar asignado a uno o varios proyectos en los que podrá trabajar con las distintas instancias de las máquinas virtuales. Estos proyectos son conocidos dentro del entorno Openstack como tenants.

Cada usuario o grupos de usuarios podrán tener sus propios tenants particulares en donde desempeñarán su trabajo de forma privada y uno o varios compartidos en donde se pondría en conocimiento su trabajo con otros usuarios acreditados.

Crear la estructura necesaria en AD

Editar keystone .conf

Añadir usuarios pertenecientes al proyecto interno **services** con sus correspondientes contraseñas en el Directorio Activo

Una vez realizado el cambio anterior ya será posible autenticarnos contra el Directorio Activo de Microsoft pero aún no será posible utilizar OpenStack. Para comenzar a utilizarlo es necesario añadir a nuestro backend de LDAP aquellos usuarios claves que hacen uso interno de OpenStack. El nombre de estos usuarios se corresponden con el nombre de los servicios que representan. Dependiendo de nuestra instalación de OpenStack deberemos añadir unos u otros usuarios. Para esta configuración concreta añadiremos los siguientes usuarios a nuestro Directorio Activo: nova, cinder y glance

En este sentido es **muy importante** que la contraseña de dichos usuarios concuerde con la especificada en los archivos de configuración de OpenStack, en caso contrario el sistema nos devolverá un error 403 para aquellos comandos relacionados con el usuario.

```
[root@lenuage ~(keystone_myadmin)]# nova image-list
ERROR: This server could not verify that you are authorized to access the
document you requested. Either you supplied the wrong credentials (e.g., bad
password), or your browser does not understand how to supply the credentials
required. (HTTP 401) (Request-ID: req-fdd1fd08-501a-4ef6-9b9c-4ffd8ccf4b54)
```

Es decir, debemos revisar el contenido de los siguientes archivos y verificar el valor de la propiedad "admin_password".

- /etc/nova/nova.conf
- /etc/cinder/cinder.conf

- /etc/swift/proxy-server.conf
- /etc/glance/glance-api.conf

Ese valor será el la contraseña que debemos configurar para cada usuario correspondiente en nuestro Directorio Activo

Podemos utilizar el la orden grep para buscar fácilmente todas las contraseñas contenidas en los archivos de configuración de estos servicios mediante el siguiente comando

```
grep -R "cadena a buscar" <directorio>
```

En nuestro caso concreto buscaremos la cadena "admin_password" dentro del directorio /etc que es donde se encuentran todos los archivos de configuración de OpenStack:

```
[root@lenuage ~(keystone_myadmin)]# grep -R "admin_password" /etc/
/etc/nova/nova.conf:#neutron_admin_password=<None>
/etc/nova/nova.conf:#admin_password=%SERVICE_PASSWORD%
/etc/nova/nova.conf:admin_password=d6487810e1874f27
/etc/swift/proxy-server.conf:admin_password = c20a92650ce14f99
/etc/cinder/api-paste.ini:admin_password=d9899d3442004fa3
/etc/cinder/cinder.conf.rpmnew:# `admin_user` and `admin_password` instead.
(string value)
/etc/cinder/cinder.conf.rpmnew:#admin_password=<None>
/etc/cinder/cinder.conf:#admin_password=<None>
/etc/glance/glance-api.conf:#admin_password=%SERVICE_PASSWORD%
/etc/glance/glance-api.conf:#admin_password=%SERVICE_PASSWORD%
/etc/glance/glance-api.conf:admin_password=8cfa0b752b884b0d
/etc/glance/glance-scrubber.conf:# admin_password = %SERVICE_PASSWORD%
/etc/glance/glance-cache.conf:# admin_password = %SERVICE_PASSWORD%
/etc/glance/glance-cache.conf:admin_password = 8cfa0b752b884b0d
/etc/glance/glance-registry.conf:#admin_password=%SERVICE_PASSWORD%
/etc/glance/glance-registry.conf:admin_password=8cfa0b752b884b0d
grep: /etc/httpd/run/wsgi.12497.1.1.sock: No such device or address
```

Revisando la salida del comando anterior nos sera muy fácil identificar las contraseñas de las cuentas

Una vez que tenemos las contraseñas de las cuentas de servicio de OpenStack, será necesario crear dichas cuentas en nuestro Active Directory. Para ello, dentro de la unidad organizativa en la que guardamos los usuarios, añadimos los usuarios de OpenStack:

Es importante marcar la opción que evita el cambio de contraseña a posteriori por parte del usuario así como la opción que evita que la contraseña caduque. De estar forma nos evitaremos posibles dolores de cabeza relacionados con este tema en el futuro.

Añadir roles

Una vez hecho lo anterior ya tendremos configurados los usuarios mínimos que OpenStack necesita para su correcto funcionamiento en nuestro Directorio Activo. A pesar de esto si comenzamos a utilizar el usuario 'admin' recién creado para por ejemplo mostrar la lista de proyectos obtendremos una desagradable sorpresa:

```
[root@lenuage ~(keystone_admin)]# source keystone_admin
[root@lenuage ~(keystone_admin)]# keystone tenant-list
Invalid user / password (HTTP 401)
[root@lenuage ~(keystone_admin)]#
```

Esto es debido a que aunque el usuario 'admin' sea un usuario reconocido por el servicio de identidad de OpenStack, éste no tiene ningún tipo de permisos sobre los proyectos existentes.

Para solucionarlo debemos añadir al usuario 'admin' al proyecto 'admin' con el rol 'admin'. De esta forma configuraremos nuestra terna y realmente daremos permisos de administrador a nuestro usuario 'admin'.

```
[root@lenuage ~]# keystone user-list
+-----+-----+-----+-----+
| id    | name  | enabled | email |
+-----+-----+-----+-----+
| admin | admin | True    |       |
| cinder| cinder| True    |       |
```

```

| glance | glance | True | |
| nova   | nova   | True | |
+-----+-----+-----+-----+
[root@lenuage ~]# keystone role-list
+-----+-----+-----+-----+
|          id          | name |
+-----+-----+-----+-----+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| fb354cf6d767441d8f53589464d90b54 | admin    |
+-----+-----+-----+-----+
[root@lenuage ~]# keystone tenant-list
+-----+-----+-----+-----+
|          id          | name   | enabled |
+-----+-----+-----+-----+
| 4e6c1002c09d41c595a55fc2a040681f | admin  | True    |
| 54244f8ede8944e4ba0fa8aa7cdd5798 | services | True    |
+-----+-----+-----+-----+
[root@lenuage ~]# keystone user-role-add --user admin --role
fb354cf6d767441d8f53589464d90b54 --tenant 4e6c1002c09d41c595a55fc2a040681f
[root@lenuage ~]#

```

Una vez realizados los pasos anteriores, configurado el usuario admin correctamente, ya tendremos acceso. Para probarlo ejecutamos lo siguiente:

```

[root@lenuage ~(keystone_admin)]# keystone tenant-list
+-----+-----+-----+-----+
|          id          | name   | enabled |
+-----+-----+-----+-----+
| 4e6c1002c09d41c595a55fc2a040681f | admin  | True    |
| 54244f8ede8944e4ba0fa8aa7cdd5798 | services | True    |
+-----+-----+-----+-----+
[root@lenuage ~(keystone_admin)]#

```

Como en nuestro caso particular queremos que el usuario admin sea un usuario administrador clásico, es decir que pueda acceder a todos los servicios de este y manejarlos y configurarlos a su antojo, tendrá el role admin para todos los proyectos.

Finalmente será necesario realizar lo anteriormente explicado para nuestros usuarios internos de OpenStack: nova, cinder y glance.

Nova

```

[root@lenuage ~(keystone_admin)]# nova list
ERROR: Unauthorized (HTTP 401)

```

Este ejemplo es idéntico al anterior con la salvedad de que deberos añadir el usuario nova en el proyecto services y no al de admin.

```

[root@lenuage ~(keystone_admin)]# keystone user-role-add --user nova --role
fb354cf6d767441d8f53589464d90b54 --tenant 54244f8ede8944e4ba0fa8aa7cdd5798

```

```
[root@lenuage ~(keystone_admin)]# nova list
+-----+-----+-----+-----+-----+-----+
| ID | Name | Status | Task State | Power State | Networks |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
[root@lenuage ~(keystone_admin)]#
```

Glance

Idéntico al caso de nova pero para el usuario de glance

```
[root@lenuage ~(keystone_admin)]# glance image-list
Request returned failure status.
Invalid OpenStack Identity credentials.

[root@lenuage ~(keystone_admin)]# keystone user-role-add --user glance --role
fb354cf6d767441d8f53589464d90b54 --tenant 54244f8ede8944e4ba0fa8aa7cdd5798

[root@lenuage ~(keystone_admin)]# glance image-list
+-----+-----+-----+-----+-----+-----+
| ID | Name | Disk Format | Container Format | Size | Status |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
[root@lenuage ~(keystone_admin)]
```

Cinder

Igual para el usuario de cinder.

```
[root@lenuage ~(keystone_admin)]# cinder list
ERROR: Unauthorized (HTTP 401)

[root@lenuage ~(keystone_admin)]# keystone user-role-add --user cinder --role
fb354cf6d767441d8f53589464d90b54 --tenant 54244f8ede8944e4ba0fa8aa7cdd5798

[root@lenuage ~(keystone_admin)]# cinder list
+-----+-----+-----+-----+-----+-----+-----+
| ID | Status | Display Name | Size | Volume Type | Bootable | Attached to |
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
[root@lenuage ~(keystone_admin)]#
```

Creación la primera máquina en OpenStack

A continuación explicaremos como crear la primera maquina virtual en nuestro recién instalado entorno de virtualización OpenStack. Para llevar a cabo esta tarea utilizaremos, tres de los componentes instalados anteriormente, glance y nova a través de sus command line interface (cli) y keystone indirectamente para temas de identidad.

Descarga de la imagen

En primer lugar deberemos descargar el fichero de la imagen que queremos integrar en OpenStack para una vez descargado proceder a integrarlo.

Para este ejemplo utilizaré una imagen de un sistema GNU/Linux llamada "custom-built Linux distribution CirrOS". CirrOS es una imagen preparada ser desplegada y que destaca por su extrema ligereza. CirrOS es una distribución Linux simplificada y contenida en menos de 10 megas. Además su instalación por defecto viene con el puerto 22 abierto y con un

usuario no administrador creado, para que conectarnos remotamente por ssh no sea ningún problema.

Para descargarla utilizaremos el comando wget:

```
[root@lenuage images(keystone_admin)]# wget --no-check-certificate
https://download.cirros-cloud.net/0.3.3/cirros-0.3.3-x86_64-disk.img
--2014-12-11 18:26:02-- https://download.cirros-cloud.net/0.3.3/cirros-0.3.3-
x86_64-disk.img
Resolving download.cirros-cloud.net... 69.163.241.114
Connecting to download.cirros-cloud.net|69.163.241.114|:443... connected.
WARNING: cannot verify download.cirros-cloud.net's certificate, issued by
`/CN=download.cirros-cloud.net':
  Self-signed certificate encountered.
HTTP request sent, awaiting response... 302 Found
Location: http://cdn.download.cirros-cloud.net/0.3.3/cirros-0.3.3-x86_64-
disk.img [following]
--2014-12-11 18:26:04-- http://cdn.download.cirros-cloud.net/0.3.3/cirros-
0.3.3-x86_64-disk.img
Resolving cdn.download.cirros-cloud.net... 2.22.63.66, 2.22.63.56,
2a02:26f0:64::170e:5d39, ...
Connecting to cdn.download.cirros-cloud.net|2.22.63.66|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13200896 (13M) [application/octet-stream]
Saving to: `cirros-0.3.3-x86_64-disk.img'

100%
[=====
=====
=====>] 13,200,896  9.62M/s   in 1.3s

2014-12-11 18:26:05 (9.62 MB/s) - `cirros-0.3.3-x86_64-disk.img' saved
[13200896/13200896]

[root@lenuage images(keystone_admin)]#
```

Una vez descargada la imagen pasaremos a importarla al servicio de imágenes de OpenStack llamado glance. Para ello usaremos el comando glance image-create que nos proporcione la CLI de OpenStack.

```
[root@lenuage images(keystone_admin)]# glance image-create --name="CirrOS 0.3.3"
--disk-format=qcow2 --container-format=bare --is-public=true --file cirros-
0.3.3-x86_64-disk.img
+-----+-----+
| Property          | Value                                     |
+-----+-----+
| checksum          | 133eae9fb1c98f45894a4e60d8736619       |
| container_format  | bare                                    |
| created_at        | 2014-12-11T17:40:32                     |
| deleted           | False                                   |
| deleted_at        | None                                    |
| disk_format       | qcow2                                   |
| id                | 71fb6cbf-1f0e-4567-bf63-962e02145fbe   |
| is_public         | True                                    |
```

```
| min_disk      | 0 |
| min_ram       | 0 |
| name          | CirrOS 0.3.3 |
| owner         | 4e6c1002c09d41c595a55fc2a040681f |
| protected     | False |
| size          | 13200896 |
| status        | active |
| updated_at    | 2014-12-11T17:40:32 |
| virtual_size  | None |
+-----+
[root@lenuage images(keystone_admin)]#
```

Para verificar que la imagen ha sido correctamente añadida podemos listar todas aquellas que tenemos disponibles con el comando image-list de glance:

```
[root@lenuage images(keystone_admin)]# glance image-list --human-readable
+-----+-----+-----+-----+-----+-----+
| ID          | Name          | Disk Format | Container Format | Size      | Status |
+-----+-----+-----+-----+-----+-----+
| e02145fbe   | CirrOS 0.3.3 | qcow2       | bare             | 12.6MB    | active |
| 45d522b25   | Ubuntu       | qcow2       | bare             | 244.3MB   | active |
+-----+-----+-----+-----+-----+-----+
[root@lenuage images(keystone_admin)]#
```

(ids editados para mantener el formato de la documentación)

```
[root@lenuage images(keystone_admin)]# nova image-list
+-----+-----+-----+-----+-----+
| ID          | Name          | Status | Server |
+-----+-----+-----+-----+-----+
| 71fb6cbf-1f0e-4567-bf63-962e02145fbe | CirrOS 0.3.3 | ACTIVE |        |
| 8dcc802f-b892-40b2-8589-08c45d522b25 | Ubuntu       | ACTIVE |        |
+-----+-----+-----+-----+-----+
[root@lenuage images(keystone_admin)]#
```

Modificaciones cosméticas OpenStack

Opensource, fuertemente configurable. Acceso al código, CSS, imágenes.

Integración OwnCloud Draw.io

Descargamos el complemento

.zip así que hay que instala unzip

Lo descomprimos en /owncloud/apps

chmod -R 755 /var/www/owncloud/apps/files_drawio

```
root@alexandria:~# ll /var/www/owncloud/apps/
total 100
drwxr-xr-x 11 root root 4096 Dec 10 19:52 activity
drwxr-xr-x  6 root root 4096 Dec 10 19:52 admin_dependencies_chk
drwxr-xr-x 12 root root 4096 Dec 10 19:52 bookmarks
drwxr-xr-x 12 root root 4096 Dec 10 19:52 calendar
drwxr-xr-x 11 root root 4096 Dec 10 19:52 contacts
```

```
drwxr-xr-x 11 root root 4096 Dec 10 19:52 documents
drwxr-xr-x 10 root root 4096 Dec 10 19:52 external
drwxr-xr-x 11 root root 4096 Dec 10 19:52 files
drwxr-xr-x  5 root root 4096 Jul 15 2014 files_drawio
drwxr-xr-x 13 root root 4096 Dec 10 19:52 files_encryption
drwxr-xr-x 11 root root 4096 Dec 10 19:52 files_external
drwxr-xr-x  6 root root 4096 Dec 10 19:52 files_pdfviewer
drwxr-xr-x 10 root root 4096 Dec 10 19:52 files_sharing
drwxr-xr-x  7 root root 4096 Dec 10 19:52 files_texteditor
drwxr-xr-x 10 root root 4096 Dec 10 19:52 files_trashbin
drwxr-xr-x  9 root root 4096 Dec 10 19:52 files_versions
drwxr-xr-x  7 root root 4096 Dec 10 19:52 files_videoviewer
drwxr-xr-x  9 root root 4096 Dec 10 19:52 firstrunwizard
drwxr-xr-x  9 root root 4096 Dec 10 19:52 gallery
drwxr-xr-x  9 root root 4096 Dec 10 19:52 search_lucene
drwxr-xr-x 11 root root 4096 Dec 10 19:52 templateeditor
drwxr-xr-x  9 root root 4096 Dec 10 19:52 updater
drwxr-xr-x  5 root root 4096 Dec 10 19:52 user_external
drwxr-xr-x 11 root root 4096 Dec 10 19:52 user_ldap
drwxr-xr-x  5 root root 4096 Dec 10 19:52 user_webdavauth
```

Reiniciamos el servicio

```
root@alexandria:~# service apache2 restart
[ ok ] Restarting web server: apache2 ... waiting .
root@alexandria:~#
```

Activamos el plugin <https://156.35.95.20:4433/index.php/settings/apps?installed>

Problemas encontrados

Esta documentación no podría estar completa sin hacer referencia a aquellos problemas encontrados durante el desarrollo de este proyecto. La realización de este proyecto no ha sido en absoluto un camino de rosas y durante muchas etapas he tenido que afrontar con lo que inicialmente no contaba y de los que apenas tenía información o medios sobre como solucionarlos.

Por ello, me ha parecido muy oportuno documentarlos en este apartado para así a parte de dejar constancia de ellos que el futuro lector de esta documentación pueda utilizarlo para su provecho y evitar así tropezar con algunas de las piedras que me he encontrado en el camino.

Heartbleed

HeartBleed el nombre que se le ha dado al bug considerado por muchos como la mayor amenaza de seguridad de la era de Internet. En abril de 2014, algunos empleados de Google Security encontraron un importante fallo de seguridad en una de las librerías OpenSSL relacionadas con el cifrado de archivos. Debido a su gran importancia, vamos a dedicarle algunas líneas para explicar que ocurrió y sus consecuencias para los millones de dispositivos que diariamente utilizan OpenSSL.

Heartbleed

El 17 de Abril de 2014 se hizo de dominio publico el bug CVE-2014-0160 que afectaba a todos los dispositivos que utilizasen una versión no parcheada de ciertas versiones de la librería OpenSSL. Las consecuencias de este problema son terriblemente graves pudiendo dejar en entre dicho la configuración de seguridad más robusta, fiable y probada del planeta. Un atacante malicioso podría obtener datos privados y sensibles de nuestro sistema simplemente utilizando la función heartbleed de la librería OpenSSL que verifica si la conexión con un servidor sigue activa. A diferencia de otras vulnerabilidades encontradas hasta la fecha, para explotar esta falla no es necesario prácticamente nada y por desgracia no deja ningún tipo de registro en los ficheros de registros del sistema.

La función heartbleed se encarga de mantener abierta una conexión entre un cliente y un servidor. De esta forma, cuando el cliente quiere verificar si el servidor sigue ahí escuchando, lo único que debe hacer una llamada a la función hearthbleed del servidor e indicarle una palabra y su longitud. Si la conexión se mantiene activa el servidor contestará con esa misma palabra. El problema surge cuando la palabra enviada y su longitud no coinciden. Pongamos un ejemplo práctico para entender bien el problema.

Para verificar si un servidor mantiene la conexión, desde el cliente le enviamos la palabra "vivo" y le decimos que su longitud es 4. De momento nada extraño, el servidor responde y verificamos que la conexión sigue activa. Hagamos otra prueba con la misma cadena de texto pero esta vez vamos a indicarle que la logitud de dicha cadena sera de 512. Al estar programado en C devuelve contenido en memoria que no te corresponde. En la mayoría de las ocasiones devolverá contenido basura, pero un atacante podría recopilar suficiente información hasta adquirir algo relevante, como ficheros de

configuración o incluso claves privadas del servidor.

```
OpenSSL 1.0.1 through 1.0.1f (inclusive) are vulnerable
OpenSSL 1.0.1g is NOT vulnerable
OpenSSL 1.0.0 branch is NOT vulnerable
OpenSSL 0.9.8 branch is NOT vulnerable
```

Gracias a la excelente trazabilidad de los proyectos de software libre, es sabido que el bug fue introducido en diciembre de 2011 y ha estado presente en la rama oficial 1.0.1 de la librería desde marzo de 2012 hasta que casi dos años mas tarde en abril de 2014 fue corregida la vulnerabilidad.

Solución

El 7 de abril de 2014, mismo día de hacerse publico el agujero de seguridad, apareció una nueva actualización de la librería que solucionaba la vulnerabilidad.

Desde en ese mismo momento en el que fue hecho público, se recomienda encarecidamente cambiar las contraseñas de aquellos servicios vulnerables que hayan podido estar expuestos, así como generar nuevos certificados para cada uno de los servidores que utilizasen las versiones vulnerables de esta librería de cifrado.

En este caso concreto, una vez publicada la vulnerabilidad se procedio a la actualización de openssl en cada una de las maquinas Linux que forma parte de este proyecto. Así mismo y a pesar del bajo indice de exposición de las máquinas de este proyecto, por precaución se cambio la contraseña de acceso de aquellos usuarios expuestos.

SELinux activado en las máquinas RH.

SELinux es el sistema de control de acceso dinámico que viene implementado en RedHat Enterprise Linux (RHEL) y derivadas (CentOS, Fedora, Scientific Linux...). Originalmente desarrollado por la Agencia de Seguridad Nacional de los Estados Unidos de America (NSA) es un sistema de control de acceso muy seguro y robusto. Su configuración está basada en etiquetas más allá del antiguo -y hoy en día poco fiable- sistema de permisos de archivo. De esta manera independientemente de los permisos, si el fichero, archivo o configuración no está etiquetado correctamente su funcionamiento podría no ser el esperado. De este etiquetado se encarga automáticamente SELinux aunque es responsabilidad del administrador de la máquina modificar ciertos comportamientos o configuraciones del mismo para adaptarlo al uso que queremos hacer del sistema. Algunas configuraciones no se estaban aplicando correctamente debido a SELinux. Openstack proporciona un paquete denominado *openstack-selinux* que nos ayuda con algunas de las configuraciones más típicas que pueden

Falta de espacio en el servidor OpenStack

Una vez creado y configurado el servidor de OpenStack uno de los problemas encontrados fue la falta de espacio en disco tras la creación de las primeras instancias de máquinas virtuales. A pesar de haber asignado más de 500Gb de disco para tal fin, OpenStack sólo estaba reconociendo menos de 20Gb para alojar las nuevas instancias. Tras analizar

minuciosamente qué podría estar pasando me di cuenta de que por defecto Openstack sólo trabaja en la partición raíz del sistema por lo que si quería aprovechar todo el espacio disponible debía extender el tamaño de dicha partición.

Extendiendo el tamaño de raíz /

```
[root@lenuage ~(keystone_admin)]# df -H
Filesystem                Size      Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root 19G    6.3G    12G   36% /
tmpfs                      34G         0    34G    0% /dev/shm
/dev/xvda1                 508M     81M   402M   17% /boot
```

```
[root@lenuage ~(keystone_admin)]# fdisk -l
Disk /dev/xvda: 536.9 GB, 536870912000 bytes
255 heads, 63 sectors/track, 65270 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0007bc25

    Device Boot      Start         End      Blocks   Id  System
/dev/xvda1    *           1           64       512000    83  Linux
Partition 1 does not end on cylinder boundary.
/dev/xvda2           64        2611      20458496    8e  Linux LVM

Disk /dev/xvdd: 119 MB, 119197696 bytes
255 heads, 63 sectors/track, 14 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/mapper/VolGroup-lv_root: 18.8 GB, 18798870528 bytes
255 heads, 63 sectors/track, 2285 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/mapper/VolGroup-lv_swap: 2147 MB, 2147483648 bytes
255 heads, 63 sectors/track, 261 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

rpm -Uvh http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

Para realizarlo utilizaremos la herramienta *growpart*. Como dicha herramienta no está

instalada y dependiendo de cada distribución la podremos encontrar integrada en un paquete u otro, usaremos la opción `whatprovides` del gestor de paquetes `yum` para saber que paquete nos proporciona dicha funcionalidad.

```
yum whatprovides */nombredelpaquete
```

```
[root@lenuage ~(keystone_admin)]# yum whatprovides */growpart
Loaded plugins: priorities, security
272 packages excluded due to repository priority protections
epel/filelists_db | 8.4 MB 00:01
openstack-havana/filelists_db | 1.5 MB 00:02
puppetlabs-deps/filelists_db | 182 kB 00:00
puppetlabs-products/filelists_db | 917 kB 00:00
sl-security/filelists_db | 1.4 MB 00:08
sl6x-security/filelists_db | 1.4 MB 00:08
cloud-utils-growpart-0.27-10.el6.x86_64 : Script for growing a partition
Repo : epel
Matched from:
Filename : /usr/bin/growpart

[root@lenuage ~(keystone_admin)]#
```

Como podemos ver en la información anterior, **cloud-utils-growpart** es el paquete que contiene dicha funcionalidad para Scientific Linux.

Conocida esta información procedemos con su instalación.

```
[root@lenuage ~(keystone_admin)]# yum install cloud-utils-growpart -y
```

```
[root@lenuage ~(keystone_admin)]# growpart /dev/xvda 2
CHANGED: partition=2 start=1026048 old: size=40916992 end=41943040 new:
size=1047536502,end=1048562550
```

Después de ejecutar el anterior comando la máquina debe ser reiniciada. Para ello utilizaremos el comando `reboot`.

```
[root@lenuage ~(keystone_admin)]# reboot
```

Una vez reniciada, nos logueamos y ejecutamos los siguientes comandos.

```
[root@lenuage ~(keystone_admin)]# pvresize /dev/xvda2
Physical volume "/dev/xvda2" changed
1 physical volume(s) resized / 0 physical volume(s) not resized
```

```
[root@lenuage ~(keystone_admin)]# lvextend -l +100%FREE /dev/mapper/VolGroup-
lv_root
Extending logical volume lv_root to 497.50 GiB
Logical volume lv_root successfully resized
```

```
[root@lenuage ~(keystone_admin)]# resize2fs /dev/mapper/VolGroup-lv_root
```

```
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/mapper/VolGroup-lv_root is mounted on /; on-line resizing
required
old desc_blocks = 2, new_desc_blocks = 32
Performing an on-line resize of /dev/mapper/VolGroup-lv_root to 130416640 (4k)
blocks.
The filesystem on /dev/mapper/VolGroup-lv_root is now 130416640 blocks long.
```

Como se puede apreciar el cambio ha sido un éxito.

```
[root@lenuage ~]# df -H
Filesystem                Size      Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root 526G    6.4G   493G    2% /
tmpfs                      34G         0    34G    0% /dev/shm
/dev/xvda1                 508M     81M   402M   17% /boot
```

Error al borrar OU desde Active Directory User and Computers

<http://technet.microsoft.com/en-us/library/cc736842%28v=ws.10%29.aspx>

Password de Microsoft

Caídas continuas de la máquina de desarrollo

Modificar el archivo bower y actualizar la versión de angular

RDO Openstack versiones cada 6 meses. Mucho de los parches (en mi caso los relacionados con Keystone Ldap) no incluidos en RedHat RDO. Por lo que tuve que realizar los backports directamente desde los repositorio de launchpad.

Problemas con xapi

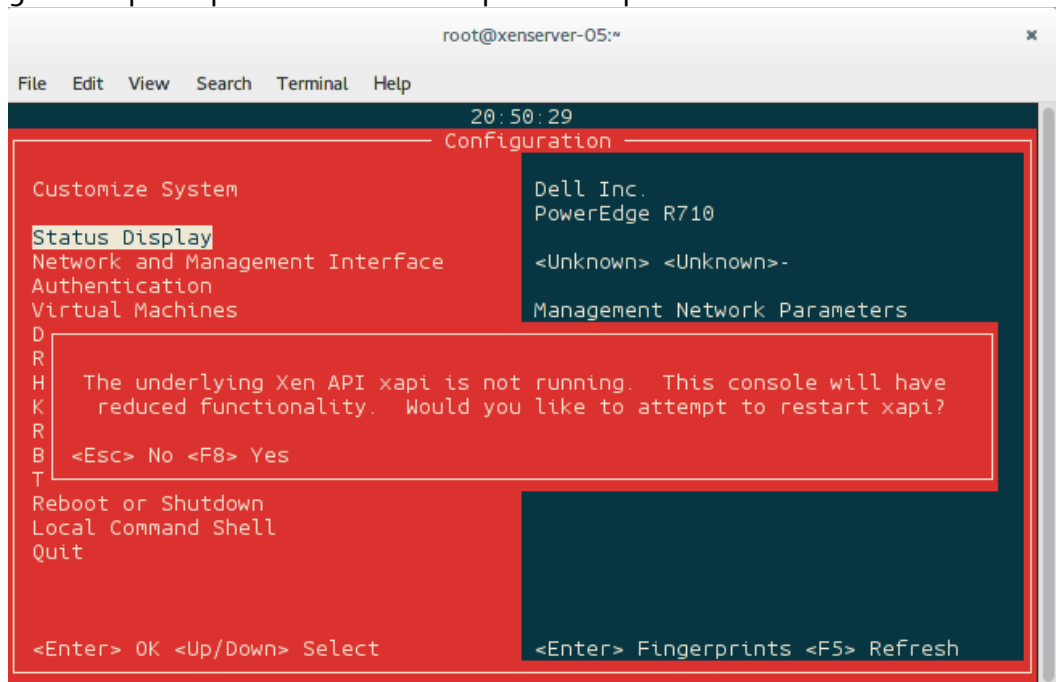
Tras varias semanas con los servidores funcionando correctamente, comencé a percibir de forma puntual problemas que propiciaban cortes de conexión intermitentes con las máquinas. Este comportamiento me resultó extraño dado que las máquinas estaban trabajando de una forma más o menos estable y no se habían producido cambios en su configuración importantes.

Para ser sincero, al principio no le di demasiada importancia y achiqué este comportamiento a un problema más en las comunicaciones como algún otro de los ya sufridos en el pasado. Con el paso de los días estos problemas se hicieron más frecuentes, hasta llegando a un punto en el que se volvieron bastante molestos dejando en muchos casos las máquinas colgadas. Desde el más absoluto desconocimiento lo primero que pensé fue en reiniciar las máquinas para ver si se podría deber a algún problema con algún servicio y quizá un reinicio de cero devolvería al sistema a su estado habitual.

Desgraciadamente, estos reinicios, solo hicieron más que agravar el problema derivando a un problema mucho mayor. Tras el reinicio del servidor 'host' todas mis máquinas virtuales basadas en Linux dejaron de poder ser iniciadas.

El error mostrado desde XenCenter no era demasiado descriptivo en sí mismo y simplemente mostraba un error al inicio del que le era imposible recuperarse. Decidí entonces revisar más a fondo los registros de errores, accediendo al sistema hipervisor desde la propia consola ordenes.

Una vez dentro, invoqué a el comando para abrir la consola de xen, pero la única información que me proporcionaba era que el servicio de xapi no estaba siendo ejecutado y como opción me ofrecía reiniciarlo. Parece que había descubierto algo más sobre el origen del problema y para intentar solucionarlo debería. al menos. devolver a un estado consistente al servicio xapi. Para ellos intenté reiniciarlo en varias ocasiones, sin éxito. Había algo más que impedía solucionar el problema por los cauces normales.



Una vez descartado el inicio manual del servicio decidí sumergirme en los registros de errores almacenados en /var/log/. Revisando éstos sin demasiadas pistas, la única referencia que encontraba era un error relacionado con xenstorage service que no podría arrancar. Poca información y ademas bastante inconexa teniendo en cuenta aquellos datos con los que ya contaba.

Después de un tiempo debuggeando las causas del problema y apoyándome en la comunidad de usuarios de Xen conseguí identificar el problema raíz que estaba desencadenando todos estos comportamientos tan extraños.

Aparentemente el problema residía en que el directorio raíz '/' de xenserver05 estaba completamente lleno y eso impedía crear una serie de archivos de log al iniciar las máquinas lo que hacia abortar el arranque de éstas.

```
root@xenserver-05:~  
File Edit View Search Terminal Help  
Quitting...  
[root@xenserver-05 ~]# df -H  
Filesystem      Size  Used Avail Use% Mounted on  
/dev/sda1        4.3G  4.3G   0 100% /  
none            391M   21k  391M   1% /dev/shm  
/opt/xensource/packages/iso/XenCenter.iso  
54M   54M   0 100% /var/xen/xl-install  
[root@xenserver-05 ~]#
```

Parecía que el problema estaba identificado, pero ahora había que liberar algo de espacio. Para ello decidí borrar todos los archivos localizados el directorio /tmp así como la gran mayoría de los logs relacionados con xenserver (miles de ellos en este punto). También se borraron las imágenes no usadas y algunos de los archivos utilizados para actualizar el sistema que se encontraban en desuso.

Una vez liberado el espacio reinicie los servicios relacionados con xen y parece que todo volvió a la calma. El error de xapi dejó de aparecer y pude iniciar de nuevo todas mis máquinas virtuales.

A pesar de que la solución es sencilla ha sido un problema difícil de diagnosticar y con unas consecuencias demoledoras para mi entorno, dado que me impedía iniciar las maquinas virtuales.

Restauración de instantáneas de las maquinas

A veces tras una mala configuración el sistema inconsistente, lo mas rápido era realizar una restauración aunque no siempre funcionó. En algunos casos tuve que restaurar la maquinas a partir de sus imagen.

Agrupacion de servicios en una sola maquinas

Hardware excelente sobre el papel

Ips a tutiplen

Sin nucleos :(

Ampliaciones

Actualización a OpenStack Juno

Como se ha comentado anteriormente en este documento el ritmo de actualización de OpenStack es bastante rápido, con una nueva versión disponible al público general cada 6 meses. La versión utilizada en este proyecto se corresponde con la versión lanzada en mayo 2014 de nombre clave IceHouse. Durante el desarrollo de este proyecto fue lanzada la décima versión de OpenStack "Juno". Como en otras actualizaciones esta versión incorpora numerosas mejoras y algún que otro componente nuevo.

Una posible ampliación de este proyecto podría ser la de actualizar OpenStack a su versión más reciente, incorporar así las nuevas mejoras y características que nos brinda el nuevo desarrollo y solucionar los posibles problemas de actualizar de versión un sistema en producción.

Diseño de un entorno multinodo

En este desarrollo y en respuesta a la prueba de concepto que pretende ser, todos los elementos de Openstack fueron instalados en un mismo servidor. En un entorno de producción real, donde rendimiento, escalabilidad y tolerancia a desastres son muy importantes sería conveniente separar al menos algunos de los los módulos clave, con el fin de mejorar nuestro diseño y hacerlo mas robusto, escalable y resistente a fallos.

De esta forma podríamos optar por crear diversos nodos "compute" una dedicado unicamente a temas relacionados con la conductividad de red con Neutron y uno dedicado a las imágenes Cinder bla bla

En realidad, el grado de dificultad de este enfoque no es más complicado que el mononodo aquí implementado. Simplemente se tratará de tener claras las condiciones hardware de las máquinas que en la que se van a instalar cada nodo.

Para esta prueba de concepto y debido a las restricciones hardware se ha optado por mantener una única instalación corriendo con todos los servicios necesarios para ejecutar un Openstack funcional.

Añadir mas computes nodes

Otra ampliación posible sería la de añadir mas computes nodes a nuestro proyecto. Los computes nodes son aquellas maquinas que hospedarán a las instancias de la maquinas virtuales. Dependiendo de las características hardware del compute host y del tamaño de las maquinas a hospedar, éste podrá albergar un mayor o menos numero de ellas. Agregar más compute nodes no tiene demasiado misterio en si mismo y para ello será necesario realizar una serie de pasos y añadir una serie de configuraciones

En este caso no se ha podido realizar por la falta de servidores físicos un mayor número de servidores físicos disponibles