ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Máster

Aplicación de LDAP como método de autenticación en entornos de Cloud Privada

(Application of LDAP as authentication method in Private Cloud environments)

Para acceder al Título de

Máster Universitario en Ingeniería de Telecomunicación

Autor: Alejandro Abascal Crespo

10 - 2017

Resumen 1
Abstract 2
1. Introducción y objetivos 3
1.1. Introducción 3
1.2. Motivación y objetivos 3
1.3. Organización del documento 3
2. Conceptos teóricos 5
2.1. Servicios de computación en la nube 5
2.2. OpenStack 7
2.2.1. DevStack 11
2.2.2. Mirantis OpenStack 12
2.2.3. Packstack 14
2.3. XenServer 15
2.4. LDAP 17
2.5. TLS 20
3. Aspectos prácticos 21
3.1. Definición del problema 21
3.2. Requerimientos 23
4. Implementación 28
4.1. XenServer y OpenStack 28
4.1.1. DevStack 28
4.1.2. Mirantis OpenStack 33
4.1.3. Resultados de OpenStack en XenServer 43
4.2. CentOS y OpenStack 44
4.2.1. Instalación de CentOS 7 y particionamiento del disco 44
4.2.2. Instalación de Packstack y despliegue de OpenStack 46
4.2.3. Configuración de la red externa y del servicio de imágenes (Glance) 52
4.3. OpenLDAP 55
4.3.1. TLS 65
4.4. Configuración de OpenStack para que use LDAP como sistema autentificador. 76
4.5. Laboratorio virtual 84
5. Ejemplo práctico 92
5.1. Kali Linux 92
5.2. Metasploitable 98
5.3. Conectividad 101
Conclusiones y líneas futuras 106
Referencias 108

Tabla de Ilustraciones

Figura 1 - Niveles de responsabilidad según el modelo de servicio 6
Figura 2 - Módulos principales de OpenStack 7
Figura 3 - Módulos de OpenStack 7
Figura 4 - Arquitectura lógica de OpenStack 8
Figura 5 - Pestaña de instancias en Horizon 10
Figura 6 - Mapa Lógico de OpenStack Kilo 11
Figura 7 - Aviso sobre cambios al sistema 12
Figura 8 - Servicios de Mirantis OpenStack 13
Figura 9 - Configuración de red recomendada para desplegar Mirantis OpenStack sobre XenServer ... 13
Figura 10 - Configuración de OpenStack en Xen 15
Figura 11 - Árbol LDAP y atributos 17
Figura 12 - Instalación de XenServer 28
Figura 13 - Habilitar thin provisioning y utilizar una configuración IP estática 28
Figura 14 - Redes creadas para OpenStack 30
Figura 15 - VM (PV) Ubuntu con DevStack 30
Figura 16 - Snapshots de DevStack 32
Figura 17 - Hipervisores que se utiliza con DevStack 32
Figura 18 - Creando VM en DevStack 32
Figura 19 - Redes en XenServer 33
Figura 20 - Interfaces de Red para las VM 34
Figura 21 - Configuración del usuario de Fuel 34
Figura 22 - Configuración de la interfaz de red externa 34
Figura 23 - Configuración de la interfaz de la red VLAN PXE 35
Figura 24 - Configuración de seguridad (SSH) 35
Figura 25 - Configuración de la red PXE 35
Figura 26 - Configuración del DNS y Hostname 35
Figura 27 - Configuración de la imagen Bootstrap 36
Figura 28 - Guardar y reiniciar 36
Figura 29 - Seleccionar el método por el que el SO se instalará en los nuevos nodos 36
Figura 30 - Plugin de Citrix XenCenter 37
Figura 31 - Internal Managment Network Tool 37
Figura 32 - Soporte de XenServer Plugin 37
Figura 33 - Login Mirantis OpenStack 38
Figura 34 - Entornos OpenStack 38
Figura 35 - Plugins Mirantis OpenStack 39
Figura 36 - Nodos descubiertos por Mirantis OpenStack 39
Figura 37 - Creación de un nuevo entorno 40
Figura 38 - Credenciales XenServer 40
Figura 39 - Configuración de la red externa 40
Figura 40 - Configuración de la red "Storage" 41
Figura 41 - Configuración de la red "Managment" 41
Figura 42 - Configuración de la red flotante 41
Figura 43 - Configuración de las interfaces de los nodos 42
Figura 44 - Comenzar despliegue 42
Figura 45 - Instalación y configuración de los nodos esclavos 42
Figura 46 - Configuración de red del host OpenStack 44
Figura 47 - Particionamiento del disco duro para OpenStack 45
Figura 48 - Red externa en OpenStack 54
Figura 49 - Imágenes en OpenStack 54
Figura 50 - Configuración de red del servidor LDAP 55
Figura 51 - Conexión con el servidor LDAP 59

Figura 52 - Árbol visto en JXplorer 59
Figura 53 - Grupos administrativos de OpenStack en LDAP 61
Figura 54 - Usuarios y grupos de OpenStack en LDAP 61
Figura 55 - Grupo del proyecto LAB1 62
Figura 56 - Usuario OpenStack en LDAP 62
Figura 57 - Edito básico de contraseñas en JXplorer 63
Figura 58 - Editor avanzado de contraseñas en JXplorer 63
Figura 59 - Herramientas para ficheros LDIF en JXplorer 63
Figura 60 - Captura de paquetes sin TLS 75
Figura 61 - Captura de paquetes con TLS 75
Figura 62 - Login de OpenStack con el multi-dominio activado 77
Figura 63 - Acceso al dominio LDAP de OpenStack con un usuario en LDAP 83
Figura 64 - Crear router entre la red externa y la red privada 84
Figura 65 - Creación de una red privada para el proyecto LAB1 85
Figura 66 - Agregar interfaz al router 85
Figura 67 - Configuración de la nueva interfaz 85
Figura 68 - Vista general del router R1 del proyecto LAB1 86
Figura 69 - Máquinas Virtuales del laboratorio virtual 86
Figura 70 - Grupo de seguridad default 87
Figura 71 - PING a la VM en el proyecto LAB1 87
Figura 72 - Pruebas de PING desde VM en el proyecto LAB1 87
Figura 73 - SSH a la VM en el proyecto LAB1 88
Figura 74 - Credenciales en formato "RC File" 88
Figura 75 - Lanzamiento del laboratorio virtual desde la red externa (Windows con WSL) 90
Figura 76 - Error al importar la imagen OVA 92
Figura 77 - Nueva Imagen y nuevo sabor para Kali Linux 93
Figura 78 - Creación de volumen para Kali Linux 93
Figura 79 - El volumen se ejecuta como instancia 93
Figura 80 - Fallo durante el boot de Kali Linux 94
Figura 81 - Asociación del volumen Kali Linux en OpenStack 94
Figura 82 - GRUB 2 de Kali Linux 94
Figura 83 - Se queda trabado intentado arrancar "User Manager for UID 131" 95
Figura 84 - Kali Linux del proyecto LAB1 en OpenStack 97
Figura 85 - Actualizar arranque (GRUB) de Kali Linux 97
Figura 86 - Subir Imagen a OpenStack 98
Figura 87 - Imágenes en OpenStack 98
Figura 88 - Se crea el volumen para Metasploitable 99
Figura 89 - Se crea una instancia del nuevo volumen 99
Figura 90 - Volumen de Metasploitable adjuntado en la ruta /dev/vda 99
Figura 91 - Fallo al arrancar Metasploitable y Shell de mantenimiento 100
Figura 92 - Metasploitable arrancado a través de la Shell de mantenimiento 100
Figura 93 - PING a Metasploitable desde Kali Linux 101
Figura 94 - PING a Kali Linux desde Metasploitable 101
Figura 95 - Configuración de las redes 101
Figura 96 - Ruta estática del proyecto LAB1 al proyecto LAB2 102
Figura 97 - Ruta estática del proyecto LAB2 al proyecto LAB1 102
Figura 98 - PING a Metasploitable desde Kali Linux 102
Figura 99 - PING a Kali Linux desde Metasploitable 102
Figura 100 – [Ejemplo práctico] Lanzamiento del laboratorio virtual (Windows con WSL) 105

Resumen

En este trabajo se hace uso de un servicio de directorio (LDAP) como servicio autentificador para un sistema basado en la nube (OpenStack). Es un trabajo centrado en el aspecto práctico y es totalmente reproducible.

A la hora de desplegar OpenStack nos encontramos con múltiples soluciones dependiendo del sistema anfitrión sobre el que se instala. Al ser un estudio eminentemente práctico, los recursos existentes en los laboratorios recomendaban el despliegue de OpenStack sobre el sistema operativo XenServer, el cual incluye su propio hipervisor. Sin embargo, tras múltiples intentos, se demostró que la integración de ambos sistemas, está lejos de ser perfecta. Los principales problemas se traducen en fallos en la estabilidad y el bloqueo de determinadas funcionalidades.

Como alternativa se ha procedido a hacer un despliegue, considerado por OpenStack como estándar, sobre sistemas operativos Linux. Partiendo de esta configuración típica, se procedió a introducir un servicio de directorio adaptado para su introducción dentro de entorno de OpenStack como sistema alternativo de autentificación de usuarios, ya que provee funcionalidades muy interesantes que no aparecen en OpenStack por defecto, como es la introducción de una capa adicional de seguridad implementada con TLS y su aislado mediante capas adicionales de cortafuegos adaptados.

El sistema de autentificación global así implementado se comporta como un sistema multi-dominio, en el que existe el dominio por defecto con los servicios de OpenStack y un segundo dominio LDAP en donde se ubican los usuarios. La consecuencia directa es la posibilidad de establecer diferentes niveles de acceso, ya no solo por tipos de usuarios, sino también por tipos de servicios solicitados.

Todo el sistema ha sido implementado y probado mediante el desarrollo de casos de uso. En concreto, se han diseñado un par de laboratorios virtuales mediante los cuales se demuestra como la autentificación en capas independientes posibilita la ejecución automática de laboratorios virtuales adaptados directamente por parte de los usuarios. Los ejemplos planteados permiten arrancar máquinas especializadas sobre diferentes redes virtuales, permitiendo simular un entorno completo de red sobre el cual probar, por ejemplo, conceptos básicos relacionados con la ciberseguridad.

Abstract

In this project a directory service (LDAP) is used as a mean of authentication for a system based on the cloud (OpenStack). This work is focused on the practical aspect and is fully reproducible.

At the time of deploying OpenStack we find multiple solutions depending of the host system on which is installed. In this study, being eminently practical, the existing resources in labs recommend the deployment of OpenStack on the XenServer OS, which includes its very own hypervisor. However, after many tries, it was proven that the integration of both systems, it is far from perfect. The main problems are translated on stability issues and the blocking of certain functionalities.

As an alternative a deployment has been made, considered by OpenStack as standard, on the Linux OS´s. Starting from this typical configuration, we proceeded to introduce a directory service adapted for its introduction in the OpenStack environment as an alternative system of user authentication, since it provides very interesting functionalities that do not appear in OpenStack by default, such as the introduction of an additional layer of security implemented with TLS and its isolation through additional layers of adapted firewalls.

The global authentication system thus implemented behaves as a multi-domain system, in which there is the default domain with the OpenStack services and a second LDAP domain where the users are located. The direct consequence is the possibility of establishing different levels of access, not only by types of users, but also by types of services requested.
The whole system has been implemented and tested through use case development. In particular, a pair of virtual laboratories have been designed by which it is demonstrated how the authentication in independent layers allows the automatic execution of virtual laboratories adapted directly by the users. The examples presented allow to start specialized machines on different virtual networks, allowing to simulate a complete network environment on which to test, for example, basic concepts related to cybersecurity.

1. Introducción y objetivos

Como punto de partida este apartado realiza una breve introducción al problema abordado, definiendo cuáles son los objetivos marcados, así como la distribución que seguirá este documento.

1.1. Introducción

En la actualidad los sistemas basados en la nube están ganando popularidad. Cada vez más sistemas hacen uso de esta tecnología y se aprovechan de la flexibilidad que ofrecen los servicios basados en ella. Desde el punto de vista del uso educativo del cloud, los sistemas basados en Cloud son especialmente adecuados para su integración dentro planes formativos muy diversos y está actualmente ampliamente extendido. Al estar siendo implantado en todos los niveles educativos, es la base de lo que se conoce hoy en día como Educación 2.0.

En algunos casos la utilización de IaaS en el mundo empresarial se plantea como una paulatina eliminación de los servidores físicos propios y su sustitución por servidores virtuales ubicados en Centros de procesamiento de datos (CPD) remotos. Esta solución redunda de forma inmediata en importantes ahorros de costes, pero no se puede plantear para determinados servicios ya que se asumirían importantes riesgos al no controlar directamente sus equipos y se adquiría una gran dependencia de un proveedor.

1.2. Motivación y objetivos

El objetivo básico de este trabajo es desplegar un servicio de computación en la nube gracias a OpenStack, que es un supervisor que cuenta con una amplia variedad de herramientas para administrar plataformas de computación en la nube ya sean públicas o privadas.

Inicialmente se intentaba utilizar OpenStack como Supervisor y XenServer como Hypervisor tipo 1 o “bare-metal”. El XenServer se encargaría de la virtualización de los recursos hardware mientras que OpenStack se encargaría de la administración de las máquinas virtuales, usuarios, proyectos y redes virtuales.

El objetivo principal es conseguir poder utilizar un servidor LDAP como sistema autentificador, y darles acceso a usuarios a ciertos proyectos dentro de OpenStack.

Otro objetivo era averiguar la manera de lanzar un laboratorio virtual en OpenStack desde un script sin la necesidad de ir al servidor y arrancar las VM desde el CLI (Command Line Interface) o desde la interfaz web que ofrece OpenStack, para simplificar el proceso de lanzar máquinas.

Un objetivo secundario era segurizar el proceso de autenticación entre OpenStack y el servidor LDAP, haciendo uso del protocolo TLS. Esto es importante porque de otra forma el usuario y contraseña viajan en claro, lo que supone un gran problema de seguridad.

1.3. Organización del documento

A continuación de este primer capítulo de introducción, el capítulo 2 explica los conceptos teóricos de las tecnologías que se van a utilizar en el trabajo. Es recomendable conocer el funcionamiento y funcionalidad de dichas tecnologías para entender qué papel juegan en el proyecto. Las tecnologías que se van a explicar que son los “servicios de computación en la nube”, que es “OpenStack” y distintas formas de desplegarlo, el Hypervisor “XenServer” que se utiliza en una de las formas de desplegar OpenStack, que es y para que se utiliza “LDAP”, y una tecnología para la seguridad en las comunicaciones “TLS”

En el capítulo 3 se van a tratar los aspectos prácticos concretos en este proyecto. De acuerdo con los objetivos del trabajo, lo que se busca es utilizar LDAP como sistema autenticador para OpenStack. Por lo que primero que se va a tratar en este capítulo es identificar lo que se va a necesitar a lo largo del trabajo. Como este trabajo es principalmente práctico, se ha descrito un ejemplo práctico reproducible y también se han descrito los requerimientos hardware y software que se han utilizado.

El capítulo 4 es que más se extiende. En este capítulo se describe la implantación que se ha utilizado en el trabajo, incluyendo todos los pasos necesarios para su replicación y explicación de los mismos. Primero se introducen las diferentes formas de desplegar OpenStack, después como instalar y configurar un servidor LDAP adecuado para la autenticación de OpenStack y sus opciones de seguridad. Una vez se disponen de las dos máquinas se procede a la configuración de OpenStack para que haga uso del servidor LDAP como sistema autenticador. Y por último se despliega un ejemplo de Laboratorio Virtual sencillo, en el que se pueden controlar las máquinas virtuales fácilmente desde un script, en este caso para iniciar las máquinas.

En el capítulo 5 se trata un ejemplo práctico más complejo en el que se crean dos máquinas virtuales (Kali Linux y Metasploitable) que se encuentran en redes distintas y hay que configurar las rutas para que se puedan comunicar y así se prepara el escenario para una práctica de seguridad en las redes.

2. Conceptos teóricos

En este capítulo se van a exponer los principios básicos de varias de las tecnologías utilizadas en este trabajo.

2.1. Servicios de computación en la nube

Los servicios de computación en la nube ofrecen recursos virtuales a los clientes. Estos recursos al ser virtuales no están ligados a una máquina física directamente, lo que hace que, si una aplicación o servicio momentáneamente requiere más o menos recursos, las capacidades contratadas pueden ser ajustadas mucho más rápidamente que lo que supondría su contrapartida hardware.

Algunas de las empresas más grandes que ofrecen servicios en la nube son Azure (Microsoft), Google App Engine, Amazon Web Service, Go-Grid, OpSource o Rackspace.

Los servicios de computación en la nube se pueden agrupar de varias formas, basándose en dónde se encuentra la nube o en el servicio que la nube ofrece. [1]

Dependiendo de su localización, según los modelos de despliegue:

- **Pública**: Se refiere a la infraestructura que está abierta al público en general. Su proveedor es una organización que vende sus servicios a terceros.

- **Privada**: Se refiere a la infraestructura que sólo sirve a una única organización. No necesariamente será dicha organización quien la gestione ni tampoco tendrá por qué estar en sus propias instalaciones.

    - **Nube comunitaria**: Se refiere a la infraestructura compartida por un conjunto limitado de organizaciones

- **Híbrida**: Parte de las funcionalidades se ofrecen de manera privada (o en comunidad) y parte en un Cloud público. Ej.: Almacenamiento privado, computación y aplicaciones en público.

También se pueden agrupar según el grado de control sobre los recursos contratados, o modelos de servicio:

- **Software as a Service** (SaaS)
    - El proveedor ofrece aplicaciones web.
    - El consumidor sólo necesita un navegador web.
- **Platform as a Service** (PaaS)
    - El proveedor ofrece un entorno de ejecución de aplicaciones.
    - El consumidor ejecuta sus propias aplicaciones, que ofrece a terceros, en el entorno que ofrece el proveedor.
- **Infrastructure as a Service** (IaaS)
    - El proveedor ofrece sus servidores.
    - El consumidor crea máquinas virtuales (VM) sobre las que tiene privilegios de administrador y sobre ellas crea todo el entorno que necesite.

Si bien el modelo de Cloud define diferentes niveles de despliegue y responsabilidad, como se muestra en la Figura 1, su utilidad va a depender casi exclusivamente de las pretensiones de los proveedores de cada modelo y de las necesidades específicas de los clientes para los que se diseña. Así. el modelo SaaS es especialmente útil para el diseño de aplicaciones, servicios y funcionalidades que. en el caso de la educación, permitiría desplegar contenidos y cursos de forma rápida y flexible. Por su parte, el interés sobre los clouds PaaS proviene fundamentalmente de los estudios relacionados con el desarrollo de software, por lo que son especialmente interesantes para la implantación de cursos especializados, interactivos y a medida. Por último, el modelo IaaS, es especialmente interesante en el caso de estudios técnicos, permitiendo desarrollar laboratorios virtuales, diseñados a medida en función de las necesidades de cada alumno, no solo en conocimientos, sino en las infraestructuras informáticas requeridas para llevar a cabo su aprendizaje.

Este trabajo se centra en el modelo de servicio de infraestructura como servicio (IaaS). Para ello se va diseñar una plataforma cloud basada en el sistema OpenStack, sobre un único host como prueba de concepto.

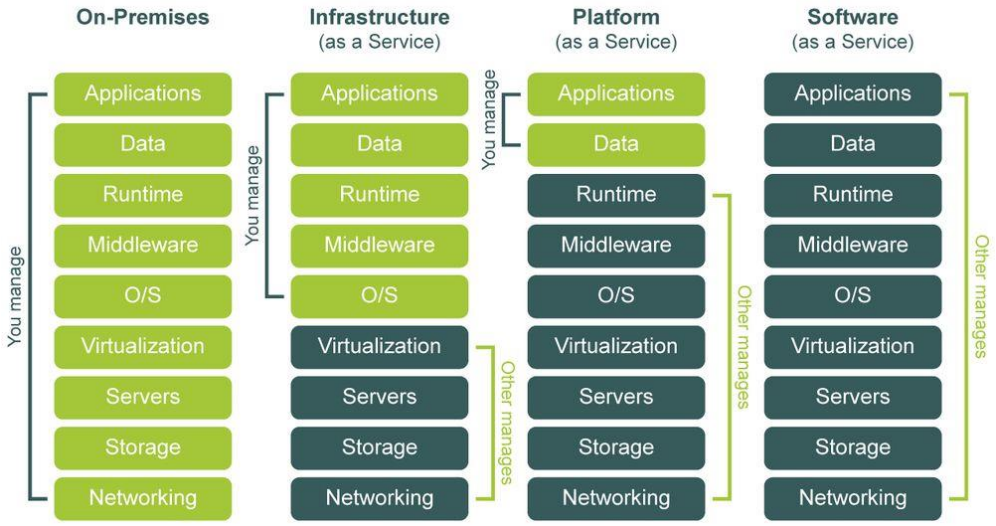


Figura 1 - Niveles de responsabilidad según el modelo de servicio

2.2. OpenStack

OpenStack es una colección de tecnologías Open Source que proporcionan un software para el despliegue escalable de un cloud computing. OpenStack proporciona Infraestructura como Servicio o IaaS (Infrastructure as a Service) y es un proyecto que se inició en el año 2010 por la empresa *Rackspace Cloud* y por la agencia espacial norteamericana, NASA. Actualmente más de 150 empresas se han unido al proyecto, entre las que se encuentran empresas tan importantes como AMD, Intel, Canonical, SUSE Linux, Red Hat, IBM, Dell, HP, Cisco, etc. OpenStack es software libre bajo los términos de la licencia Apache.

En la Figura 2 se pueden ver los módulos principales en los que está basado OpenStack. Esos módulos son **Compute**, que administra recursos como CPU y RAM de las VM, **Networking**, que administra, las redes virtuales que se pueden definir en OpenStack además de la conectividad con las redes físicas y **Storage** que es el almacenamiento en la nube.

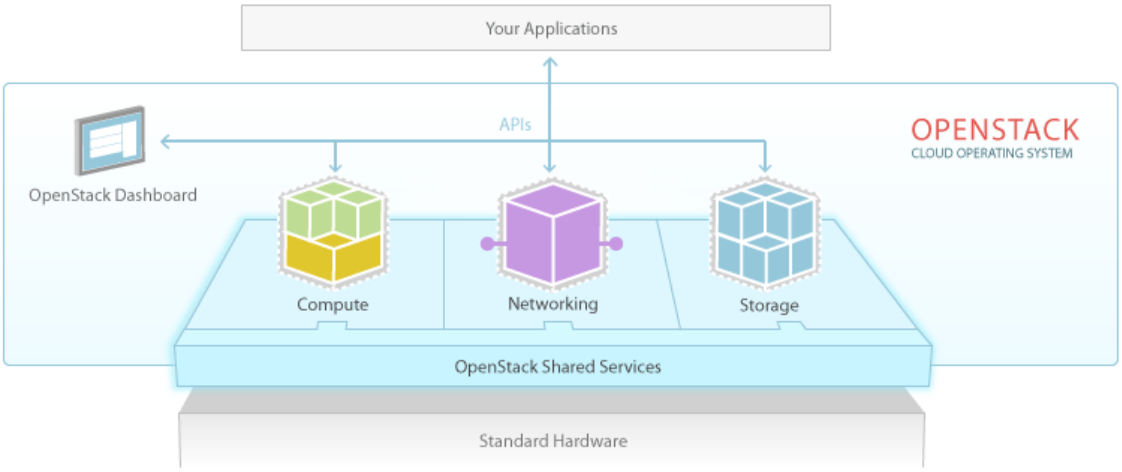


Figura 2 - Módulos principales de OpenStack

Desde el punto de vista de software, la Figura 3muestra cómo OpenStack es una colección de proyectos de software libre mantenidos por la comunidad que incluyen varios componentes, de los cuales los más importantes y utilizados son:

- **Nova**: OpenStack Compute.
- **Glance**: OpenStack Image Service.
- **Swift**: OpenStack Object Storage
- **Cinder**: OpenStack Block Storage.
- **Neutron**: OpenStack Networking Service
- **Heat**: Orchestration Service
- **Ceilometer**: Monitor Service
- **Keystone**: Identity Service
- **Horizon**: Dashboard

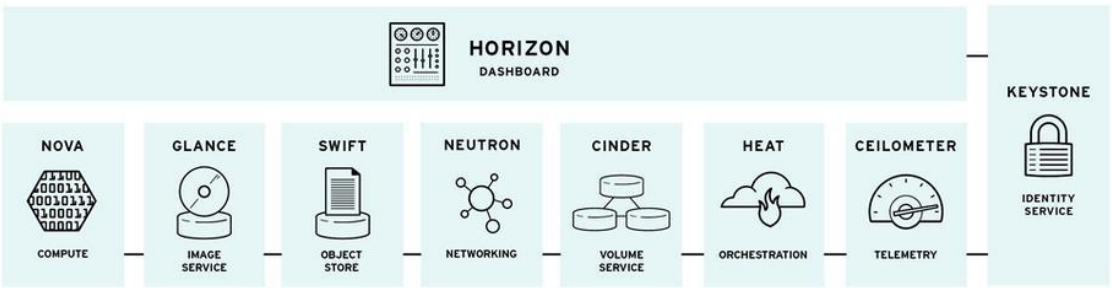


Figura 3 - Módulos de OpenStack

La arquitectura real de una nube es mucho más compleja que unos pocos servicios, pero estos son los principales módulos y algunos agrupan múltiples servicios. En la Figura 4, se representa el funcionamiento OpenStack a groso modo.

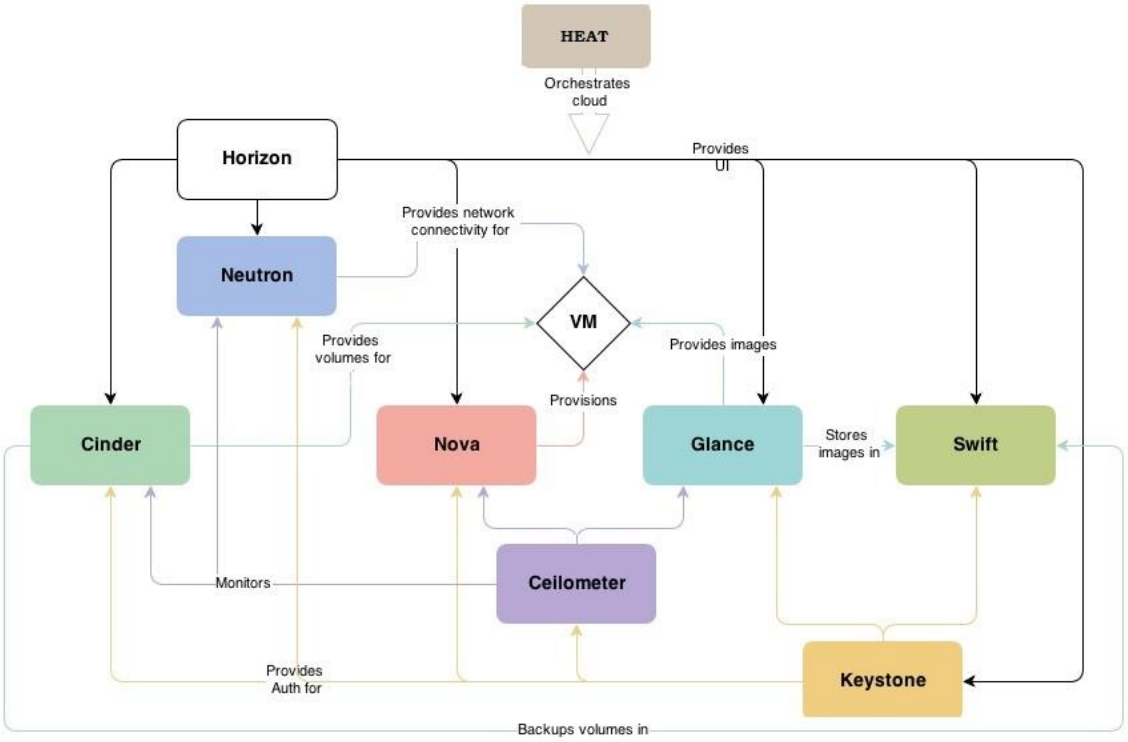


Figura 4 - Arquitectura lógica de OpenStack

A continuación se hace una breve descripción de cada elemento:

Nova

Es el módulo que permite implementar los servicios y las bibliotecas asociadas de forma que su provisión sea escalable y bajo demanda, y permita el acceso a los recursos en modo autoservicio.

Está preparado para ser desplegado tanto en grandes como en pequeños grupos de hosts, pudiendo escalarse los nodos de manera relativamente sencilla. Este módulo es el que otorga a los usuarios las cantidades de los recursos que piden, dentro de unos límites. Pudiendo ajustarse a las necesidades de los clientes de forma dinámica.

Glance

Implementa los denominados servicios de imagen de máquinas virtuales. Entre sus funciones están el descubrimiento, el registro y la recuperación de dichas imágenes, las cuales no dejan de ser la realidad física de los ordenadores virtuales que son utilizados por las aplicaciones, los servicios y los clientes. Glance permite la consulta de metadatos de una imagen VM (Virtual Machine), así como la recuperación de la imagen real, así como el resto de operaciones de gestión gracias al uso de una API RESTful, es decir, un mecanismo de interacción con el sistema de gestión a base de comandos HTTP que pueden ser ejecutados directamente desde el lado del cliente, del servicio o de las aplicaciones.

Las imágenes de VM disponibles a través de Glance se pueden almacenar en una variedad de ubicaciones, desde un simple sistema de archivos, hasta sistemas de almacenamiento remotos y/o especializados, como el propuesto por el proyecto OpenStack Swift. Es en estos almacenes donde se suben las imágenes, normalmente archivos con extensión ISO, VMDK o QCOW2, entre otras, que luego se van a utilizar para crear volúmenes o instancias en Nova, es decir, los ordenadores virtuales.

Swift

Como se ha indicado anteriormente, Swift se comporta como un servicio de almacenamiento de objetos, altamente disponible y distribuido. El proveedor puede utilizar Swift para almacenar gran cantidad de datos de manera eficiente, segura y barata, ya que está diseñado para ser escalado, optimizado, y ofrecer durabilidad, disponibilidad y concurrencia en todo el conjunto de datos. Swift es especialmente ideal para el almacenamiento de datos no estructurados, los cuales presentan el inconveniente de que normalmente no presentan ningún límite a la hora de crecer.

Cinder

Es un servicio de almacenamiento de bloques para OpenStack. A la contra de Swift, Cinder virtualiza la administración de dispositivos de almacenamiento en bloque y proporciona a los usuarios finales una API de autoservicio para solicitar y consumir esos recursos sin requerir ningún conocimiento de dónde se despliega realmente su almacenamiento ni en qué tipo de dispositivo. Este nivel de abstracción se consigue mediante el uso de una implementación de referencia, denominada LVM o *plugin drivers* en función del tipo de almacenamiento virtualizado.

Neutron

OpenStack Neutron es un proyecto de redes SDN (Software-Defined Networking), enfocado en la entrega de redes como servicio (NaaS) en entornos virtuales de computación. Funcionalmente, este módulo se encarga de proveer conectividad entre las redes e interfaces virtuales (vNIC), y las interfaces reales de comunicación, y es a su vez controlado por Nova. Al igual que el resto Implementa una API de interacción

Heat

Este elemento permite realizar la orquestación de los recursos de infraestructura para una determinada aplicación en la nube. Para ello se definen una serie de archivos de texto que son tratados como plantillas y procesados como si fuese un script (secuencia de comandos). Heat proporciona una API REST nativa de OpenStack y una API de consulta compatible con AWS CloudFormation, que es servicio dedicado a la estandarización, a traves de plantillas, y replicacion delas arquitecturas para el abastecimiento de recursos.

Ceilometer

El objetivo de Ceilometer es recopilar, normalizar y transformar, de forma eficiente, los datos producidos por los servicios de OpenStack. Los datos que recolecta están

destinados a ser utilizados para crear diferentes vistas y ayudar a resolver varios casos de uso de telemetría.

Aodh y Gnocchi son dos ejemplos de servicios que amplían los datos de Ceilometer. Aodh habilita la capacidad de activar acciones basadas en reglas definidas contra datos de muestra o eventos recopilados por Ceilometer. Por su parte, Gnocchi es el proyecto de un TDBaaS (Time Series Database as a Service), cuyo objetivo era almacenar datos de series de tiempo que fuesen recopilados.

Horizon

Es la implementación canónica del panel de control para OpenStack. Es extensible y proporciona una interfaz de usuario basada en web para los servicios de OpenStack.

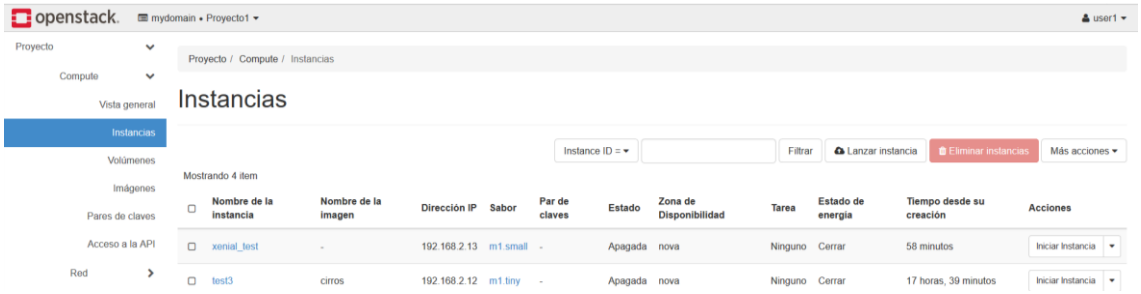


Figura 5 - Pestaña de instancias en Horizon

Keystone

Keystone es un servicio de OpenStack que proporciona la denominada Identity API. En este caso, los comandos permiten acceder a un cliente de autenticación, descubrimiento de servicio y autorización para servicios en nube. La identificación puede realizarse de forma federada, es decir, una misma identidad permitirá acceder a múltiples redes, servicios o aplicaciones, pertenecientes al mismo o diferentes proveedores.

Especialmente interesante para este trabajo, Keystone es la entidad que autentifica la identidad de cualquier usuario de OpenStack, o cualquiera de los módulos que lo compongan. Esto afecta a administradores, proveedores, clientes y usuarios, y cobra especial sentido si se tiene en cuenta que no todos los módulos están en el mismo host y que se encuentran distribuidos entre múltiples máquinas.

La autentificación implementada por Keystone está basada en el intercambio de tickets personalizados para permitir el acceso (denominados tokens) Así, un determinado usuario que desee acceder a Openstack o cualquiera de sus servicios, recibirá un token que podrá usar duante un periodo de tiempo limitado. Pasado ese tiempo el usuario tiene que pedir un nuevo token.

El proceso de autentificación es compatible con LDAP, OAuth, OpenID Connect, SAML y SQL. Keystone, el cual se implementa por defecto. Esto significa que tras la instalación de Openstack, todo el sistema de gestión de identidades se encuentra físicamente instalado en alguna de las máquinas que componen la infraestructura física de OpenStack. En el caso de que los clientes dispusieran de su propio mecanismo de gestión de identidades, como es el caso de la mayoría de las empresas que cuentan son servicios en red, sería necesario seleccionar e implementar el

correspondiente mecanismo de federación, es decir, aceptar que la autentificación pueda ser realizada por un servidor diferente a Keystone, y una vez certificada su identidad, se garantice el acceso a los recursos de Openstack que hayan sido pactados. En el caso por ejemplo de las universidades, es bastante frecuente el que los correspondientes Campus Virtuales ofrecidos a alumnos y trabajadores presenten sistemas de autentificación propios, basados en la mayoría de los casos en sistemas de directorio LDAP. En este trabajo se lleva a cabo precisamente una prueba de concepto que permite conocer cuáles son los pasos requeridos para la integración de un sistema de autentificación propietario basado en LDAP como sistema federado para el acceso a un despliegue basado en OpenStack.

Aunque el mapa lógico de OpenStack es en realidad mucho más complejo de lo que se ha tratado aquí, las anteriores definiciones sirven para hacerse una idea de cómo funciona internamente y qué función tiene cada módulo. En la Figura 6, podemos ver el Mapa lógico para OpenStack, en su versión Kilo (Abril 2015).

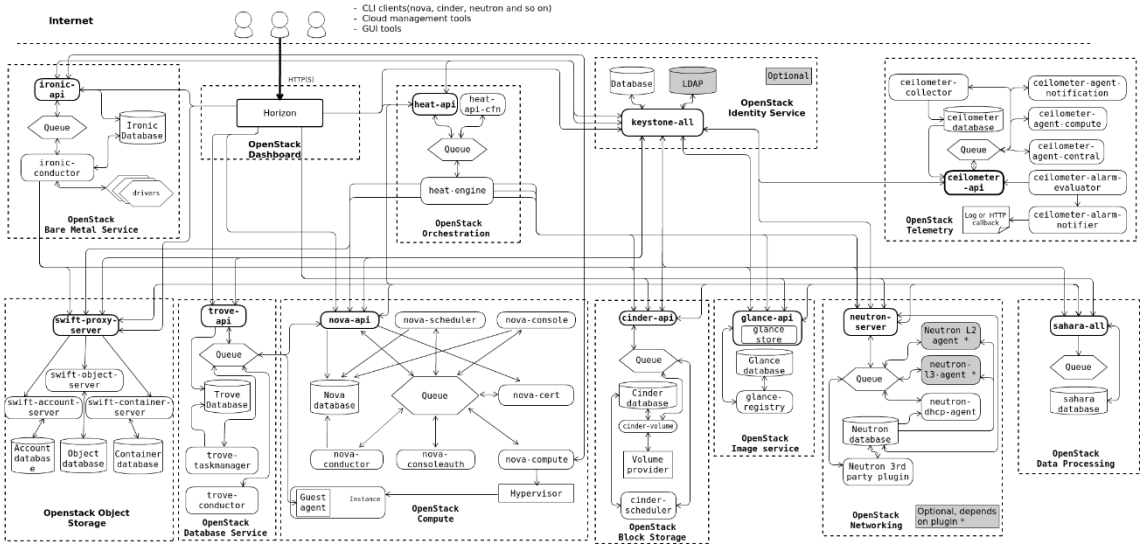


Figura 6 - Mapa Lógico de OpenStack Kilo

Toda la complejidad del esquema anterior queda patente a la hora de proceder a la instalación de Openstack desde cero, de varias horas en las configuraciones más simples a varias jornadas en el peor de los casos. No es de extrañar, que desde su aparición, no hayan dejado de aparecer alternativas que faciliten la implementación de sistemas basados en OpenStack de la forma más simple posible, aunque normalmente encaminadas a facilitar un rápido despliegue de configuraciones de prueba, que permitan ver "rápidamente" (en unas pocas horas) las bondades de OpenStack.

2.2.1. DevStack

Esta fue la primera de las opciones que apareció. Su origen está en facilitar una configuración básica funcional de OpenStack, a modo de prueba. Se basa en una serie de scripts extensibles utilizados para levantar rápidamente un entorno completo de OpenStack basado en las últimas versiones de todo, ya que se descarga desde GitHub en la rama master.

Se utiliza interactivamente como un entorno de desarrollo y como base para gran parte de las pruebas funcionales del proyecto OpenStack. Y el código está disponible en el GitHub:

- `https://github.com/openstack-dev/devstack`

En la documentación sobre DevStack, en la página web de OpenStack se advierte sobre los cambios a bajo nivel que realiza el script. Estos cambios pueden afectar a la estabilidad del sistema por lo que se recomienda que se utilice solo sobre máquinas que tengan ese propósito, ya sean servidores dedicados o máquinas virtuales.

> **Warning:** DevStack will make substantial changes to your system during installation. Only run DevStack on servers or virtual machines that are dedicated to this purpose.

Figura 7 - Aviso sobre cambios al sistema

DevStack configura rápidamente una versión de OpenStack, completamente actualizada, pero no está pensado para despliegues reales, ya que solo es una suite para pruebas en la que los desarrolladores pueden probar la instalación antes de llevarla a cabo en un servidor real. Según el proyecto original, DevStack está pensado para desplegar rápidamente un OpenStack en un entorno Ubuntu o Fedora, y así ayudar a desarrolladores a contribuir productivamente al proyecto OpenStack sin necesidad de entender todas las partes del sistema al mismo tiempo. Es por eso que para evitar que se utilice DevStack como atajo a la hora de hacer un despliegue real, cuando se apaga el host, se pierde toda la configuración.

En el pasado había un script que volvía a configurar OpenStack después de reiniciar el sistema *rejoin-stack.sh* pero en la actualidad ya no está. Para volver a configurar DevStack se debe ejecutar un script *unstack.sh* de desinstalación, antes de lanzar otra vez *stack-sh*.

El apagado de la máquina no solo mata todos los servicios OpenStack, sino que al reiniciar y reinstalar también se reinician y reconfiguran todas las bases de datos creadas en ejecuciones previas, con lo que vuelve al mismo estado de la primera instalación (en las versiones anteriores, el script *rejoin-stack.sh* permitía reanudar la sesión anterior y no borraba las bases de datos).

Para instalar OpenStack a través de DevStack se puede configurar la instalación haciendo uso de un fichero de configuración *local.conf*. La información en la sección *[localrc]* permite especificar las variables de entorno, como contraseñas u opciones de red entre otras cosas. Dependiendo del tipo de despliegue que se desee probar puede hacerse uso de hipervisores como XenServer [2], ESXi o Hyper-V. También existe la posibilidad de instalar DevStack sobre sistemas virtualizados mediante VirtualBox o VMWare, lo que permite probar el sistema sobre equipos personales, con recursos más discretos.

En cualquier caso, el comportamiento de los scripts y la limitación a la hora de reiniciar el sistema desaconseja su uso como opción a largo plazo, aunque existe la posibilidad de evitar en cierta manera dicha limitación mediante la toma de instantáneas (Snapshots) justo antes de apagar la máquina, y restaurar siempre la última instantánea en el reinicio de la misma. Estos *snapshots* suelen ser soportados por las funcionalidades incluidas en los hipervisores mencionados anteriormente.

- `http://docs.openstack.org/developer/devstack`

2.2.2.Mirantis OpenStack

Este software constituye una distribución de OpenStack que viene completamente empaquetada y preparada para facilitar la instalación, e integración con el Hypervisor XenServer. Fuel tiene los servicios principales de OpenStack y algunos de los extras más utilizados, como se puede ver en la Figura 8

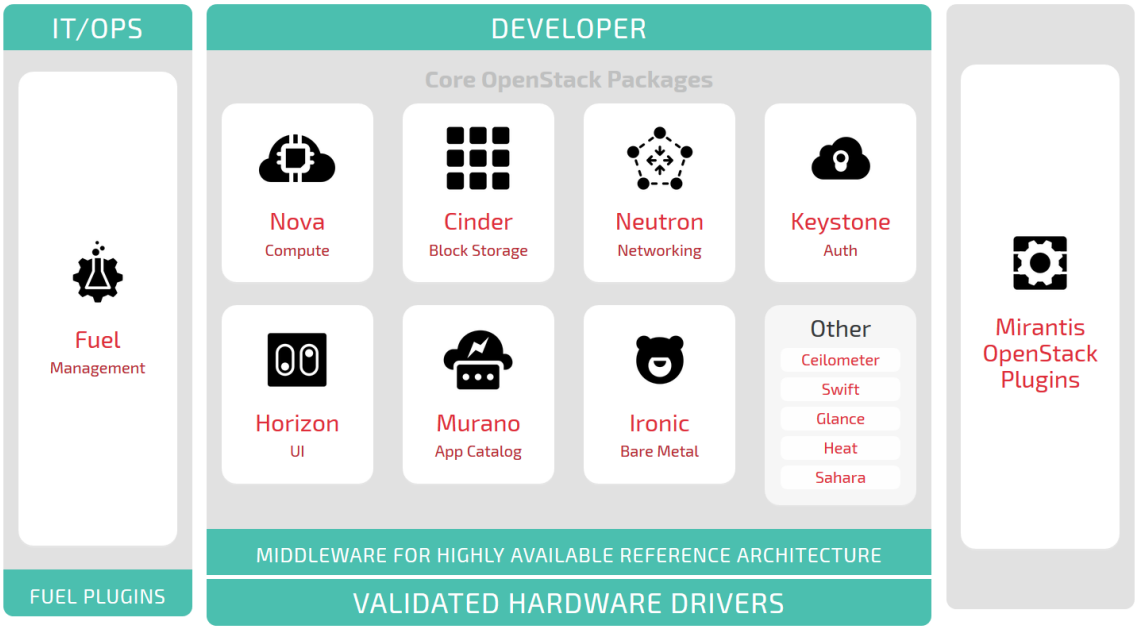


Figura 8 - Servicios de Mirantis OpenStack

En el caso de utilizar Mirantis OpenStack en conjunción con XenServer, hay que preparar la red para que el instalador localice y controle las máquinas virtuales en el host. Mirantis OpenStack hace uso de una red PXE[1] para cargar sistemas operativos y configurar los nodos que formarán parte de OpenStack.

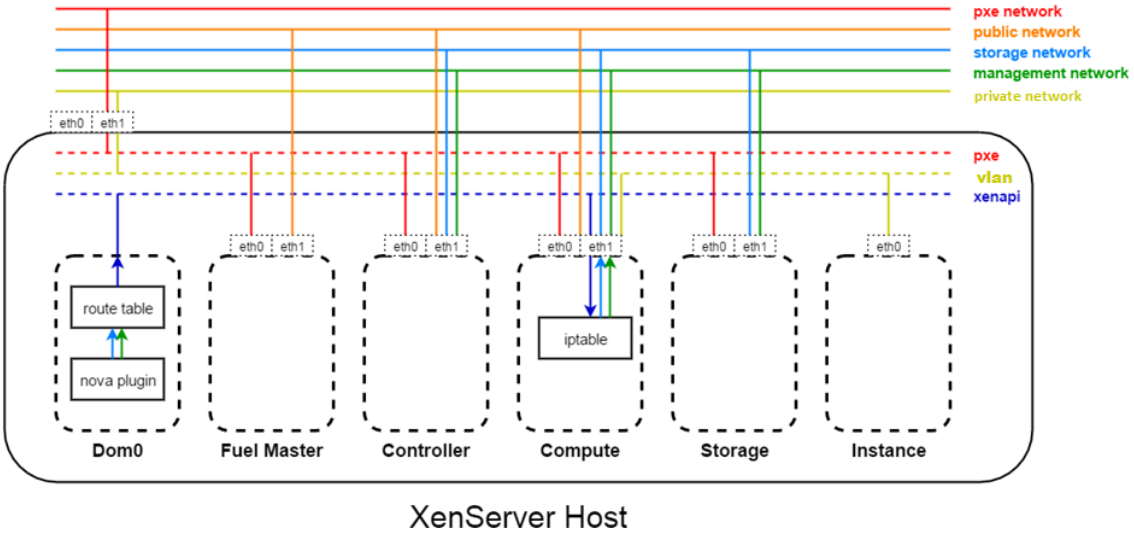


Figura 9 - Configuración de red recomendada para desplegar Mirantis OpenStack sobre XenServer

Una vez preparada la red, se debe instalar un plugin que permite hacer uso del Hypervisor. Actualmente la versión de OpenStack que se incluye en el paquete resulta algo antigua, a lo que hay que añadir que tampoco se encuentran fácilmente nuevas versiones del plugin.

[1] Una red PXE es simplemente un entorno de red en el que se ejecutan instrucciones antes del arranque de una máquina. Suele ser utilizado para arrancar e instalar sistemas operativos en diversos clientes a través de una red.

2.2.3.Packstack

En esencia se parece a DevStack, ya que sirve para hacer un despliegue rápido de OpenStack. A diferencia de DevStack no obtiene el código de GitHub donde está el master, la versión más actualizada de OpenStack, sino que ofrece un paquete con el código de cuando se liberó la versión. Otra diferencia con DevStack es que OpenStack mantiene las configuraciones entre reinicios del host. Esto es porque los procesos de OpenStack se ejecutan como servicios en Packstack y pueden ser reiniciados con el sistema.

Afortunadamente las versiones que tiene Packstack están actualizadas, no como Mirantis OpenStack, y a fecha de este trabajo está actualizado con la versión más reciente Ocata. Actualmente está disponible para Red Hat y para CentOS (Una distribución basada en Red Hat).

Packstack usa Puppet para desplegar los diversos módulos que componen OpenStack. Se puede hacer uso de un fichero de configuración al igual que en DevStack para personalizar la instalación de OpenStack. En este fichero se puede modificar casi todas las propiedades que puede tener OpenStack, desde contraseñas, red, Hipervisores, tamaños de disco o incluso si instalar más o menos módulos del proyecto OpenStack.

Todas estas configuraciones son fáciles de entender y están acompañadas de una extensa documentación en la web. Como Packstack es ampliamente usado tanto por empresas como por particulares hay mucha gente detrás de estos paquetes, lo que es beneficioso a la hora de buscar errores y configuraciones alternativas. Además RDO, una comunidad online dedicada al mantenimiento de paquetes OpenStack para Red Hat Enterprise Linux (RHEL) y los sistemas derivados como CentOS, que tiene documentación muy detallada de algunas aplicaciones en relación con Packstack y OpenStack.

2.3. XenServer

XenServer está basado en el Xen ProjectTM Hypervisor. El Hypervisor del proyecto Xen es un virtualizador "bare-metal" de código abierto que es una de las plataformas de virtualización más utilizadas en el mundo, además de estar integrado en múltiples proyectos de orquestación en la nube como OpenStack.

La diferencia entre Xen Project y XenServer, es que el primero es un proyecto el cual no viene con un sistema operativo (dom0) lo cual no lo convierte en un Hypervisor tipo 1, y que XenServer es un producto listo para instalar, está preparado para ser instalado, con un kernel Linux y un sistema operativo mínimo, en el dom0.

Dom0 sería el espacio reservado para el propio sistema virtualizador, XenServer. Aquí se encuentran las aplicaciones y archivos que requiere el sistema operativo y desde donde se puede controlar el hardware directamente.

El resto de máquinas virtuales que se ejecutarían sobre el Hypervisor se les conoce como domU y son en estas máquinas donde se instalaría OpenStack.

Para instalar OpenStack es necesario contar con una API para la comunicación entre los domU y el dom0, porque para poder crear y administrar las maquinas desde OpenStack (domU) hay que poder acceder a (dom0) quien controla el hardware. [3]

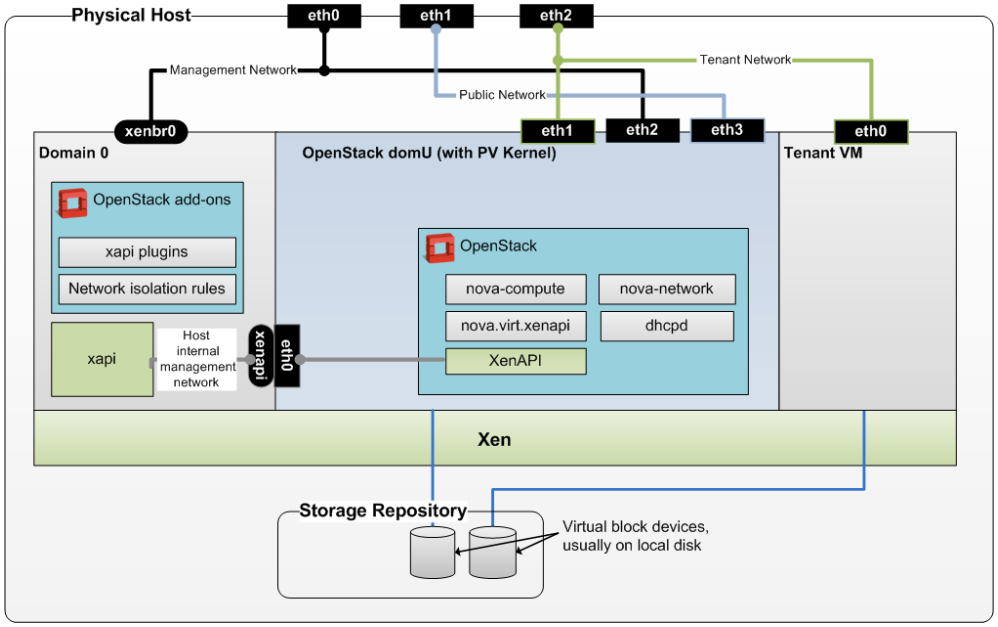


Figura 10 - Configuración de OpenStack en Xen

Una de las ventajas de XenServer, es que se puede obtener de forma gratuita y cuenta con un software para la administración de las maquinas creadas en el host desde Windows, XenCenter. No hay que confundir esta funcionalidad con computación en la nube, ya que es más limitada y no ofrece la misma flexibilidad y adaptabilidad de OpenStack.

Inicialmente fue una de las opciones barajadas para utilizar en la prueba de concepto, ya que se dispone de servidores de virtualización cuyo sistema operativo es XenServer, por lo que eran perfectos candidatos. Sin embargo, durante la realización de este trabajo se pudo comprobar en la práctica cómo OpenStack y XenServer presentan bastantes problemas de compatibilidad, especialmente en las versiones más actuales, presentando un comportamiento errático y lento.

Es por eso, que finalmente se decidiera probar una implementación basada en CentOS + Packstack, ya que además de funcionar mejor, permitía su instalación en equipos personales, es decir, requería de menos recursos de computación.

2.4. LDAP

LDAP corresponde con las siglas en inglés “Lightweight Directory Access Protocol” que hacen referencia a un protocolo que permite el acceso a través de la red a un servicio de directorio. Si bien los servicios de directorio pueden ser considerados como meras bases de datos, su estructura y manejo resulta bastante diferente. De hecho, los servicios de directorio están basados en el estándar X.500, del cual LDAP es su evolución más simple y con soporte para TCP/IP. Así, los puertos por defecto en los que se presta el servicio son en el 389, en comunicaciones sin encriptación y en el 636 en el caso de utilizar SSL.

Se trata de un protocolo que es usado para comunicarse con un directorio, al que se refiere como “directorio LDAP”, el cual puede ser soportado por casi cualquier tipo directorio.

Como se ha dicho anteriormente, un directorio es una base de datos especializada en almacenar información referente a usuarios, como el id de usuario, contraseña, nombre, email, dirección, localización, etc. Sus aplicaciones no solo se centran en las personas, sino también en las cosas, y más concretamente en los equipos, aplicaciones y servicios.

El sistema de directorio más conocido recibe el nombre de Active Directory (AD), de Windows. Esta es la implementación que ha hecho Microsoft para crear un directorio de usuarios, equipos o grupos, con el objetivo de administrar los inicios de sesión en los equipos conectados a la red, así como también la administración de políticas en toda la red. Este sistema, dadas sus aplicaciones, tiene una estructura compleja que se basa en múltiples protocolos, entre los que se incluye LDAP. Sin embargo, en el mundo del código abierto, LDAP da paso a implementaciones alternativas, como es el caso de OpenLDAP.

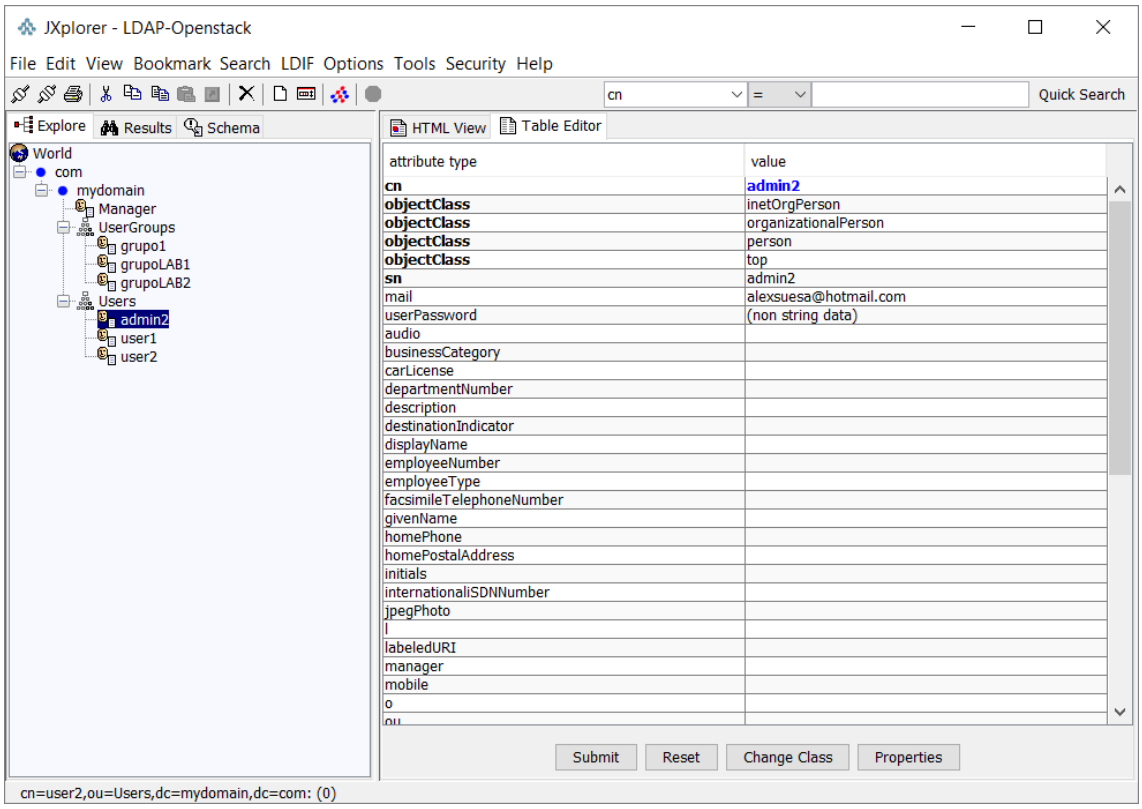


Figura 11 - Árbol LDAP y atributos

En el caso de OpenStack, la estructura de directorio utilizada, aunque compleja, solo requiere de unos pocos atributos para poder implementar servicios complejos, como por ejemplo el de autentificación ya que, en muchos casos, basta con que coincida el tipo de atributo o el valor.

En cualquier caso, los despliegues actuales de LDAP tienden a usar nombres siguiendo la estructura del Sistema de Nombres de Dominio, o DNS por sus siglas en inglés "Domain Name System", para estructurar los niveles más altos de la jerarquía. Conforme se desciende en el directorio pueden aparecer entradas que representan personas, unidades organizacionales, impresoras, documentos, grupos de personas o cualquier cosa que represente cualquier entrada dada en el árbol (o múltiples entradas), como por ejemplo las credenciales necesarias para autenticarse en un sistema.

Los esquemas de LDAP son modulares, esto significa que se pueden cargar más o menos objetos y atributos dependiendo de las necesidades, normalmente a través de un archivo de configuración. Algunos de los objetos y atributos manejados pueden ser de uso obligatorio mientras que otros pueden ser opcionales, todos ellos definidos a través de la Clase que los contiene.

Las Clases son colecciones de atributos predefinidos, que se pueden importar. Si uno de los atributos necesarios no está en la Clase que se está utilizando, se puede cargar otra clase además de la primera, sumándose los nuevos atributos a los que ya estaban. Puede darse el caso de que haya clases incompatibles y no puedan utilizarse al mismo tiempo.

Para poblar el directorio LDAP se pueden emplear ficheros LDIF. En estos ficheros se describe la información de forma legible, los atributos y su correspondiente valor. También son útiles para hacer backups e importar/exportar los elementos de un árbol a otro, facilitando enormemente la gestión de la información contenida en el directorio. Así, por ejemplo un LDIF que inlcuyera una cuenta de usuario para OpenStack tendría la forma siguiente:

```
1   version: 1
2   dn: cn=user1,ou=Users,dc=mydomain,dc=com
3   objectClass: inetOrgPerson
4   objectClass: organizationalPerson
5   objectClass: person
6   objectClass: top
7   cn: user1
8   sn: user1
9   userPassword:: e1NIQX01ZW42RzZNZXpScm9UM1hLcWtkUE9tWS9CZlE9
```

user1.ldif

Por el contrario, si en vez de un usuario individual se correspondiera con un grupo de usuarios, el LDIF contendría la siguiente secuencia:

```
1   version: 1
2   dn: cn=grupoLAB1,ou=UserGroups,dc=mydomain,dc=com
3   objectClass: groupOfNames
4   objectClass: top
5   cn: grupoLAB1
6   member: cn=admin2,ou=Users,dc=mydomain,dc=com
7   member: cn=user1,ou=Users,dc=mydomain,dc=com
8   ou: Grupo LAB1
```

grupoLAB1.ldif

Nótese que *member* es un campo multi-value, es decir, permite listar varios valores, y que en este caso contiene a los usuarios para un proyecto concreto de OpenStack.

Las búsquedas en el directorio se hacen a través de filtros, lo cuales funcionan con operadores lógicos para construir filtros más complejos, como los que se muestran a continuación.

```
9   (uid=testuser)
10  Compara a todos los usuarios que tienen exactamente el valor testuser para el atributo uid.
11  (uid=test*)
12  Compara a todos los usuarios que tienen valores para el atributo uid que empiezan por test.
13  (!(uid=test*))
14  Compara a todos los usuarios que tienen valores para el atributo uid que no empiezan por test.
15  (&(department=1234)(city=Paris))
16  Compara a todos los usuarios que tienen exactamente el valor 1234 para el atributo department
    y exactamente el valor Paris para el atributo city.
17  (|(department=1234)(department=56*))
18  Compara a todos los usuarios que tienen exactamente el valor 1234 o un valor que empieza por
    56 para el atributo department.
19  (&(department=12*)(!(department=123*)))
20  Compara a todos los usuarios que tienen un valor que empieza por 12, pero no empieza por 123
    para el atributo department.
```

Ejemplos de filtros

Las operaciones principales disponibles en LDAP son las siguientes:

- **Bind**: Conexión y autenticación.
- **Unbind**: Desconexión.
- **Search**: Búsqueda.
- **Modify**/**Add**/**Delete**: Modificar/añadir/eliminar una entrada.
- **Modify RDN**.
- **Compare**: Comprobar si una entrada tiene pareja atributo/valor.
- **Abandon**: Cancelar una petición pendiente.

Por su parte, las URL de LDAP son formadas de acuerdo con la siguiente estructura:

```
➢  ldap://host:port/DN?attributes?scope?filter?extensions
```

Estructura URL para LDAP

Así, por ejemplo, *"ldap://ldap.example.com/cn=John%20Doe,dc=example,dc=com"* se refiere a todos los usuarios en la entrada de *John Doe* en *ldap.example.com*, mientras *"ldap:///dc=example,dc=com??sub?(givenName=John)"* busca por la entrada, en el servidor por defecto (notar la triple barra inclinada, omitiendo el host, y la marca de doble pregunta, omitiendo los atributos). Así como en otras URL, los caracteres especiales deben ser codificados con signos de porcentaje.

2.5. TLS

Transport Layer Security y su antecesor Secure Socket Layer (SSL) son los protocolos que proporcionan un canal de comunicación seguro a través de criptografía asimétrica. Esta capa se coloca en la capa de Transporte de la pila de protocolos OSI, justo por encima del protocolo TCP.

En principio es el protocolo que proporciona autenticación y privacidad de la información extremo a extremo en una comunicación sobre internet, para lo cual hace uso de RSA[2].

En este trabajo se van a generar las claves necesarias para el establecimiento de las comunicaciones mediante OpenSSL, que es la implementación libre del protocolo. En concreto se ha utilizado la versión TLS 1.2, aunque TLS 1.3 se está empezando a ser implementado.

La justificación del uso de TLS para securizar la comunicación entre el servicio de identificación de OpenStack y el servidor LDAP reside en el hecho de que las actuales implementaciones basadas en SSL comienzan a ser consideradas en algunos círculos como débiles, por lo que se ha buscado una solución lo más robusta posible.

[2] En criptografía, RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10^{200}, y se prevé que su tamaño crezca con el aumento de la capacidad de cálculo de los ordenadores.

Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada.

3. Aspectos prácticos

A continuación, se van a describir las principales actuaciones y aportaciones que se han llevado a cabo en este trabajo. Al ser eminentemente práctico, la exposición pretende hacer totalmente reproducible cada uno de los procesos y soluciones planteadas.

3.1. Definición del problema

De acuerdo con lo indicado en capítulo de introducción, el objetivo principal es conseguir la autenticación en OpenStack a través de un servidor LDAP. Como primera aproximación, para cumplir con lo anterior va a ser necesario disponer de un servidor LDAP y de un host que ejecute OpenStack.

Como servidor LDAP se utilizará el paquete *openldap-servers* y se instalará en una máquina con *CentOS7 - minimal*, ya que no necesita muchos recursos y puede ser virtualizada mediante una VM en *VirtualBox* de solo 1024 MB de RAM, 1 núcleo de procesador a 2.20 GHz y 8 GB de almacenamiento para la instalación. De esta configuración, lo más importante es la configuración correcta de conexión de Red, la cual va a hacer uso de un adaptador puente, esto es, para que vea al mismo nivel de red que el resto de máquinas virtuales del equipo anfitrión, de las cuales la que más interesa es la que contenga OpenStack. En este punto, es recomendable crear una conexión de red con dirección IP estática desde el instalador de CentOS7. De esta manera se habilita la interfaz y se puede configurar desde una consola, facilitando la configuración de red.

En cuanto a OpenStack, el sistema requiere de más recursos, cuantos más mejor. Por ello, para esta máquina se recomienda hacer una instalación real de *CentOS7 - minimal*, aunque también es posible hacer uso de una máquina virtual, siempre que se disponga de suficientes recursos. En concreto, la configuración utilizada hace uso de 12 GB de RAM, 8 núcleos (con *HyperThreading*) a 4.00 GHz y 250 GB de almacenamiento. Dichos recursos pueden ser optimizados dependiendo del número de máquinas virtuales desplegadas en OpenStack para las pruebas, y dependiendo de los módulos de OpenStack que se estén ejecutando, aunque el valor de la memoria RAM suele ser el punto crítico. Al igual que con la máquina virtual para el servidor LDAP, es recomendable hacer la configuración de la interfaz de red durante la instalación del SO *CentOS7*.

Entre los objetivos indicados, la instalación de OpenStack sobre el hipervisor XenServer requiere de la instalación de XenServer sobre el host, y tener en cuenta algunas opciones durante la instalación de SO. Sobre XenServer se pueden instalar tanto DevStack como Mirantis OpenStack, además de las instalaciones manuales de OpenStack. Sin embargo, estas últimas se vuelven demasiado complejas y tienen requerimientos especiales de las VM donde se instalan, como por ejemplo, que sean PV (Paravirtualization), es decir, que compartan parte de la memoria RAM y kernel con el host, no siendo completamente virtuales.

Una vez se tienen ambos servidores, el escenario que se plantea es que OpenStack confíe en el servidor LDAP como servicio de autentificación. Una vez esto se consiga, esta conexión tiene que ser asegurada para que pueda defenderse frente a ataques externos y no se roben, por ejemplo, las credenciales de los usuarios.

La mejor forma de comprobar el correcto funcionamiento del sistema planteado es abordar un caso práctico, para lo cual se procede a la definición de una configuración de un laboratorio virtual que sirva de ejemplo:

LAB_VIRT_1: Se define como un laboratorio virtual en el que el alumno dispondrá de un proyecto soportado por OpenStack en el que haya configurado dos equipos (máquinas virtuales) localizados en redes distintas, representado respectivamente a la red en donde se encuentra un atacante, y la red en la que se encuentra la víctima. Las máquinas en concreto montan los sistemas Kali Linux y Metasploitable. Kali es una distribución basada en Linux que incluye una suite de programas especializados en el análisis, detección y ejecución de ciberataques. Por su parte, Metasploitable es otra distribución también basada en Linux, pero en este caso, preparada para servir como víctima propiciatoria, al incluir versiones inseguras de los principales servicios. La combinación de ambos sistemas completa una configuración típica para llevar a cabo el aprendizaje sobre las técnicas de análisis de seguridad de redes (también denominado Pentesting). El proceso de generación del laboratorio puede resultar largo y complejo para un alumno poco experimentado, ya que supone la necesidad de montar ambas máquinas sobre OpenStack, crear las redes virtuales necesarias y asegurar que tienen conexión entre ambas, antes de poder comenzar con el aprendizaje en sí. Es precisamente por esta razón por la que el laboratorio se plantea como un proceso semiautomático, al menos en su configuración, de forma que un alumno pueda acceder directamente al proyecto de OpenStack ya pre configurado. Para ello, en este trabajo, se ha procedido a definir un ejemplo de script que permita configurar todo el proyecto automáticamente para cada alumno que así lo necesite.

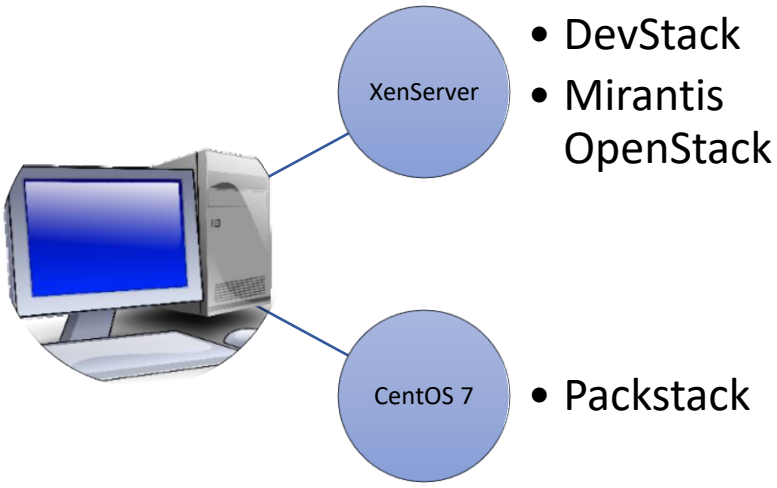
Por último, y como respuesta al principal objetivo del trabajo, el alumno debe poder utilizar todas las máquinas del proyecto sin necesidad de acceder a la interfaz web Horizon (noVNC), y sin estar físicamente situado, siquiera, en la red de OpenStack, ya que realmente podrá acceder desde un equipo situado bien en uno de los laboratorios docentes, o bien desde su propio ordenador. Para ello se establece un mecanismo de acceso al proyecto a través de conexiones SSH, autentificadas por un servidor LDAP en el que se almacenan las credenciales de acceso de los alumnos.

3.2. Requerimientos

Antes de abordar la implementación del supuesto anterior, es necesario establecer la infraestructura principal que dará soporte al laboratorio virtual, esto es, el servidor LDAP que llevará el peso de la autentificación y el servidor OpenStack, que llevará acabo las virtualizaciones. A continuación, se van a indicar la lista de requerimientos, separada por máquinas, de forma que en cada máquina se indican los recursos Hardware y Software utilizados, así como el sistema operativo y su configuración de red. Puesto que el objetivo fundamental es la comprobación del sistema de autentificación, toda la infraestructura se ha intentado simplificar al máximo, evitando el uso de múltiples equipos físicos. Es por ello que todo el sistema ha sido diseñado e implementado en dos equipos físicos, uno se ha utilizado como máquina real par la infraestructura de las virtualizaciones y en el otro se ha virtualizado una maquina en VirtualBox para la autenticación. Las características del equipo sobre el que se ha desplegado todo el sistema son:

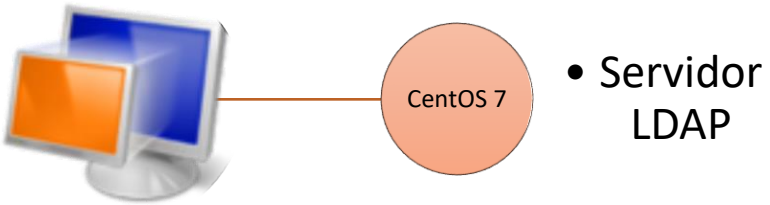
Host Real

- 8 CPU Core @ 4 GHz
- 12 GB RAM
- 250 GB HDD



Máquina Virtual (Sobre un Portátil)

- 1 CPU Core @ 2.20 GHz
- 1024 MB RAM
- 8 GB HDD



La red sobre la que se despliegan las máquinas es una red privada un CIDR 192.168.1.0/24. Tiene un rango de direccionamiento de 256 direcciones y para no interferir con otras máquinas se utilizarán direcciones entre 192.168.1.150 y 192.168.1.199.

Servidor LDAP

Aunque se recomienda utilizar valores superiores, se ha tomado una configuración de referencia considera como típica, lo cual no significa que puedan incluso bajarse las cifras dependiendo de la aplicación concreta.

La máquina virtual va a hacer uso de la siguiente configuración Hardware:

- 1 CPU Core @ 2.20 GHz
- 1024 MB RAM
- 8 GB HDD

En cuanto al Software, como sistema operativo se ha escogido un CentOS 7 con las aplicaciones mínimas, así es más ligero. Esto quiere decir que solo cuenta con las aplicaciones más esenciales y utilizadas, por lo que no dispone de una interfaz gráfica.

Además se hace uso de OpenLDAP. La suite instalada incluye:

- Slapd – Es el Daemon de LDAP (servidor)
- Librerías implementando el protocolo LDAP, y utilidades, herramientas, etc.

Esta aplicación será la base de todo el trabajo, ya que será la base de datos que hace de directorio.

Para la edición de los scripts se hará uso de Nano, el editor de texto para terminal. Como el sistema operativo sobre el que se despliega el servidor LDAP es un CentOS sin interfaz gráfica, todo se hace a través de terminal. Se puede utilizar cualquier editor, como el incluido *Vi*, o el que se prefiera, *Vim*, *EMACS*, …

Por su parte, la securización de las comunicaciones se lleva a cabo usando OpenSSL, que es el encargado de generar los certificados para encriptar las comunicaciones entre OpenStack y el servidor LDAP, en un tunel TLS 1.2. Además, nos servirá para dignosticar los problemas que puedan surgir durante la configuración de los certificados.

En cuanto a la configuración de la red en el servidor LDAP, se hace en los ajustes de red de VirtualBox y en la propia VM. En VirtualBox se selecciona Adapatador puente para que todas las maquinas se vean al mismo nivel de red, y en la VM se ha escogido la dirección 192.168.1.151 como IP. Por último, se deben abrir los puertos 389 y 636 en el firewall.

OpenStack (XenServer)

Tal como se indicó en los apartados anteriores, una de las implementaciones de OpenStack ha sido realizada sobre el sistema XenServer. En este caso, el servidor es realmente un equipo físico como tal, en un principio aprovechando las infraestructuras de virtualización existentes en el Laboratorio Docente de Telemática, 2 servidores Huawei con sistema operativo XenServer. Al final se utilizó un equipo de sobremesa personal con las siguientes especificaciones Hardware:

- 8 CPU Cores @ 4.00 GHz
- 12 GB RAM
- 250 GB HDD

OpenStack se ha instalado de dos formas diferentes, lo que conlleva requeriminntos Sofware distintos para cada instalación (DevStack y Mirantis OpenStack).

Para la instalación a traves de DevStack se necesita descargar e instalar en XenServer las XenAPI, que son unas interfaces de programación de aplicaciones que permiten intercomunicar los nodos que componen OpenStack (especialmente los nodo Compute) con el Dom0, que es donde se producen las virtualizaciones, y la parte más baja del Hardware. En concreto, os-xenapi es como se llama el paquete de XenAPI que hay que descargar e instalar en el Dom0, tras lo cual se debe proceder a descargar los scripts para la instalación de OpenStack a través de GitHub.

Para la instalación a través de Mirantis OpenStack se necesitan dos cosas, una es la ISO del Fuel Master y la otra son los plugins para hacer uso del Hipervisor:

- Mirantis OpenStack 9 ISO se trata de una imagen de la distribución del Sistema Operativo liberado por Mirantis, adaptado para hacer la instalación de OpenStack mucho más sencilla. Esta máquina se comporta como un Master, instala el SO y controla las máquinas para convertirles en nodos de OpenStack. Figura 9.
- XenServer Fuel Plugin es un accesorio que permite la utilización del hipervisor XenServer en la configuración de Mirantis OpenStack.

En cuanto a la red. También aparecen diferencias según la instalación:

- Cuando se hace uso de DevStack, por defecto la nueva máquina que contiene OpenStack, un ubuntu mínimo, es solo una y se le da una IP por DHCP.
- En la instalación a través de Mirantis OpenStack deben generarse a mano las máquinas virtuales diferenciando, por lo menos, entre Compute y Controller. En total, contando con el Fuel Master, hay que generar cuatro o cinco VM, dependiendo de si se quiere separar el Storage, una máquina específica para almacenamiento. Al igual que en DevStack la asignación de IP viene dada por el DHCP.

En todo momento se ha utilizado la versión 6.5 de XenServer, actualizada con todos los parches que se han liberado hasta el momento. Valga indicar que esta será la configuración mínima de referencia para futuras implementaciones.

OpenStack (CentOS 7)

Como alternativa a la configuración física planteada con XenServer, difícil de transportar y configurar, se optó por una configuración más portable, basada en un a máquina virtual con sistema operativo CenOS versión 7.0. Se utiliza el mismo sistema operativo que en el servidor LDAP (CentOS 7) debido al buen rendimiento de este sistema y a lo ligero que es. Además está basado en Red Hat un sistema profesional pensado para ser utilizado en grandes servidores. En este caso, se reprodujeron las mismas condiciones Hardware que en el sistema físico:

- 8 CPU Cores @ 4.00 GHz
- 12 GB RAM
- 250 GB HDD

Al contrario que en los casos anteriores, aquí no se utiliza un hipervisor alternativo como XenServer, sino que se utiliza el de por defecto, QEMU. El resto de software incluye:

- Packstack: Los paquetes de packstack contienen todos los componentes necesarios para el despliegue de OpenStack. Lo bueno de esta versión es que a pesar de no ser el Master (La versión más actualizada de OpenStack con los cambios más nuevos), cuenta con la versión “release” más reciente, en este caso denominada Ocata.
- Nano, al igual que en el servidor LDAP, en el host de OpenStack se necesitará un editor de textos, para editar los ficheros de configuración. Se puede utilizar cualquier editor de texto que funcione en el terminal.
- OpenSSL encargado de la seguridad de los mensajes entre el servidor LDAP y el servicio de identidad Keystone en OpenStack. Al igual que en el servidor LDAP se utilizará también como diagnóstico para posibles errores en la conexión entre ambas máquinas.

En cuanto a la red, al realizarse una instalación “allinone”, lo que quiere decir que todos los servicios de OpenStack estarán en la misma máquina, que todos los servicios tendrán la misma dirección IP. En este trabajo se ha utilizado 192.168.1.150.

Entorno de trabajo

Los recursos hardware del entorno de trabajo no son muy relevantes, dado que no afecta al despliegue anterior. Lo único, es que se ha utilizado el equipo de trabajo para virtualizar el servidor LDAP, pero perfectamente podría estar en otra máquina. En el desarrollo de este trabajo se ha utilizado un portátil con procesador "dual core" con "hyperthreading" y 8 GB de memoria, pero debido a los pocos recursos necesarios para ejecutar la VM de pruebas no es importante las limitaciones de hardware en para esta máquina.

VirtualBox, (Hipervisor tipo 2). Se ha virtualizado el servidor LDAP en el entorno de trabajo. Podría utilizarse cualquier otro software de virtualización como Parallels o VMWare, pero al estar disponible para múltiples sistemas operativos, hace muy sencillo el trabajar desde cualquier plataforma. La única restricción es que. para ejecutar VMs, es necesario tener los recursos hardware necesarios en el equipo que está ejecutando el software de virtualización (Host).

Putty SSH sirve para realizar una sesión SSH con las máquinas. En principio no es necesario pero es muy útil. Como el terminal no dispone de interfaz gráfica, como muchos servidores, es de mucha utilidad poder copiar y pegar en el terminal. En algunos casos hay que obtener un código identificador, o se puede copiar y pegar el contenido de un archivo, sin necesidad de transferir dicho archivo a otro equipo para editarlo, o para copiar el certificado. Todas las máquinas tienen algún sevidor SSH por defecto, así que no es necesario instalarlo a posteriori. El paquete de SSH en la mayoria de los equipos es OpenSSH.

Notepad++ es un editor de texto para Windows. Este tipo de editores colorean el texto cuando identifican el lenguaje utilizado para facilitar la lectura. En este trabajo en concreto no hay muchas ventajas ya que solo se utiliza en unos pocos ficheros, *python* y *bash*. Hay muchas opciones a la hora de elegir un editor de texto, muchas veces depende de que sea compatible con el SO. Algunos ejemplos son gedit, para linux, sublimetext, … También es posible utilizar los editores para terminal si se prefiere.

JXplorer es un programa que permite gestionar el directorio de LDAP con una interfaz gráfica, a través de peticiones y respuestas del servidor LDAP. Este programa servirá para múltiples tareas. Servirá para comprobar la existencia de entradas, gracias a que en su interfaz se construye un árbol con las entradas disponibles, también deja crear nuevas entradas, poblándolo, de una forma mucho más sencilla que escribir ficheros LDIF y añadirlos a través de comandos. También facilita mucho la creación de contraseñas, ya que dispone de una herramienta que facilita todo el proceso. Una de las ventajas de este programa es que existe para múltiples SO, tales como Windows o Ubuntu. Existen otros exploradores de directorios LDAP, como pueden ser LDAP Administration Tool o phpLDAPadmin. Pero debido a la sencillez que supone instalar y utilizar JXplorer, se utilizará este programa.

Navegador Web. Es necesario un navegador para poder acceder a Horizon, en este trabajo se ha utilizado Mozilla Firefox. La elección puede ser clave, ya que existen bastantes informes de incompatibilidades con otros navegadores, especialmente con Explorer.

Wireshark es un analizador de protocolos, que sirve para comprobar la seguridad de las comunicaciones entre el servidor LDAP y OpenStack. Servirá para localizar los paquetes que se intercambian en la comunicación TLS y comprobar que realmente se forma el túnel con la comunicación encriptada.

XenCenter es una interfaz gráfica desde la que se puede administrar XenServer. Está disponible para Windows y desde esta interfaz es desde donde se crean las VM, redes, pools, etc. Servirá para crear las máquinas virtuales cuando sea preciso, además de redes VLAN como por ejemplo la red PXE que requiere Mirantis OpenStack.

XenCenter HIMN plugin habilita ciertas funcionalidades de red que precisa Mirantis OpenStack. Básicamente crea una interfaz virtual donde conecta una VLAN.

4. Implementación

Durante el trabajo se ha desplegado OpenStack de tres formas distintas. Las dos primeras intentan hacer uso de XenServer como Hypervisor, y aunque se consigue desplegar en algunas ocasiones, no siempre que se consigue es lo suficientemente estable como para ser funcional. Estos dos intentos fueron realizados conjuntamente con otro estudiante, Martín Pereira Diéguez, el cual presentó todos los resultados en su Trabajo de Fin de Máster por lo que no se va a profundizar mucho en los detalles de ambas instalaciones. Aun así, se ha creído conveniente hacer al menos una descripción de los trabajos llevados a cabo en ambas soluciones. Por su parte, el tercer despliegue es el que mejor funciona, se ignora el Hypervisor Xen y se utiliza QEMU, el Hypervisor que viene por defecto, y es precisamente el eje central de este trabajo.

4.1. XenServer y OpenStack

En la primera tentativa se procedió a utilizar el Sistema Operativo XenServer, especializado para la virtualización, para la instalación de OpenStack como sistema gestor de Cloud, en alguna de sus versiones portables, DevStack y Mirantis.

4.1.1. DevStack

Para instalar DevStack en XenServer, se necesita cumplir algunos requerimientos [4]. La red tiene que tener acceso a internet y disponer de un DHCP, que asigne automáticamente las direcciones IP en la red, para lo cual XenServer debe estar conectado a la misma red.

Durante la instalación de XenServer, proceso al que corresponde la imagen mostrada en la Figura 12, se pide configurar, entre otras cosas, el direccionamiento de red para el servidor, indicado en la Figura 13.

Figura 12 - Instalación de XenServer

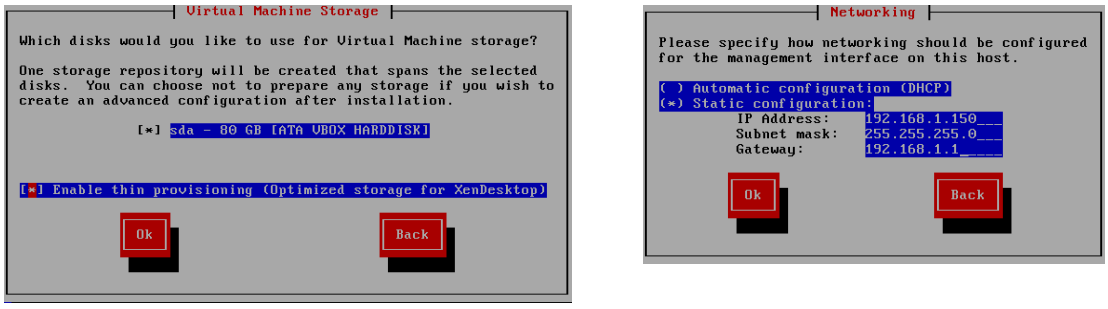


Figura 13 - Habilitar thin provisioning y utilizar una configuración IP estática

Una vez configurado el host XenServer, se descarga DevStack desde GitHub.

```
# wget --no-check-certificate https://github.com/openstack-dev/devstack/zipball/master
```

Esta descarga nos provee de los scripts necesarios para lanzar DevStack en XenServer. Todos los archivos vienen comprimidos en un archivo zip, por lo que antes deben descomprimirse para poder situar el prompt en el directorio “devsack”.

```
# unzip -o master -d ./devstack

# cd devstack/*/
```

Como paso previo a la instalación, se edita un fichero de configuración, denominado “local.conf”) en donde se establecen las contraseñas para los distintos módulos de OpenStack, redes virtuales que se van a crear y el tamaño de la VM que ejecutará OpenStack. Hay que tener en cuenta que OpenStack usa las XenAPI para comunicarse con XenServer y no queda limitado por el Hardware de la VM donde se ejecuta OpenStack. En el fichero de configuración también se establece los drivers del Host (Xen) y credenciales para acceder a él. Se pueden añadir muchas configuraciones extra.

```
# nano local.conf
```

Este fichero usa el nombre “local.conf” y reemplaza al anterior “localrc”. Básicamente es lo mismo ya que localrc es una sección de local.conf que ahora es más flexible. Se ha utilizado el siguiente localrc que recomiendan en GitHub:

```
1   # At the moment, we depend on github's snapshot function.
2   GIT_BASE="http://github.com"
3
4   # Passwords
5   # NOTE: these need to be specified, otherwise devstack will try
6   # to prompt for these passwords, blocking the install process.
7
8   DATABASE_PASSWORD=my_super_secret
9   ADMIN_PASSWORD=my_super_secret
10  SERVICE_PASSWORD=my_super_secret
11  RABBIT_PASSWORD=my_super_secret
12  SWIFT_HASH="66a3d6b56c1f479c8b4e70ab5c2000f5"
13  # This will be the password for the OpenStack VM (both stack and root users)
14  GUEST_PASSWORD=my_super_secret
15
16  # XenAPI parameters
17  # NOTE: The following must be set to your XenServer root password!
18
19  XENAPI_PASSWORD=telematica
20
21  XENAPI_CONNECTION_URL="http://192.168.1.150"
22  VNCSERVER_PROXYCLIENT_ADDRESS=192.168.1.150
23
24  # Explicitly set virt driver
25  VIRT_DRIVER=xenserver
26
27  # Explicitly enable multi-host for nova-network HA
28  MULTI_HOST=1
29
30  # Give extra time for boot
31  ACTIVE_TIMEOUT=45
32
33  enable_plugin os-xenapi https://github.com/openstack/os-xenapi.git
```

localrc

Durante el tiempo de realización del trabajo, parte del código en GitHub ha sufrido modificaciones, de forma que si se ejecuta con un fichero como el que se indica que hay que

usar en el README.md de DevStack para XenServer [5], las máquinas virtuales que se intenten crear devuelven un error en el que se indica que hay que habilitar las os-xenapi. Para evitarlo fue necesario investigar en los cambios hechos en GitHub para encontrar la forma de habilitar dichas API [6].

Este fichero tiene varias funciones, configura los XenAPI plugins, crea y nombra las redes si no existen (Figura 14), crea una VM (PV) basada en Ubuntu con el nombre DevStackOSDomU e inicia en ella DevStack (Figura 15).

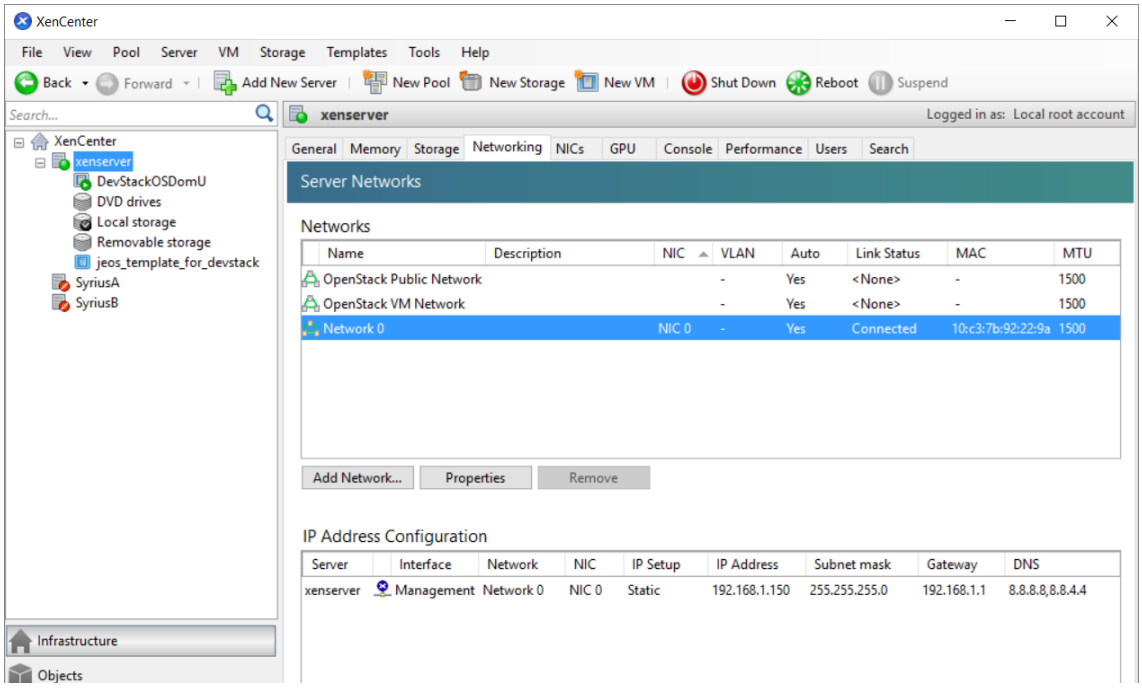


Figura 14 - Redes creadas para OpenStack

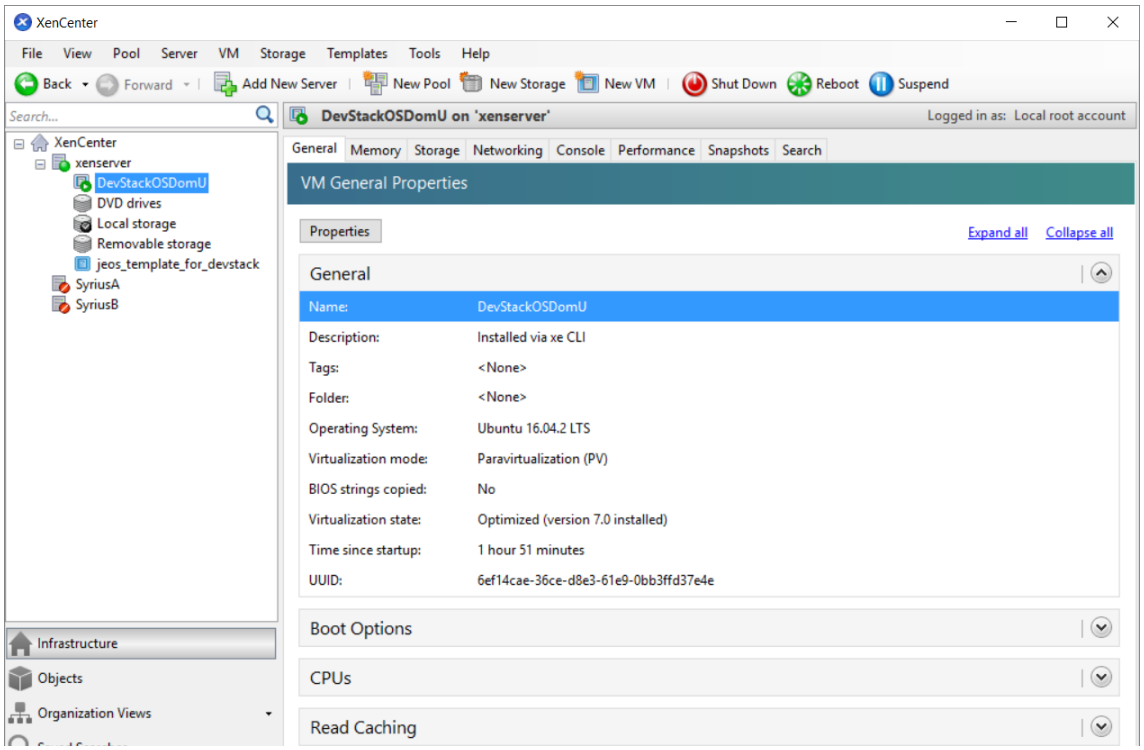


Figura 15 - VM (PV) Ubuntu con DevStack

Si todo va bien, XenServer retornara la siguiente información:

```
                                                ...
#############################################################################

All Finished!
Now, you can monitor the progress of the stack.sh installation by
looking at the console of your domU / checking the log files.

ssh into your domU now: 'ssh stack@192.168.1.43' using your password
and then do: 'sudo systemctl status devstack' to check if devstack is still running.
Check that /opt/stack/runsh.succeeded exists

When devstack completes, you can visit the OpenStack Dashboard
at http://192.168.1.43, and contact other services at the usual ports.
[root@xenserver xen]#
```

La nueva VM continuará ejecutando el script que despliega OpenStack y al final retornará un resultado parecido al siguiente.

```
For more information see:
https://docs.openstack.org/developer/devstack/systemd.html

DevStack Version: pike
Change:
OS Version: Ubuntu 16.04 xenial

2017-07-15 17:41:55.896 | stack.sh completed in 3741 seconds.
+ touch /opt/stack/runsh.succeeded
+ echo 'OpenStack VM - Installed by DevStack'
+ IPADDR=
+ echo '  Management IP:   '
+ echo -n '  Devstack run:    '
+ '[' -e /opt/stack/runsh.succeeded ']'
+ echo SUCCEEDED
+ echo ''
+ sudo cp /opt/stack/issue /etc/issue
+ rm /opt/stack/run_sh.pid
[  OK  ] Started Install OpenStack by DevStack.
[  OK  ] Reached target Multi-User System.
[  OK  ] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes...
```

Para evitar la desconfiguración de DevStack cada vez que se apaga la VM, y tener que limpiar la máquina y volver a ejecutar una instalación que puede durar entre 20 minutos y 1 hora, o

incluso más dependiendo de la conexión a internet, se toma una instancia del estado de la máquina, un “Snapshot” haciendo uso de las herramientas de XenServer, como se muestra en la Figura 16.

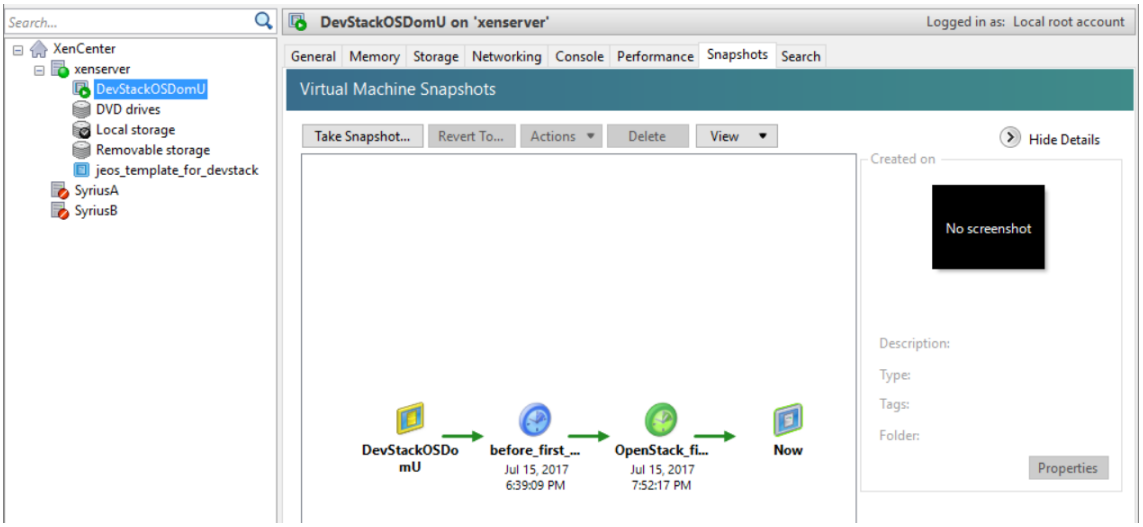


Figura 16 - Snapshots de DevStack

Se puede comprobar que realmente utiliza XenServer como hipervisor desde el usuario admin, en la pestaña de Administrador, Hipervisores, como se muestra en la Figura *17*.

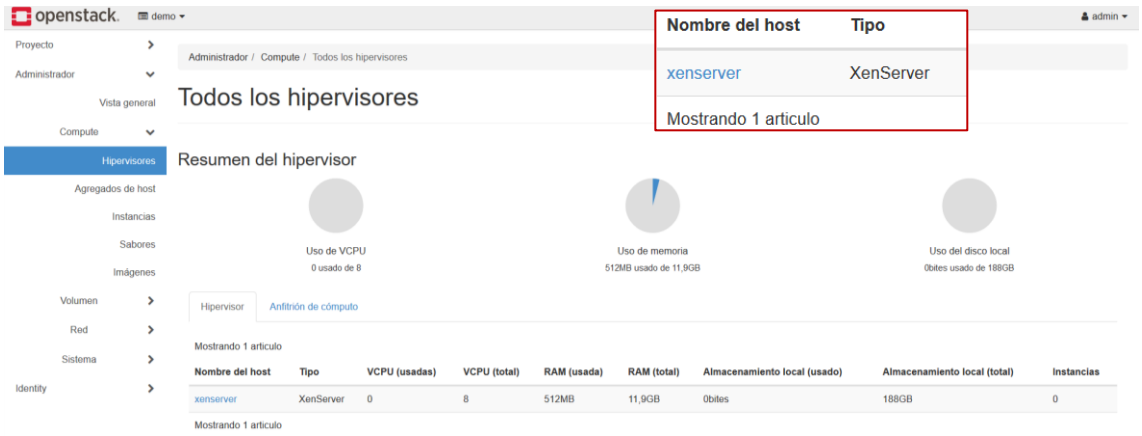


Figura 17 - Hipervisores que se utiliza con DevStack

A partir de aquí debería ser posible crear nuevas *máquinas* virtuales directamente desde DevStack, como se muestra en la Figura 18:

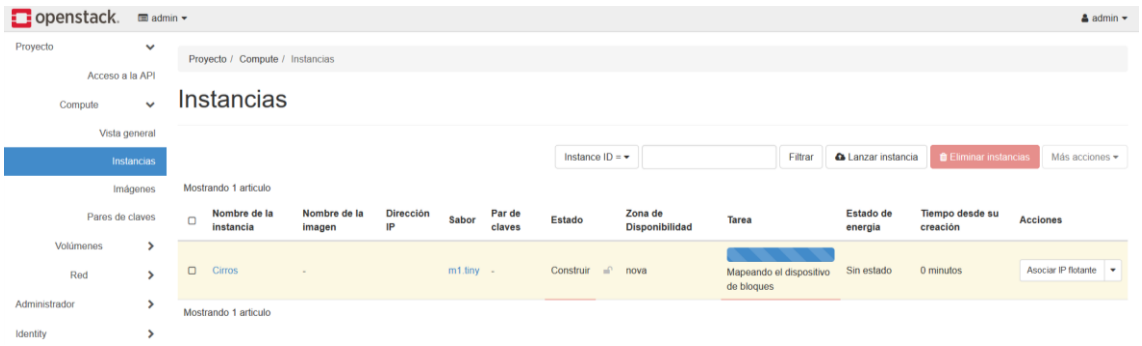


Figura 18 - Creando VM en DevStack

Sin embargo, debido a las limitaciones del hardware, especialmente por capacidades de almacenamiento, no se han llevado a cabo pruebas para todas las funcionalidades de OpenStack asociadas con esta configuración.

4.1.2.Mirantis OpenStack

Como alternativa se procedió a la instalación de OpenStack a través de Mirantis OpenStack. Este proceso se lleva a cabo desde una máquina que se especializa en el despliegue y en controlar el entorno (Environment). Esta máquina es conocida como Fuel Master, y se instala con una imagen de un sistema operativo basado en CentOS. Es en esta máquina dónde se instalan los plugin que se quieran incorporar al entorno.

Lo primero que hay que hacer es crear las redes que se van a emplear en la configuración entre los nodos, tal como se muestra en la Figura 19, y bastará con crearlas como VLAN. La primera de ellas y que dispone de una interfaz de red en cada una de las máquinas es la red PXE, que es a través de la cual se instalarán los SO a los nodos de OpenStack, y por donde se configurarán.

También se crean las otras dos redes que se utilizarán en OpenStack. Una de las redes es para la administración y otra de las redes es para el almacenamiento. Estas dos VLAN no necesitan de una interfaz en los nodos, y es importante recordar la numeración de la red VLAN.

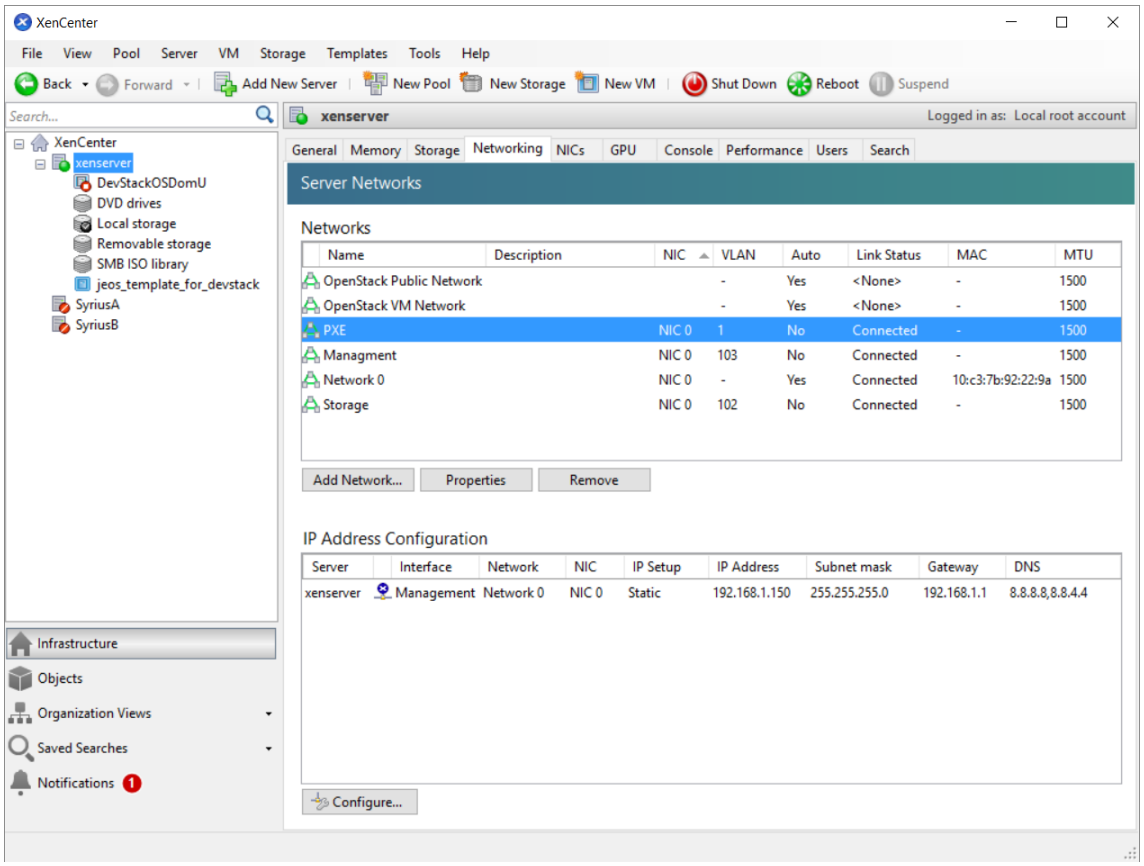


Figura 19 - Redes en XenServer

Una vez creadas las redes, se procede a la instalación del *Fuel Master*, el nodo que se encarga del despliegue y de la administración de los entornos de trabajo.

A la hora de crear la VM cabe destacar un par de puntos. El template que se selecciona al crear la nueva VM es *other*, y en Networking hay que crear una interfaz para la red PXE, como muestra la Figura 20.

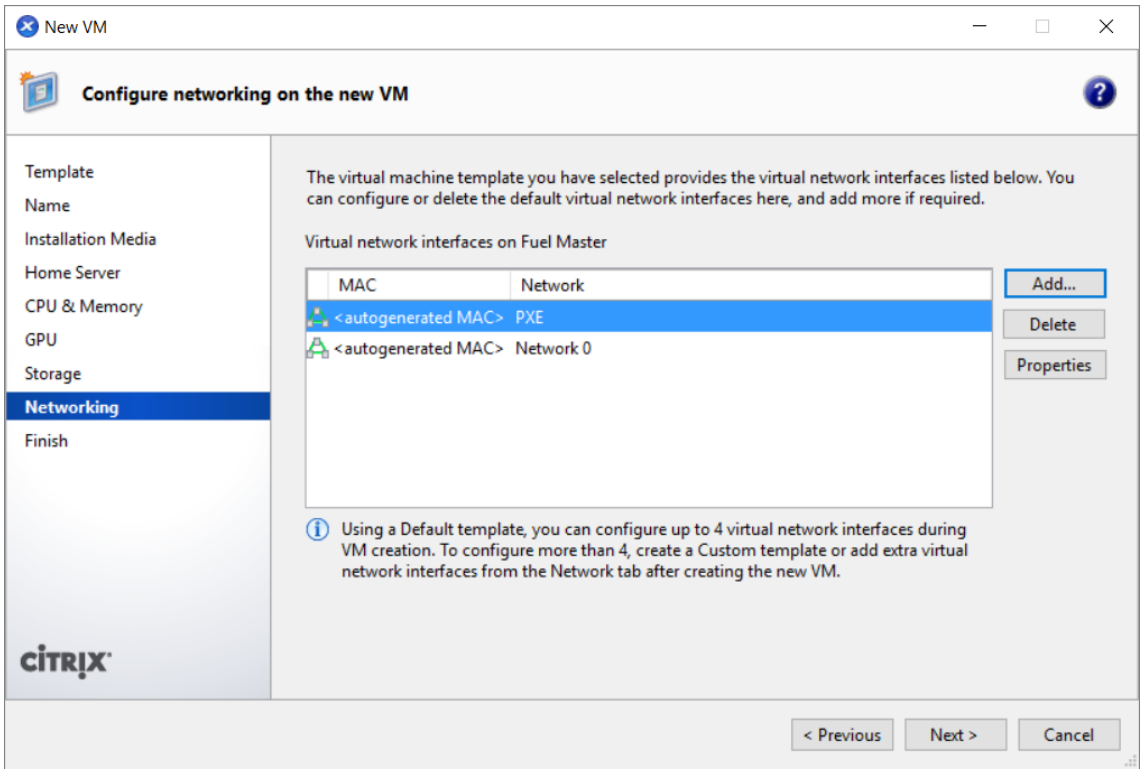


Figura 20 - Interfaces de Red para las VM

Se procede con la instalación. La instalación puede tardar un rato y después de reiniciar (habiendo expulsado la imagen para no sobrescribir la instalación) automáticamente se inicia sesión y se ejecuta el menú de configuración (fuelmenu), incluido el proceso mostrado en la figura siguiente, la creación del usuario administrador.

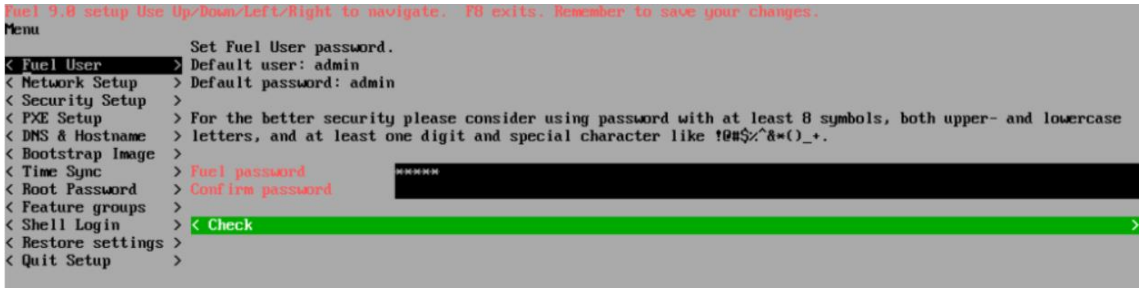


Figura 21 - Configuración del usuario de Fuel

En la configuración de red es importante establecer el direccionamiento de las dos interfaces, el de la red externa y de la red PXE, como muestran las Figuras 22 y 23 .

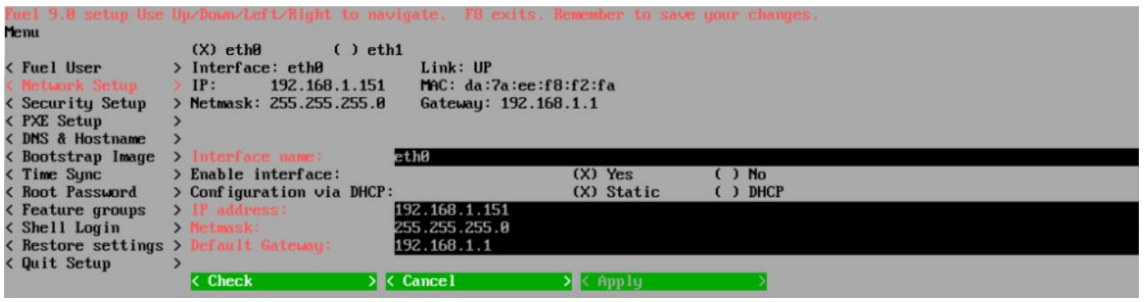


Figura 22 - Configuración de la interfaz de red externa

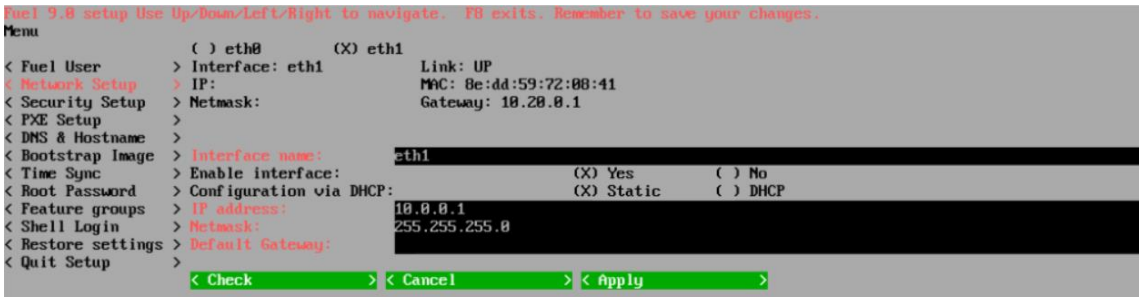


Figura 23 - Configuración de la interfaz de la red VLAN PXE

Además suele ser interesante habilitar el uso de SSH, y para ello en la sección de seguridad se establece desde qué redes es posible acceder, como muestra la Figura 24. Esto, en un despliegue real, debería estar deshabilitado o muy limitado.

Figura 24 - Configuración de seguridad (SSH)

Llegados a este punto, es recomendable comprobar la configuración de la red PXE, ya que necesita poder controlar los nodos esclavos. La Figura 25 muestra los valores utilizados durante las pruebas.

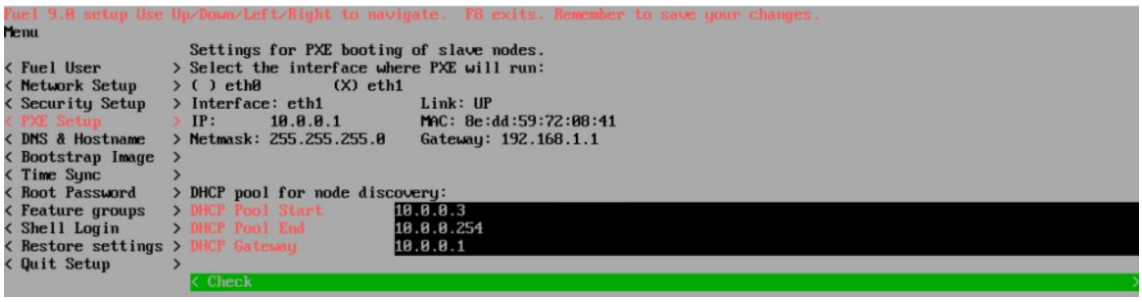


Figura 25 - Configuración de la red PXE

El siguiente paso se muestra en la Figura 26, ya que para poder acceder a la red externa es necesario configurar el DNS externo.

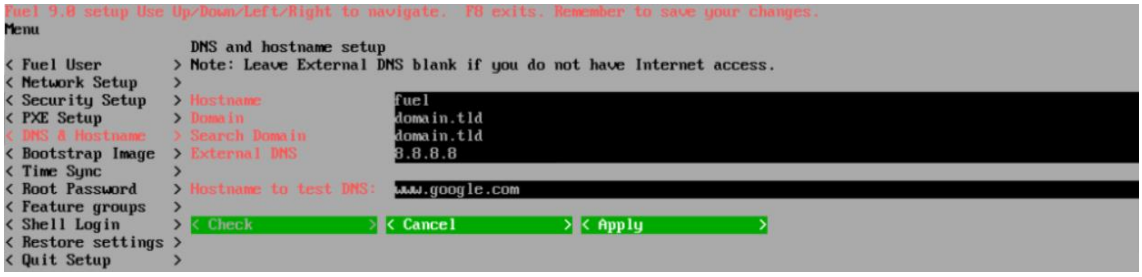


Figura 26 - Configuración del DNS y Hostname

A continuación es recomendable comprobar la configuración de la imagen Bootstrap, que se necesitará para la instalación de un SO en los nodos esclavos. La Figura 27 muestra un ejemplo de dicha configuración.

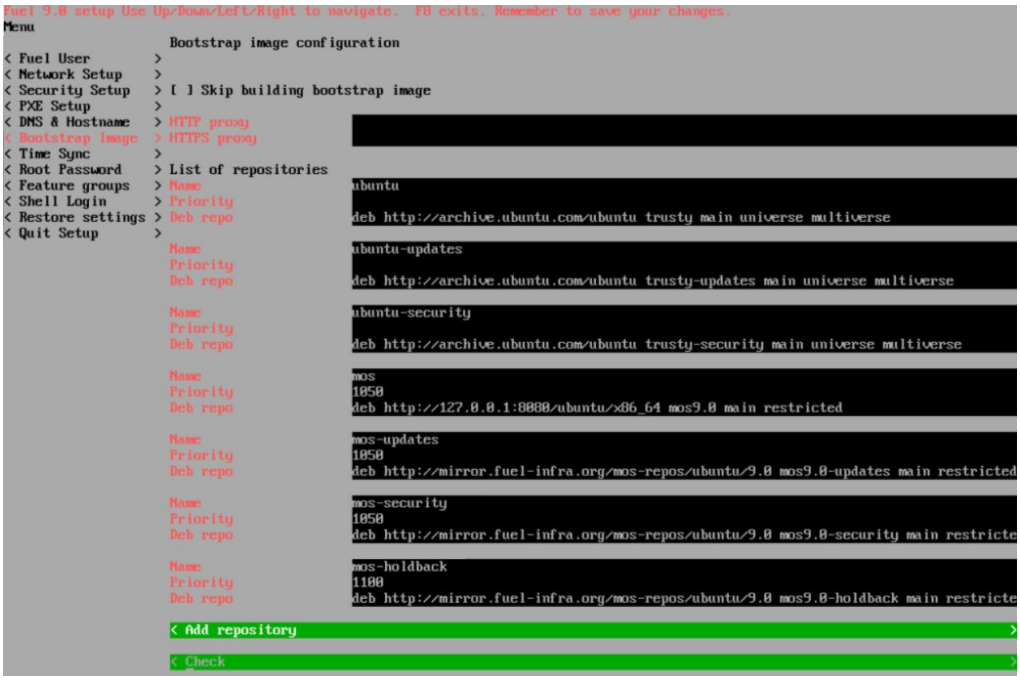


Figura 27 - Configuración de la imagen Bootstrap

Por último, tal como se ve en la siguiente figura, se guardan los cambios y se reinicia.

Figura 28 - Guardar y reiniciar

Debido a que se tiene que descargar por completo la imagen Bootstrap, el proceso puede variar en función de la velocidad de conexión.

Una vez creado el nodo Master, se posible crear el resto de nodos esclavos (Controller, Compute y Cinder). Estos nodos esclavos, al igual que el nodo máster, cuentan con una interfaz para la red PXE (véase la Figura 20), por la cual se realiza una instalación a través de la red. Para ello basta con seleccionarlo en el menú correspondiente, tal como muestra la Figura 29.

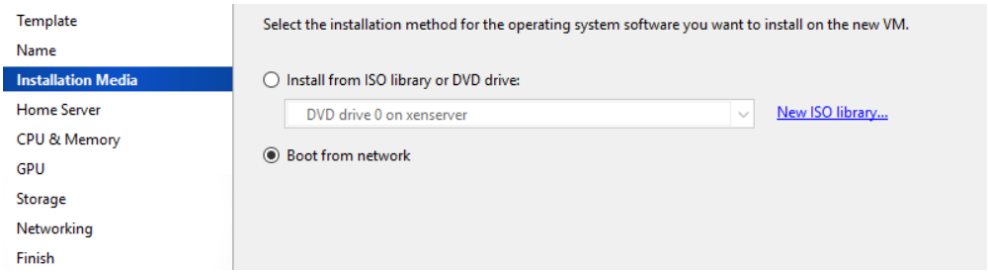


Figura 29 - Seleccionar el método por el que el SO se instalará en los nuevos nodos

Como característica especial del nodo Compute, se suele añadir una nueva interfaz, pero en esta ocasión es distinto, ya que es necesario descargar e instalar un plugin para Citrix XenCenter, llamado “SetupHIMN”, tal como se muestra en la Figura 30, el cual nos permite añadir en unos sencillos pasos una VIF (interfaz virtual).

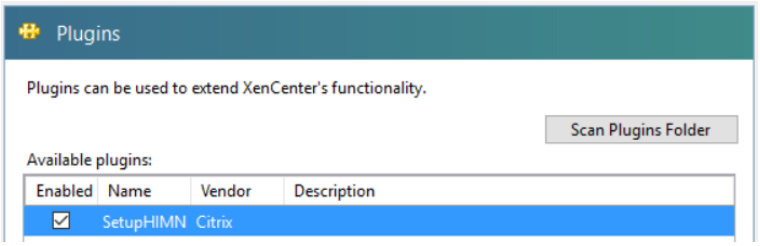


Figura 30 - Plugin de Citrix XenCenter

A continuación, seleccionando los Compute de los que se disponga, se hace clic con el botón derecho del ratón sobre uno de los nodos seleccionados y se selecciona “Manage internal managment network” en el menú desplegable. En la nueva ventana, mostrada en la Figura 31, aparecerán los nodos seleccionados y, si disponen de la interfaz VIF, permitirá añadir dicha interfaz a los nodos Compute.

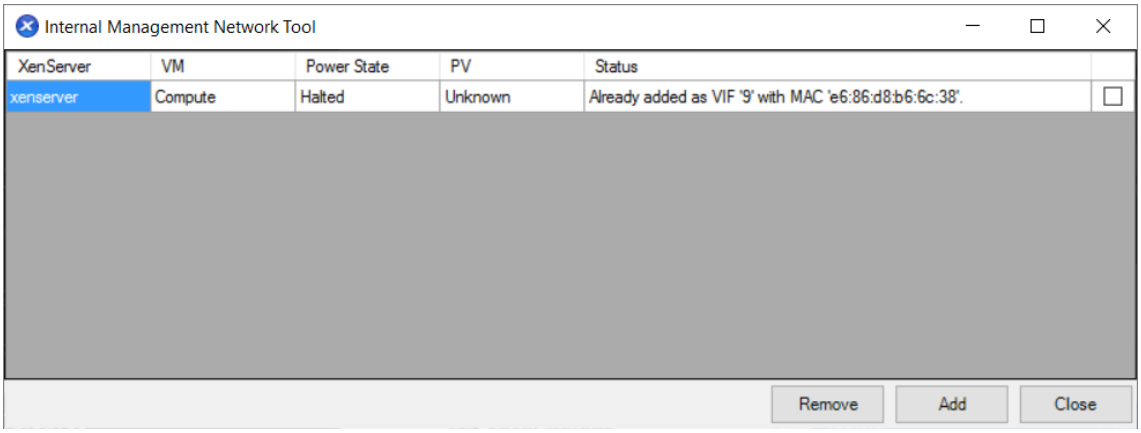
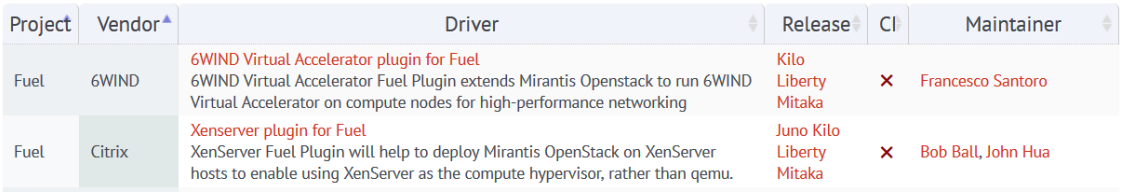


Figura 31 - Internal Managment Network Tool

Adicionalmente, para poder hacer uso de los recursos de XenServer, es necesario añadir un plugin específico para Mirantis OpenStack, sin embargo, dicho plugin ya no se puede encontrar en la página oficial, por lo que es necesario recurrir al proyecto original que permanece en GitHub. Es preciso tener en cuenta que este plugin solo soporta ciertas versiones de OpenStack, la mayoría relativamente antiguas, tal como se muestra en la Figura 32.



Project	Vendor	Driver	Release	CI	Maintainer
Fuel	6WIND	6WIND Virtual Accelerator plugin for Fuel 6WIND Virtual Accelerator Fuel Plugin extends Mirantis Openstack to run 6WIND Virtual Accelerator on compute nodes for high-performance networking	Kilo Liberty Mitaka	✕	Francesco Santoro
Fuel	Citrix	Xenserver plugin for Fuel XenServer Fuel Plugin will help to deploy Mirantis OpenStack on XenServer hosts to enable using XenServer as the compute hypervisor, rather than qemu.	Juno Kilo Liberty Mitaka	✕	Bob Ball, John Hua

Figura 32 - Soporte de XenServer Plugin

Una vez obtenido el plugin, tiene que ser subido a la VM Fuel Master y se instala con el siguiente comando.

```
# fuel plugins --install fuel-plugin-xenserver-4.0-4.0.37-1.noarch.rpm
```

```
# fuel plugins
  id | name                  | version | package_version | releases
  ---+-----------------------+---------+-----------------+----------
  1  | fuel-plugin-xenserver | 4.0.37  | 4.0.0           | ubuntu (mitaka-9.0)
```

En un navegador web se puede acceder a la interfaz UI, justo en la dirección que se ha configurado en el paso de la Figura 22. Para acceder es probable que haya que aceptar algún certificado de seguridad. Entonces aparecerá una ventana de acceso como la de la Figura 33, en la que se deberá indicar el usuario, que por defecto es admin, y lo mismo para la contraseña, salvo que se haya modificado el usuario Fuel en el paso de la Figura 21.

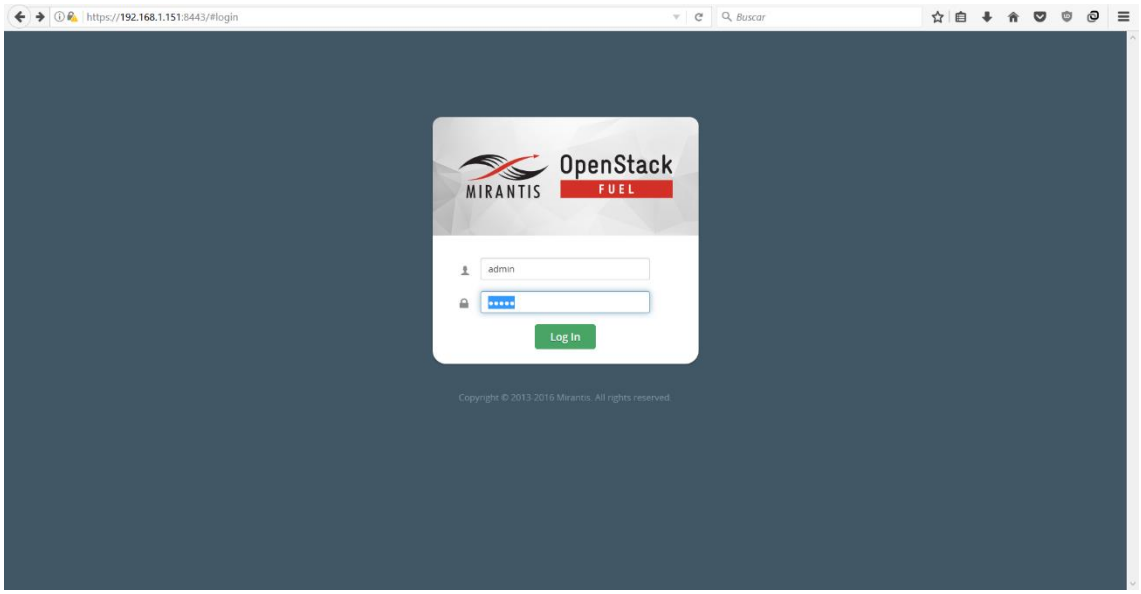


Figura 33 - Login Mirantis OpenStack

Tras la autentificación, aparece la ventana de la Figura 34, en donde se encuentran los diferentes entornos, incluidos los clústeres de nodos, pudiendo administrar múltiples despliegues. Arriba a la derecha se puede ver los nodos disponibles. Estos aparecen después de haber arrancados los nodos esclavos a través de la interfaz PXE.

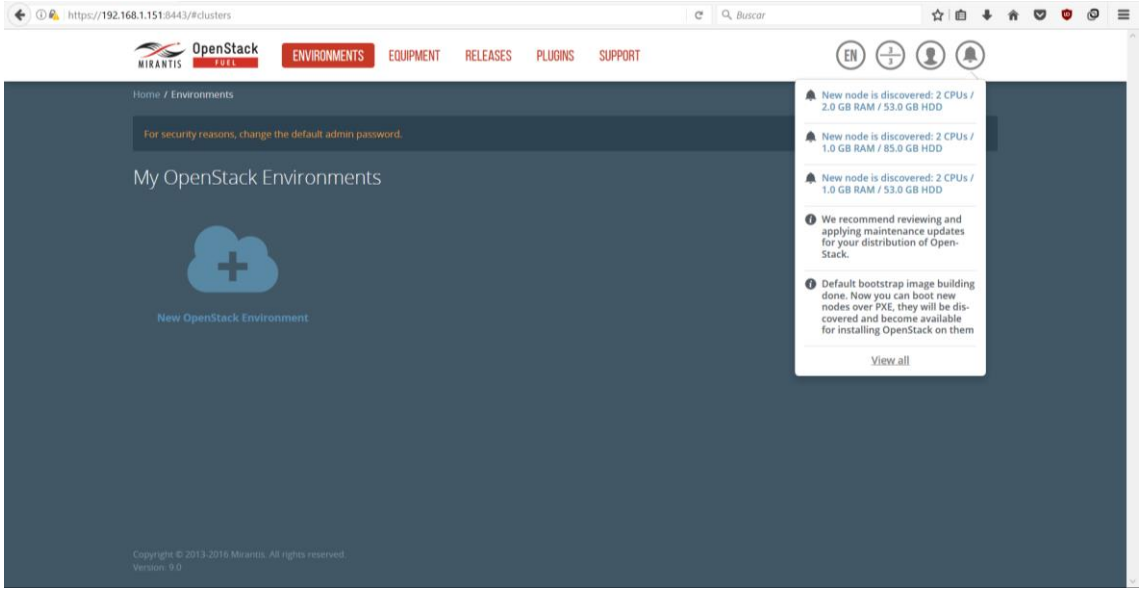


Figura 34 - Entornos OpenStack

Una de las primeras cosas que se puede hacer antes de comenzar con ningún despliegue es comprobar que el plugin de XenServer para Fuel está correctamente instalado a través de la pestaña “Plugins” en la interfaz web, tal como se muestra en la Figura 35.

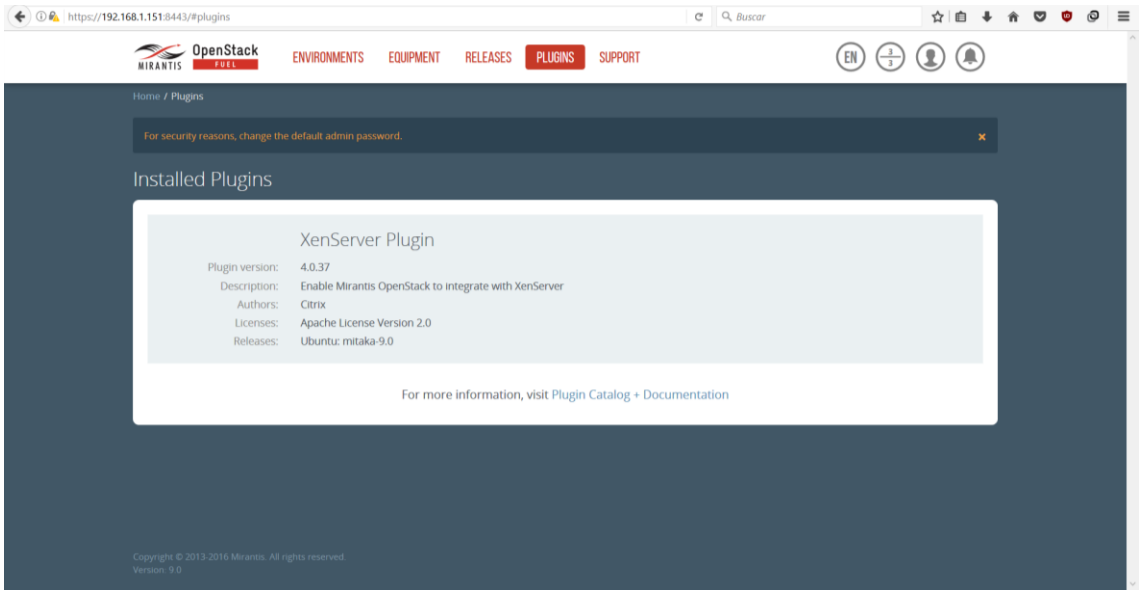


Figura 35 - Plugins Mirantis OpenStack

También se puede observar los nodos descubiertos en la pestaña “Equipment”, y que se muestra en la Figura 36. Para identificarlos se les puede dar un nombre sustituyendo el de por defecto **Untitled (XX:XX)**, donde las X representan los dos últimos octetos de la MAC de la primera de las interfaces de red, en este caso el de la red PXE.

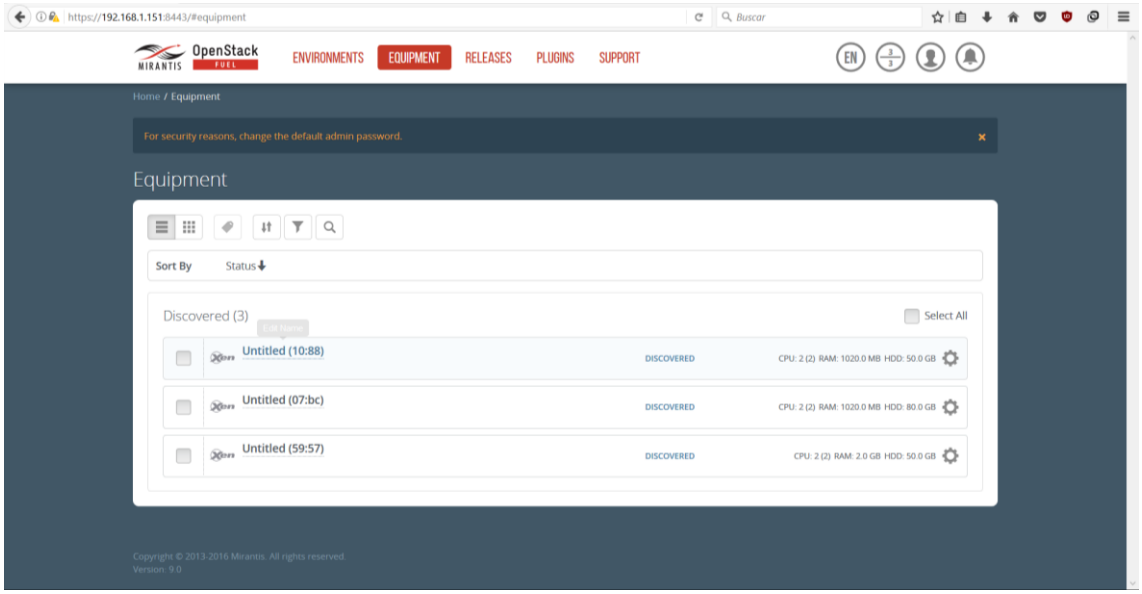


Figura 36 - Nodos descubiertos por Mirantis OpenStack

A continuación se crea un nuevo “environment” en el que se instalará “Mitaka on Ubuntu 14.04”. En las opciones de Compute no se pude seleccionar XenServer hasta que primero se deselecciona la opción de QEMU-KVM, esta opción solo aparece si el plugin está instalado, tal como se muestra en la Figura 37.

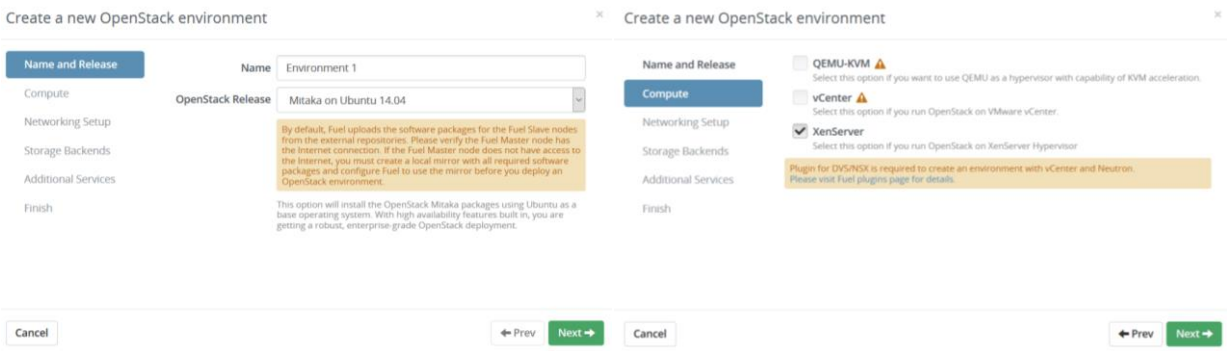


Figura 37 - Creación de un nuevo entorno

Lo primero que hay que hacer en el entorno es introducir la contraseña del host XenServer para poder manipular sus recursos, como muestra la Figura 38, y si es necesario, instalara también las API requeridas.

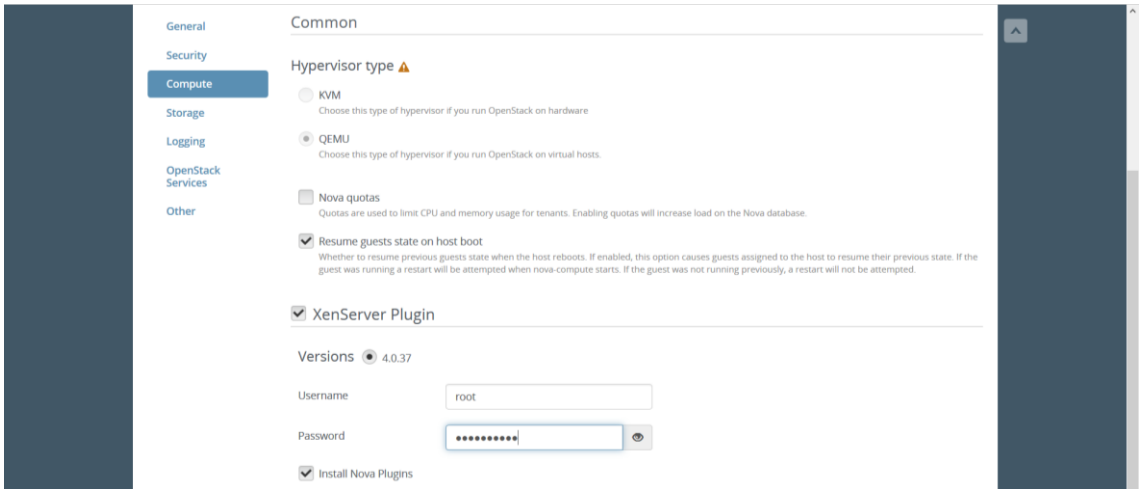


Figura 38 - Credenciales XenServer

En el caso de las redes, se configuran de una manera bastante sencilla. La configuración tiene varias partes, la primera es la configuración de la red externa, cuya ventana aparece en la Figura 39. Para la red externa se define un pool de direcciones de las que dispondrá OpenStack para asignar IP a VM en proyectos u otros nodos.

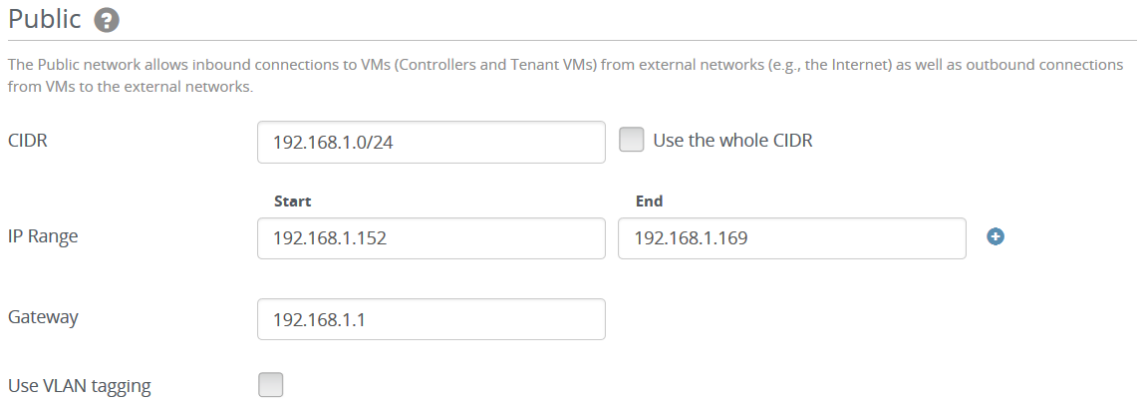


Figura 39 - Configuración de la red externa

También se configura la red para acceder al almacenamiento, a través de una VLAN. En este caso la VLAN utiliza la etiqueta con el número 102, como se puede comprobar en la Figura 19. Se configura con una red distinta que las incluidas en la red externa para evitar confusiones de que red es la adecuada.

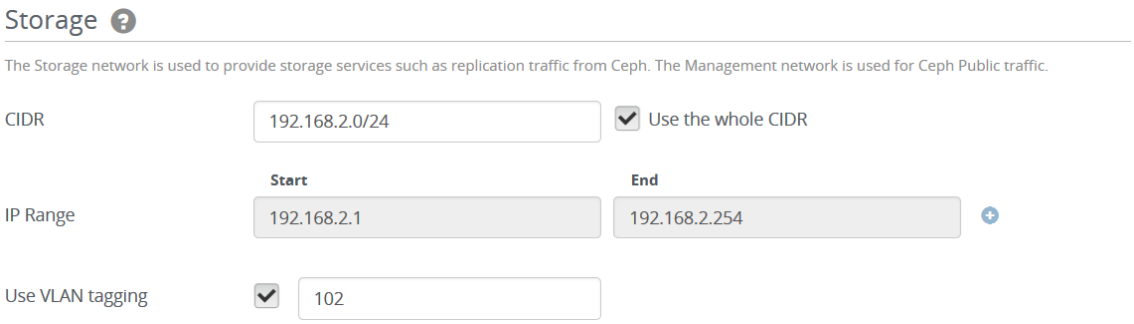


Figura 40 - Configuración de la red "Storage"

Por último. para la configuración de la red de administración se utiliza el número 103, que al igual que en el caso de la red de almacenamiento se le da un direccionamiento diferente a los anteriores y se puede comprobar el número de la VLAN en la 1.

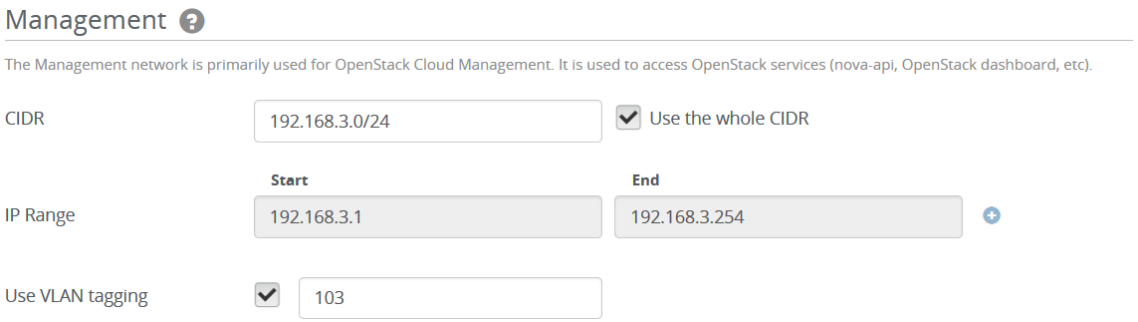


Figura 41 - Configuración de la red "Managment"

En el submenú “Neutron L3”, que se muestra en la Figura 42, se configura un pool de direcciones para la red externa que se le puede asignar de manera flotante a las VM, pero estas direcciones no deben solaparse con el rango de direcciones que se ha definido en la configuración de la red externa.

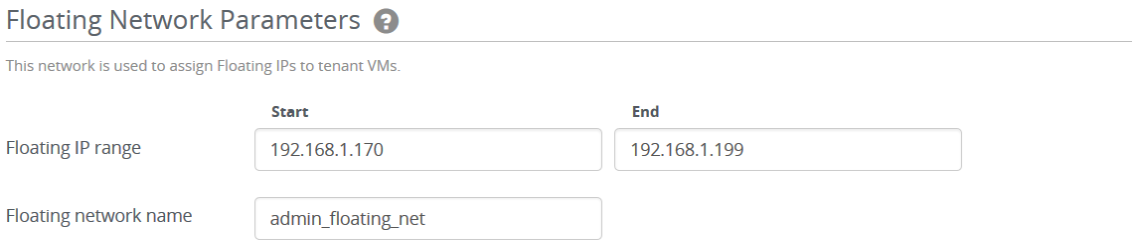


Figura 42 - Configuración de la red flotante

Por último, es recomendable ejecutar una comprobación de red, para asegurar que todas las configuraciones anteriores son correctas, que hay conectividad con los nodos y si además hay conexión con internet.

Es posible que la comprobación anterior falle debido a cómo están configuradas las interfaces en los nodos. La configuración de los nodos Controller y Cinder es como se muestra en la siguiente figura y se lleva a cabo en la pestaña de “Nodes” -> “Configure interfaces”.

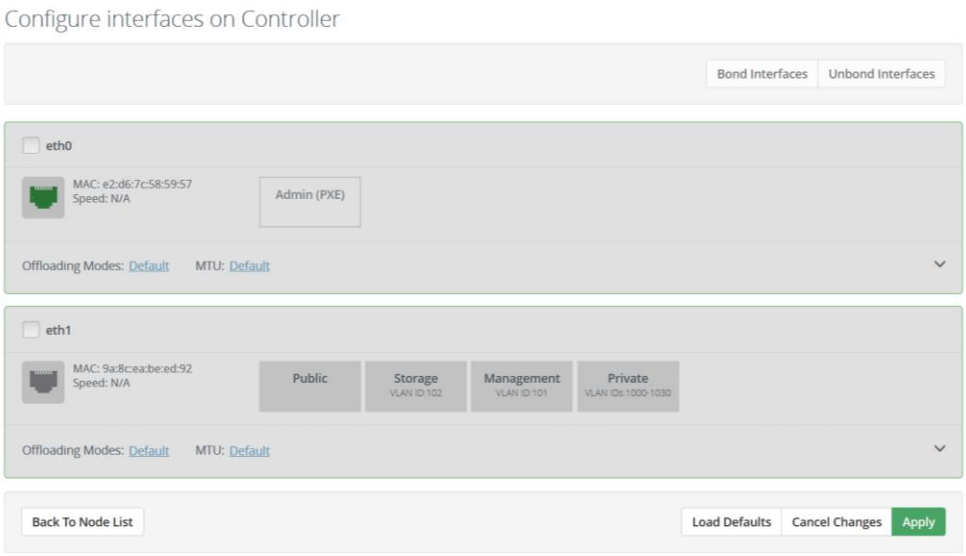


Figura 43 - Configuración de las interfaces de los nodos

El nodo de Compute es un poco especial. En este nodo se había añadido anteriormente un VIF, a través de XenCenter, y en la pestaña de la Figura 43 aparece una interfaz más, pero se puede ignorar y dejar vacía, dejando las otras dos interfaces como las de los otros nodos.

Llegados a este punto todo debería funcionar correctamente, al menos desde el punto de vista de la conectividad con Internet, con lo que podría comenzar el despliegue. Para ello, desde la primera pestaña, se le da al botón “Deploy Changes”, que como muestran las figuras 44 y 45, comenzarán con la instalación de un SO (Ubuntu) en los nodos y de la instalación de OpenStack en los mismos.

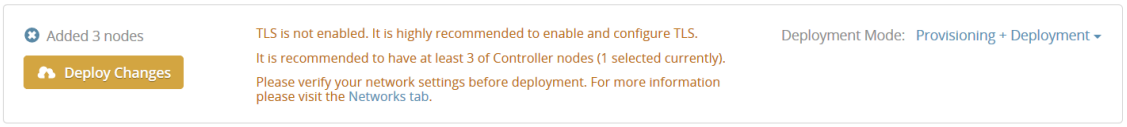


Figura 44 - Comenzar despliegue

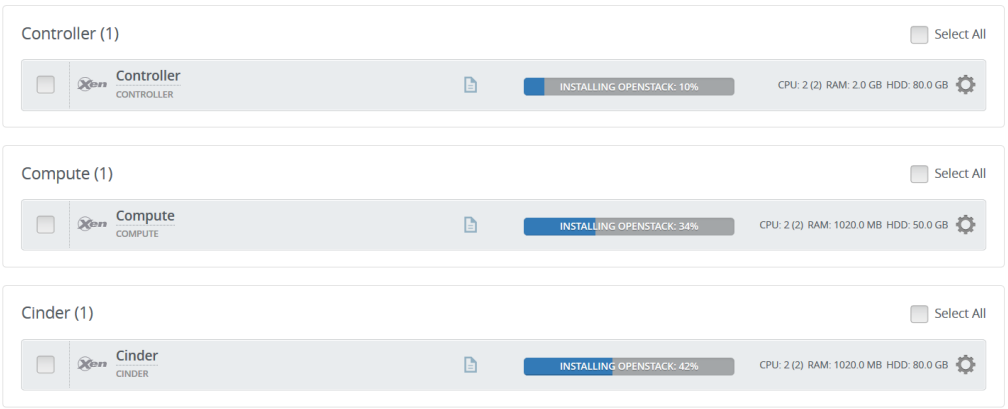


Figura 45 - Instalación y configuración de los nodos esclavos

Si no surgen errores, se debería poder usar OpenStack Mitaka a través de la IP del Controller.

4.1.3.Resultados de OpenStack en XenServer

Ambas configuraciones tienen varios problemas, y ninguna de las dos sirve para el uso continuado de los servicios de virtualización de OpenStack:

- En el caso de DevStack el mayor problema es la conservación del estado de la VM. Mantener ejecutándose los procesos todo el tiempo no permite reiniciar procesos específicos, especialmente importante cuando es necesario modificar los archivos de configuración para modificar el comportamiento de ciertas características de OpenStack, como es el caso de lo que se intenta hacer en este trabajo (que utilice LDAP como sistema autentificador). Además, cada vez que se produzca un cambio es obligatorio realizar un Snapshot de la VM DevStack que, hasta donde se ha observado, no siempre funcionan correctamente o bien no incluyen todas las funcionalidades.
- En el caso de Mirantis OpenStack la instalación de los nodos es más sencilla, pero el tratar con plugins y paquetes para configurar XenServer no lo hace nada sencillo, y provoca que falle muchas veces durante la instalación de OpenStack en los nodos. Al igual que en DevStack el funcionamiento o el rendimiento es bastante errático, especialmente cuando aumenta el tiempo de uso.

Para continuar con este trabajo se abandona la idea de utilizar OpenStack como supervisor sobre XenServer como hipervisor y se decide utilizar el hipervisor de OpenStack por defecto QEMU, sobre un host con CentOS 7. De esta forma OpenStack puede acceder a todos los recursos a través de CentOS 7, que se encuentra instalado sobre la máquina real y no en una VM.

4.2. CentOS y OpenStack

Debido a los problemas que suponía utilizar soluciones como DevStack o Mirantis OpenStack en conjunción con XenServer, se ha decidido utilizar el hipervisor por defecto en OpenStack, QEMU.

Para instalar OpenStack se ha optado por Packstack, una solución que está disponible para Red Hat y CentOS. Además de contar con las versiones más nuevas y de ser muy estable, contar con una gran comunidad por detrás que ofrece mucha documentación. Es por todo eso que Packstack es ampliamente utilizado, en muchos ámbitos diferentes.

Como ya se ha dicho anteriormente Packstack es una utilidad de instalación, basada en comandos, que utiliza Puppet para desplegar OpenStack haciendo uso de un fichero de configuración. Packstack es adecuado tanto para nodos individuales, para realizar pruebas, como para instalaciones más complejas en las que existen múltiples nodos. En este trabajo se va a utilizar un solo nodo para realizar las pruebas.

4.2.1. Instalación de CentOS 7 y particionamiento del disco

La versión de CentOS que se va a instalar es la 7, con interfaz gráfica y solo con un mínimo de aplicaciones muy utilizadas, como un servidor SSH, o un editor básico como Vi.

Al igual que con el servidor LDAP, se asigna un nombre al host, *packstack*. También se configura la interfaz de red como una IP estática, en este caso la 192.168.1.150, tal como se muestra en la Figura 46, y se activa para comprobar que está bien configurada y así no tener que configurar y habilitar desde los ficheros de configuración.

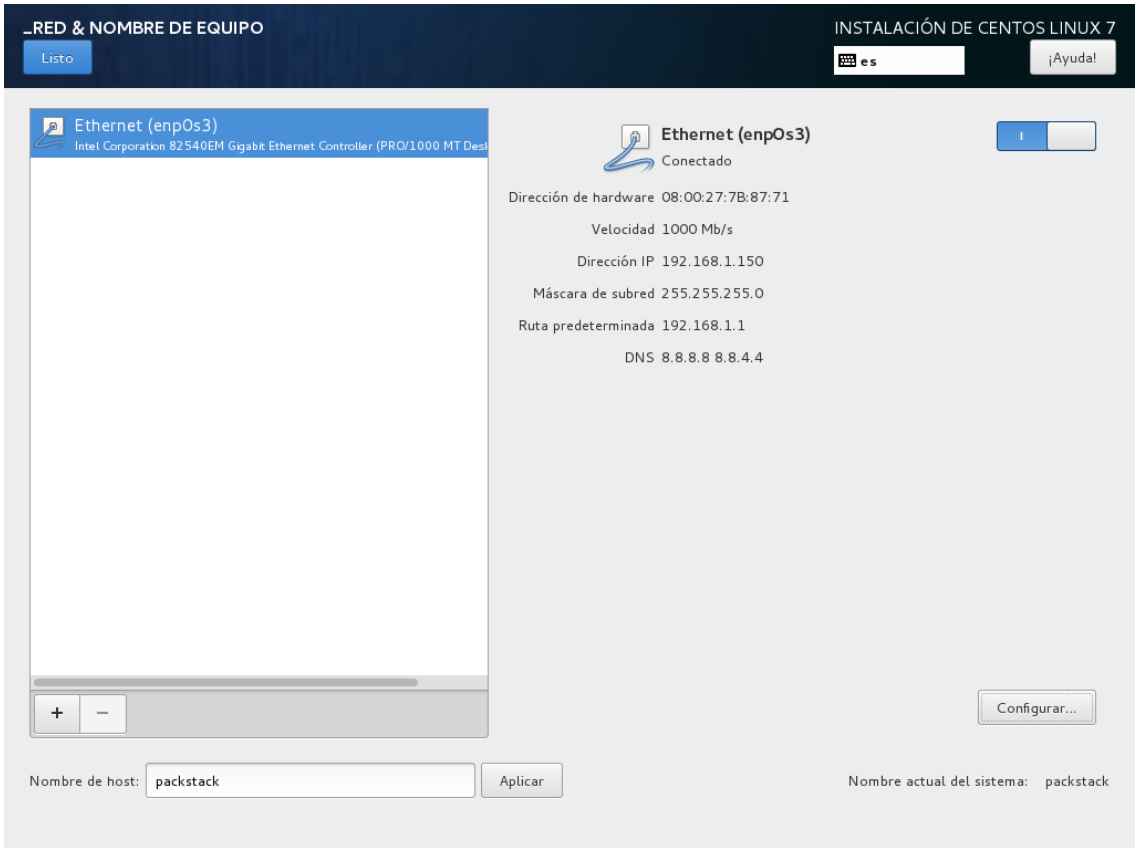


Figura 46 - Configuración de red del host OpenStack

Hay otra configuración a la que atender durante la instalación. En el host que se ha usado para las pruebas se disponía de un disco Duro de unos 230 GiB. Si se ejecuta la instalación con el resto de la configuración por defecto se crearía una partición para el usuario root de 50 GiB, lo que a su vez limitaría lo que se puede almacenar en OpenStack, ya que Packstack se instala desde el root y solo podría utilizar reservado para es partición.

```
#  lsblk

NAME                  MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                     8:0    0   2,7T  0 disk
├─sda1                  8:1    0   128M  0 part
└─sda2                  8:2    0   2,7T  0 part
sdb                     8:16   0 232,9G  0 disk
├─sdb1                  8:17   0     1G  0 part /boot
└─sdb2                  8:18   0 231,9G  0 part
  ├─cl_packstack-root 253:0    0    50G  0 lvm  /
  ├─cl_packstack-swap 253:1    0   5,9G  0 lvm  [SWAP]
  └─cl_packstack-home 253:2    0   176G  0 lvm  /home

            . . .

            . . .

sr0                    11:0    1  1024M  0 rom
```

Como se puede comprobar con el anterior comando, *lsblk*, se listan los dispositivos de almacenamiento y sus particiones. La instalación se ha llevado a cabo en el dispositivo *sdb* siendo la primera partición para el arranque, y la segunda partición donde se almacenan todos los datos, con tres particiones más.

El problema surge con la primera de las particiones, la partición "/". Esta es la raíz del sistema de archivos en Linux, todo cuelga de ella y está limitado a 50G. Sin embargo, la partición automática de los discos durante la instalación le asigna la carpeta "/home" a otra partición con 176G. En esta carpeta se supone que se guarda información de los usuarios, pero no de OpenStack. A través de la interfaz de configuración de CentOS, es posible configurar un nuevo esquema de particiones, tal como se muestra en la Figura 47.

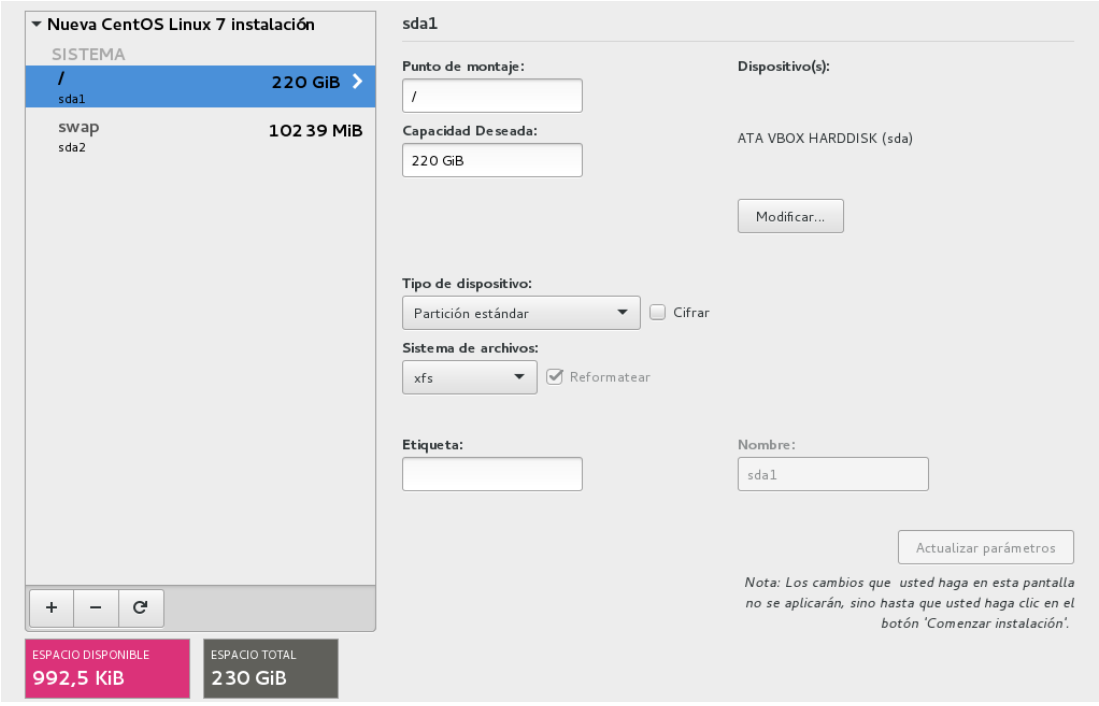


Figura 47 - Particionamiento del disco duro para OpenStack

Una vez se hace una partición manual de los discos se puede ejecutar de nuevo el mismo comando para comparar las particiones creadas y comprobar que la partición "/" tiene disponible todo el espacio de almacenamiento.

```
# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0   2,7T  0 disk
├─sda1   8:1    0   128M  0 part
└─sda2   8:2    0   2,7T  0 part
sdb      8:16   0 232,9G  0 disk
├─sdb1   8:17   0   220G  0 part /
└─sdb2   8:18   0  12,9G  0 part [SWAP]
           . . .
           . . .
sr0     11:0    1  1024M  0 rom
```

A partir de este momento ya se puede utilizar una conexión SSH para conectarse con la máquina. Al igual que en el caso del servidor LDAP, un terminal con SSH no permite copiar y pegar textos y comandos de forma muy sencilla. Esto es más sencillo que montar un servidor ftp en cada una de las máquinas CentOS para poder copiar el certificado, ya que teniendo en cuenta que éste es un texto plano, puede ser copiado y pegado en otro archivo sin ningún tipo de problema. Además, para utilizar en algunos comandos los identificadores alfanuméricos que utiliza OpenStack es más sencillo copiar y pegar que escribirlos a mano.

4.2.2.Instalación de Packstack y despliegue de OpenStack

Lo primero que se hace es actualizar el sistema e instalar un editor de texto en modo terminal, en este caso nano.

```
# yum -y update
# yum -y install nano
```

Una vez se tiene el sistema preparado, se procede a instalar los paquetes de Packstack para CentOS. La versión que se va a instalar es "Ocata" que es la última versión en este momento.

```
# yum -y install centos-release-openstack-ocata
# yum -y update
# yum -y install openstack-packstack
```

Así quedarán instalados todos los paquetes que se necesita para desplegar OpenStack. Además, el tiempo que se tarda en obtener los paquetes es mucho menor que en el caso de DevStack. Tan solo tarda unos pocos minutos.

Si se quisiese instalar OpenStack en un único host para realizar las pruebas y con una configuración mínima, podría usarse el siguiente comando.

```
# packstack --allinone
```

Pero en nuestro caso, además de una configuración mínima se quiere configurar también la red externa, para dar internet a las VM de los proyectos. Por lo que hay que utilizar el siguiente comando. [7]

```
# packstack --allinone --provision-demo=n --os-neutron-ovs-bridge-mappings=extnet:br-ex --os-
  neutron-ovs-bridge-interfaces=br-ex:eno1 --os-neutron-ml2-type-drivers=vxlan,flat
```

En el parámetro del bridge externo se escribe el nombre de la interfaz que tiene acceso a internet. El nombre de esta interfaz pude ser diferente dependiendo del hardware de la máquina o del software, básicamente depende del Sistema Operativo. Para averiguar cuál es nombre de la interfaz se listan las interfaces de red.

```
#  ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 10:c3:7b:92:22:9a brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.150/24 brd 192.168.1.255 scope global eno1
       valid_lft forever preferred_lft forever
    inet6 fe80::76ff:f8f:537:2d3/64 scope link
       valid_lft forever preferred_lft forever
```

En este sistema la interfaz que tiene acceso a internet es la interfaz “eno1”, con una IP 192.168.1.150/24.

Hay otra forma de instalar OpenStack con Packstack que además deja controlar una gran cantidad de variables para ajustar la configuración de la instalación. Esta última forma es la que se va a utilizar, ya que no solo hay que configurar la red, sino que hay otros ajustes que precisan ser modificados, para una poder utilizar OpenStack sin muchas limitaciones. Debido a las restricciones de hardware del host, algunas funcionalidades opcionales de OpenStack no van a ser instaladas.

Todas estas configuraciones están disponibles en un archivo de configuración que se obtiene con el siguiente comando.

```
#  packstack --gen-answer-file=answerfile.txt
Packstack changed given value  to required value /root/.ssh/id_rsa.pub
```

En el fichero answerfile.txt se modifica la instalación de ciertos servicios secundarios para que liberen suficiente memoria RAM, como para después poder asignar recursos a VM en los proyectos.

Por defecto, el tamaño del almacenamiento de Cinder para los volúmenes de las máquinas en OpenStack es de tan solo *20 G*, por lo que en este caso se le ha dado *120 G*, que debería ser suficiente para el ejemplo práctico más adelante.

Se ha configurado la red externa, que creará el bridge *br-ex* y se le asignará a la interfaz *eno1*. También se ha omitido la creación de un usuario y un proyecto “demo”.

En este fichero también se establece la configuración de Keystone. En ese apartado se puede configurar un apartado para LDAP, pero se ha optado por una configuración multi-dominio. La configuración del nombre y la contraseña para el usuario administrador se realiza en la misma sección. En este caso es *admin* como nombre de usuario y *telematica* como contraseña.

```
# nano answerfile.txt
```

```
40    # Specify 'y' to install OpenStack Object Storage (swift). ['y', 'n']
41    CONFIG_SWIFT_INSTALL=n
42
43    # Specify 'y' to install OpenStack Metering (ceilometer). ['y', 'n']
44    CONFIG_CEILOMETER_INSTALL=n
45
46    # Specify 'y' to install OpenStack Telemetry Alarming (Aodh). Note
47    # Aodh requires Ceilometer to be installed as well. ['y', 'n']
48    CONFIG_AODH_INSTALL=n
49
50    # Specify 'y' to install OpenStack Metering as a Service (gnocchi).
51    # ['y', 'n']
52    CONFIG_GNOCCHI_INSTALL=n
          . . .
          . . .
333   # User name for the Identity service 'admin' user.  Defaults to
334   # 'admin'.
335   CONFIG_KEYSTONE_ADMIN_USERNAME=admin
336
337   # Password to use for the Identity service 'admin' user.
338   CONFIG_KEYSTONE_ADMIN_PW=telematica
          . . .
          . . .
562   # Size of Block Storage volumes group. Actual volume size will be
563   # extended with 3% more space for VG metadata. Remember that the size
564   # of the volume group will restrict the amount of disk space that you
565   # can expose to Compute instances, and that the specified amount must
566   # be available on the device used for /var/lib/cinder.
567   CONFIG_CINDER_VOLUMES_SIZE=120G
          . . .
          . . .
811   # Comma-separated list of network-type driver entry points to be
812   # loaded from the neutron.ml2.type_drivers namespace. ['local',
813   # 'flat', 'vlan', 'gre', 'vxlan']
814   CONFIG_NEUTRON_ML2_TYPE_DRIVERS=vxlan,flat
          . . .
          . . .
877   # Comma-separated list of bridge mappings for the OpenStack
878   # Networking Open vSwitch plugin. Each tuple in the list must be in
879   # the format <physical_network>:<ovs_bridge>. Example: physnet1:br-
880   # eth1,physnet2:br-eth2,physnet3:br-eth3
881   CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=extnet:br-ex
882
883   # Comma-separated list of colon-separated Open vSwitch
884   # <bridge>:<interface> pairs. The interface will be added to the
885   # associated bridge. If you desire the bridge to be persistent a value
886   # must be added to this directive, also
887   # CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS must be set in order to create
888   # the proper port. This can be achieved from the command line by
889   # issuing the following command: packstack --allinone --os-neutron-
890   # ovs-bridge-mappings=ext-net:br-ex --os-neutron-ovs-bridge-interfaces
891   # =br-ex:eth0
892   CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:eno1
          . . .
          . . .
1157  # Specify 'y' to provision for demo usage and testing. ['y', 'n']
1158  CONFIG_PROVISION_DEMO=n
          . . .
          . . .
1308  # Password to use for the Magnum to authenticate with the Identity
1309  # service.
1310  CONFIG_MAGNUM_KS_PW=PW_PLACEHOLDER
```

answerfile.txt

Una vez configurado el fichero de configuración answerfile.txt, se instala OpenStack usando Packstack, al que se le pasa el fichero de configuración que se ha modificado.

```
# packstack --answer-file=answerfile.txt

Welcome to the Packstack setup utility

The installation log file is available at: /var/tmp/packstack/20170727-153945-WEzJgA/openstack-
setup.log

Installing:
Clean Up                                             [ DONE ]
Discovering ip protocol version                      [ DONE ]
Setting up ssh keys                                  [ DONE ]
Preparing servers                                    [ DONE ]
Pre installing Puppet and discovering hosts' details [ DONE ]
Preparing pre-install entries                        [ DONE ]
Setting up CACERT                                    [ DONE ]
Preparing AMQP entries                               [ DONE ]
Preparing MariaDB entries                            [ DONE ]
Fixing Keystone LDAP config parameters to be undef if empty[ DONE ]
Preparing Keystone entries                           [ DONE ]
Preparing Glance entries                             [ DONE ]
Checking if the Cinder server has a cinder-volumes vg[ DONE ]
Preparing Cinder entries                             [ DONE ]
Preparing Nova API entries                           [ DONE ]
Creating ssh keys for Nova migration                 [ DONE ]
Gathering ssh host keys for Nova migration           [ DONE ]
Preparing Nova Compute entries                       [ DONE ]
Preparing Nova Scheduler entries                     [ DONE ]
Preparing Nova VNC Proxy entries                     [ DONE ]
Preparing OpenStack Network-related Nova entries     [ DONE ]
Preparing Nova Common entries                        [ DONE ]
Preparing Neutron LBaaS Agent entries                [ DONE ]
Preparing Neutron API entries                        [ DONE ]
Preparing Neutron L3 entries                         [ DONE ]
Preparing Neutron L2 Agent entries                   [ DONE ]
Preparing Neutron DHCP Agent entries                 [ DONE ]
Preparing Neutron Metering Agent entries             [ DONE ]
Checking if NetworkManager is enabled and running    [ DONE ]
Preparing OpenStack Client entries                   [ DONE ]
Preparing Horizon entries                            [ DONE ]
Preparing Puppet manifests                           [ DONE ]
Copying Puppet modules and manifests                 [ DONE ]
Applying 192.168.1.150_controller.pp
192.168.1.150_controller.pp:                         [ DONE ]
Applying 192.168.1.150_network.pp
192.168.1.150_network.pp:                            [ DONE ]
Applying 192.168.1.150_compute.pp
192.168.1.150_compute.pp:                          [ ERROR ]
Applying Puppet manifests                          [ ERROR ]

ERROR : Error appeared during Puppet run: 192.168.1.150_compute.pp
Error: Execution of '/usr/bin/yum -d 0 -e 0 -y install genisoimage' returned 1: Error
downloading packages:
You will find full trace in log /var/tmp/packstack/20170727-153945-
WEzJgA/manifests/192.168.1.150_compute.pp.log
Please check log file /var/tmp/packstack/20170727-153945-WEzJgA/openstack-setup.log for more
information
Additional information:
 * Time synchronization installation was skipped. Please note that unsynchronized time on server
instances might be problem for some OpenStack components.
 * Warning: NetworkManager is active on 192.168.1.150. OpenStack networking currently does not
work on systems that have the Network Manager service enabled.
 * File /root/keystonerc_admin has been created on OpenStack client host 192.168.1.150. To use
the command line tools you need to source the file.
 * To access the OpenStack Dashboard browse to http://192.168.1.150/dashboard .
Please, find your login credentials stored in the keystonerc_admin in your home directory.
```

Es probable que la instalación termine con errores en el despliegue del compute. El error sugiere que se comprueben los logs de Packstack durante la instalación.

```
# nano /var/tmp/packstack/20170727-153945-WEzJgA/manifests/192.168.1.150_compute.pp.log
```

```
        . . .
35  ^[[mNotice: Compiled catalog for packstack in environment production in 1.59 seconds^[[0m
36  ^[[mNotice: /Stage[main]/Main/Sshkey[ecdsa-sha2-nistp256.packstack]/ensure: created^[[0m
37  ^[[mNotice: /Stage[main]/Main/Sshkey[ssh-rsa.packstack]/ensure: created^[[0m
38  ^[[1;31mError: Execution of '/usr/bin/yum -d 0 -e 0 -y install genisoimage' returned 1: Error
    downloading packages:
39    libusal-1.1.11-23.el7.x86_64: [Errno 256] No more mirrors to try.
40    genisoimage-1.1.11-23.el7.x86_64: [Errno 256] No more mirrors to try.^[[0m
41  ^[[1;31mError: /Stage[main]/Nova::Compute/Package[genisoimage]/ensure: change from purged to
    present failed: Execution of '/usr/bin/yum -d 0 -e 0 -y install genisoimage' returned 1: Error
    $
42    libusal-1.1.11-23.el7.x86_64: [Errno 256] No more mirrors to try.
43    genisoimage-1.1.11-23.el7.x86_64: [Errno 256] No more mirrors to try.^[[0m
44  ^[[mNotice: /Stage[main]/Packstack::Nova::Compute/Exec[tuned-virtual-host]/returns: executed
    successfully^[[0m
45  ^[[1;31mError: Execution of '/usr/bin/yum -d 0 -e 0 -y install libvirt-daemon-config-nwfilter'
    returned 1: Error downloading packages:
46    gnutls-dane-3.3.24-1.el7.x86_64: [Errno 256] No more mirrors to try.
47    autogen-libopts-5.18-5.el7.x86_64: [Errno 256] No more mirrors to try.
48    unbound-libs-1.4.20-28.el7.x86_64: [Errno 256] No more mirrors to try.
49    ldns-1.6.16-10.el7.x86_64: [Errno 256] No more mirrors to try.
50    libvirt-client-2.0.0-10.el7_3.9.x86_64: [Errno 256] No more mirrors to try.
51    libvirt-daemon-driver-nwfilter-2.0.0-10.el7_3.9.x86_64: [Errno 256] No more mirrors to try.
52    cyrus-sasl-2.1.26-20.el7_2.x86_64: [Errno 256] No more mirrors to try.
53    gnutls-utils-3.3.24-1.el7.x86_64: [Errno 256] No more mirrors to try.
54    numad-0.5-17.20150602git.el7.x86_64: [Errno 256] No more mirrors to try.
55    yajl-2.0.4-4.el7.x86_64: [Errno 256] No more mirrors to try.
56    libvirt-daemon-config-nwfilter-2.0.0-10.el7_3.9.x86_64: [Errno 256] No more mirrors to try.
57    avahi-libs-0.6.31-17.el7.x86_64: [Errno 256] No more mirrors to try.
58    cyrus-sasl-md5-2.1.26-20.el7_2.x86_64: [Errno 256] No more mirrors to try.
59    libvirt-daemon-2.0.0-10.el7_3.9.x86_64: [Errno 256] No more mirrors to try.^[[0m
60  ^[[1;31mError:                /Stage[main]/Nova::Compute::Libvirt::Services/Package[libvirt-
    nwfilter]/ensure: change from purged to present failed: Execution of '/usr/bin/yum -d 0 -e 0
    -y install libvirt$
61    gnutls-dane-3.3.24-1.el7.x86_64: [Errno 256] No more mirrors to try.
62    autogen-libopts-5.18-5.el7.x86_64: [Errno 256] No more mirrors to try.
63    unbound-libs-1.4.20-28.el7.x86_64: [Errno 256] No more mirrors to try.
64    ldns-1.6.16-10.el7.x86_64: [Errno 256] No more mirrors to try.
65    libvirt-client-2.0.0-10.el7_3.9.x86_64: [Errno 256] No more mirrors to try.
66    libvirt-daemon-driver-nwfilter-2.0.0-10.el7_3.9.x86_64: [Errno 256] No more mirrors to try.
67    cyrus-sasl-2.1.26-20.el7_2.x86_64: [Errno 256] No more mirrors to try.
68    gnutls-utils-3.3.24-1.el7.x86_64: [Errno 256] No more mirrors to try.
        . . .
```

192.168.1.150_compute.pp.log

En el log aparece el problema descrito durante el paso de la instalación en la línea 38, y en las siguientes líneas aparecen los paquetes que no es capaz de descargar. De acuerdo con el *log*, es debido a que no los encuentra y no tiene más direcciones donde buscarlas. Esto sugiere un problema con la resolución de nombres de dominio, o DNS. Este problema surge al configurar el bridge *br-ex*, que copia la información de configuración en la interfaz *eno1*, pero no vuelve a configurar los DNS. Para arreglar este problema basta con introducir los DNS manualmente.

```
# nano /etc/resolv.conf
```

```
1   # Generated by NetworkManager
2   nameserver 8.8.8.8
3   nameserver 8.8.4.4
```

resolv.conf

Ahora, simplemente se vuelve a lanzar la instalación.

```
#  packstack --answer-file=answerfile.txt

 Welcome to the Packstack setup utility

 The installation log file is available at: /var/tmp/packstack/20170727-160250-oGrodP/openstack-
 setup.log

 Installing:
 Clean Up                                             [ DONE ]
 Discovering ip protocol version                      [ DONE ]
 Setting up ssh keys                                  [ DONE ]
 Preparing servers                                    [ DONE ]
 Pre installing Puppet and discovering hosts' details [ DONE ]
 Preparing pre-install entries                        [ DONE ]
 Setting up CACERT                                    [ DONE ]
 Preparing AMQP entries                               [ DONE ]
 Preparing MariaDB entries                            [ DONE ]
 Fixing Keystone LDAP config parameters to be undef if empty[ DONE ]
 Preparing Keystone entries                           [ DONE ]
 Preparing Glance entries                             [ DONE ]
 Checking if the Cinder server has a cinder-volumes vg[ DONE ]
 Preparing Cinder entries                             [ DONE ]
 Preparing Nova API entries                           [ DONE ]
 Creating ssh keys for Nova migration                 [ DONE ]
 Gathering ssh host keys for Nova migration           [ DONE ]
 Preparing Nova Compute entries                       [ DONE ]
 Preparing Nova Scheduler entries                     [ DONE ]
 Preparing Nova VNC Proxy entries                     [ DONE ]
 Preparing OpenStack Network-related Nova entries     [ DONE ]
 Preparing Nova Common entries                        [ DONE ]
 Preparing Neutron LBaaS Agent entries                [ DONE ]
 Preparing Neutron API entries                        [ DONE ]
 Preparing Neutron L3 entries                         [ DONE ]
 Preparing Neutron L2 Agent entries                   [ DONE ]
 Preparing Neutron DHCP Agent entries                 [ DONE ]
 Preparing Neutron Metering Agent entries             [ DONE ]
 Checking if NetworkManager is enabled and running    [ DONE ]
 Preparing OpenStack Client entries                   [ DONE ]
 Preparing Horizon entries                            [ DONE ]
 Preparing Puppet manifests                           [ DONE ]
 Copying Puppet modules and manifests                 [ DONE ]
 Applying 192.168.1.150_controller.pp
 192.168.1.150_controller.pp:                         [ DONE ]
 Applying 192.168.1.150_network.pp
 192.168.1.150_network.pp:                            [ DONE ]
 Applying 192.168.1.150_compute.pp
 192.168.1.150_compute.pp:                            [ DONE ]
 Applying Puppet manifests                            [ DONE ]
 Finalizing                                           [ DONE ]

  **** Installation completed successfully ******

 Additional information:
  * Time synchronization installation was skipped. Please note that unsynchronized time on server
 instances might be problem for some OpenStack components.
  * Warning: NetworkManager is active on 192.168.1.150. OpenStack networking currently does not
 work on systems that have the Network Manager service enabled.
  * File /root/keystonerc_admin has been created on OpenStack client host 192.168.1.150. To use
 the command line tools you need to source the file.
  * To access the OpenStack Dashboard browse to http://192.168.1.150/dashboard .
 Please, find your login credentials stored in the keystonerc_admin in your home directory.
  * The installation log file is available at: /var/tmp/packstack/20170727-160250-
 oGrodP/openstack-setup.log
  * The generated manifests are available at: /var/tmp/packstack/20170727-160250-oGrodP/manifests
```

Con esto la instalación de OpenStack ha finalizado correctamente y ya se puede utilizar.

4.2.3.Configuración de la red externa y del servicio de imágenes (Glance)

Con la configuración hecha hasta ahora, la interfaz de red está preparada pero no configurada. La configuración consiste en crear una red y una subred públicas, en las que se define las características, como IP, rango de IP, DNS, puerta de enlace, etc.

Para ejecutar cualquier comando OpenStack, es necesario identificarse, para ello se almacena en variables de entorno la información de autenticación. Esa información está en un fichero llamado *keystonerc_admin* y se puede ejecutar con un *source (.)*, pero no con un *bash (./)*, debido a que *bash* lo ejecutaría en otro hilo de procesado y las variables no estarían disponibles en el terminal.

```
4   unset OS_SERVICE_TOKEN
5       export OS_USERNAME=admin
6       export OS_PASSWORD=telematica
7       export OS_AUTH_URL=http://192.168.1.150:5000/v3
8       export PS1='[\u@\h \W(keystone_admin)]\$ '
9
10  export OS_PROJECT_NAME=admin
11  export OS_USER_DOMAIN_NAME=Default
12  export OS_PROJECT_DOMAIN_NAME=Default
13  export OS_IDENTITY_API_VERSION=3
```

keystonerc_admin

```
[root@packstack ~]# . keystonerc_admin

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

Para crear la red se puede usar varios comandos. Hasta ahora se ha estado usando los comandos de *neutron* para las instrucciones de red, pero en las nuevas versiones de OpenStack, como es el caso de Ocata, se está estandarizando todo bajo el comando *openstack*.

```
#  openstack network create --share --external --provider-physical-network extnet --provider-
   network-type flat external_network

+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2017-07-29T06:39:01Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | e762b201-3082-490e-bb3c-c6104e2f9c5d |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | False                                |
| mtu                       | 1500                                 |
| name                      | external_network                     |
| port_security_enabled     | False                                |
| project_id                | 81e29c7ded5f4f4d90d3b1111d9c08f8     |
| provider:network_type     | flat                                 |
| provider:physical_network | extnet                               |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 3                                    |
| router:external           | External                             |
| segments                  | None                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| updated_at                | 2017-07-29T06:39:01Z                 |
+---------------------------+--------------------------------------+
```

La opción `--share` permite que todos los proyectos usen la red virtual.

La opción `--external` define que la red virtual es externa. Si se quiere crear una red interna, se puede usar `--internal`, pero el valor por defecto ya es `internal`.

Las opciones `--provider-physical-network extnet` y `--provider-network-type flat` la red plana virtual con la red plana (nativa/sin etiquetar) física en la interfaz `eno1` del host.

Se crea la subred en la red virtual que se acaba de configurar.

```
# openstack subnet create --network external_network --allocation-pool
  start=192.168.1.152,end=192.168.1.199 --dns-nameserver 8.8.8.8 --dns-nameserver 8.8.4.4 --
  gateway 192.168.1.1 --subnet-range 192.168.1.0/24 external_network

+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 192.168.1.152-192.168.1.199          |
| cidr              | 192.168.1.0/24                       |
| created_at        | 2017-07-29T07:03:14Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.4.4, 8.8.8.8                     |
| enable_dhcp       | True                                 |
| gateway_ip        | 192.168.1.1                          |
| host_routes       |                                      |
| id                | 984f7596-eee6-4414-a651-101ef23879e6 |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | external_network                     |
| network_id        | e762b201-3082-490e-bb3c-c6104e2f9c5d |
| project_id        | 81e29c7ded5f4f4d90d3b1111d9c08f8     |
| revision_number   | 2                                    |
| segment_id        | None                                 |
| service_types     |                                      |
| subnetpool_id     | None                                 |
| updated_at        | 2017-07-29T07:03:14Z                 |
+-------------------+--------------------------------------+
```

La red externa “external_network” ya tiene acceso a internet y las VM en los proyectos serían accesibles desde internet. Pero para ello tal vez sea necesario agregar algunas rutas, dependiendo de si las VM en el proyecto tienen IP flotante o están en una red virtual privada.

Estos son los comandos, que pronto dejarán de estar disponibles, para configurar la red externa. Pueden ser de utilidad para versiones antiguas de OpenStack:

```
# neutron net-create external_network --provider:network_type flat --provider:physical_network
  extnet  --router:external

# neutron subnet-create --name public_subnet --enable_dhcp=False --allocation-
  pool=start=192.168.1.152,end=192.168.1.199 --gateway=192.168.1.1 external_network
  192.168.1.0/24
```

Como no se ha desplegado el proyecto “demo”, no existe ninguna imagen en Glance. Para probar a subir una imagen desde comandos se va a subir una imagen cirros, ya que es la imagen más básica para probar el funcionamiento de las VM en OpenStack.

```
# curl http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img | openstack image
  create "cirros" --disk-format qcow2 --container-format bare --public

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 12.6M  100 12.6M    0     0   470k      0  0:00:27  0:00:27 --:--:--  501k
+------------------+------------------------------------------------------+
```

```
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | f8ab98ff5e73ebab884d80c9dc9c7290                     |
| container_format | bare                                                 |
| created_at       | 2017-07-29T06:23:35Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/41dd774e-3790-4bfa-8c0e-d704d22e12e4/file |
| id               | 41dd774e-3790-4bfa-8c0e-d704d22e12e4                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | cirros                                               |
| owner            | 81e29c7ded5f4f4d90d3b1111d9c08f8                     |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 13267968                                             |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2017-07-29T06:24:02Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

Se puede comprobar que todas las configuraciones que se han hecho funcionan correctamente, desde la UI web. Cuando la instalación de Packstack termina, para acceder hay que dirigirse a la dirección del Controller en http://192.168.1.150/dashboard.

Las figuras 60 y 61 muestran sendos ejemplos de las ventanas asociadas a la UI de OpenStack.

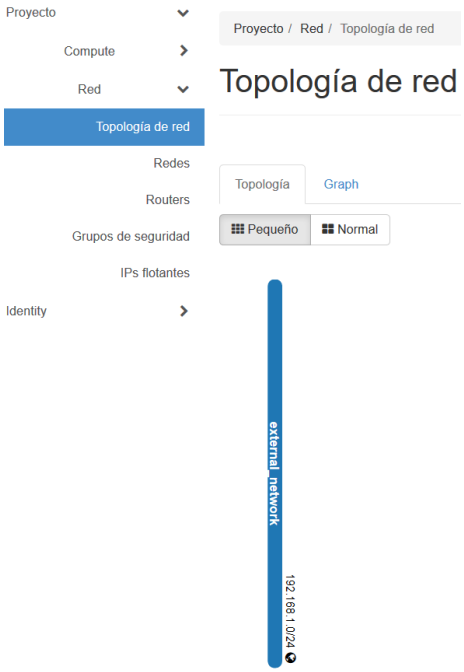


Figura 48 - Red externa en OpenStack

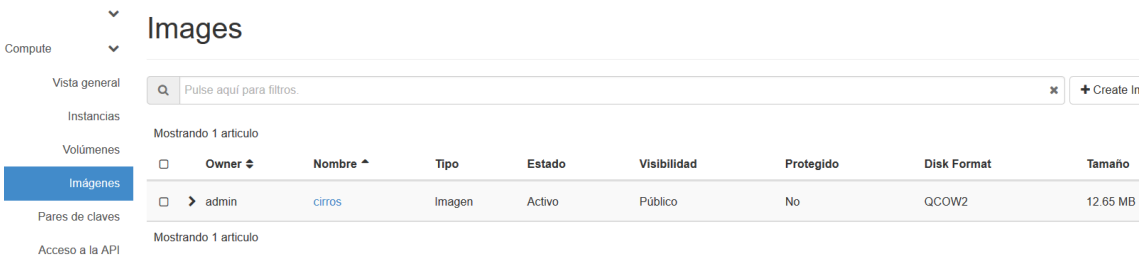


Figura 49 - Imágenes en OpenStack

4.3. OpenLDAP

Una vez se tiene desplegado el sistema que se quiere controlar, en este caso, una nube basada en OpenStack, se procede a la instalación del sistema de autentificación basado en LDAP que complementará al identificador por defecto o Keystone. Para ello se utilizarán los paquetes de OpenLDAP para montar un servidor LDAP, sobre una VM en VirtualBox con un CentOS 7 mínimo como sistema operativo.

Lo primero que se hace es instalar la máquina virtual, que como no necesita muchos recursos incluye solo un solo procesador, 1024 MB de memoria RAM y 8GB de almacenamiento. La red se configura como “Adaptador puente”.

La instalación es básicamente seguir los pasos que se indican. Durante la instalación renombramos el host como “ldap” y configuramos la interfaz de red con una IP estática, en este caso 192.168.1.151 y se activa la interfaz para comprobar que está bien configurada y no tener que activarla desde los archivos de configuración más adelante, tal como muestra la figura siguiente.

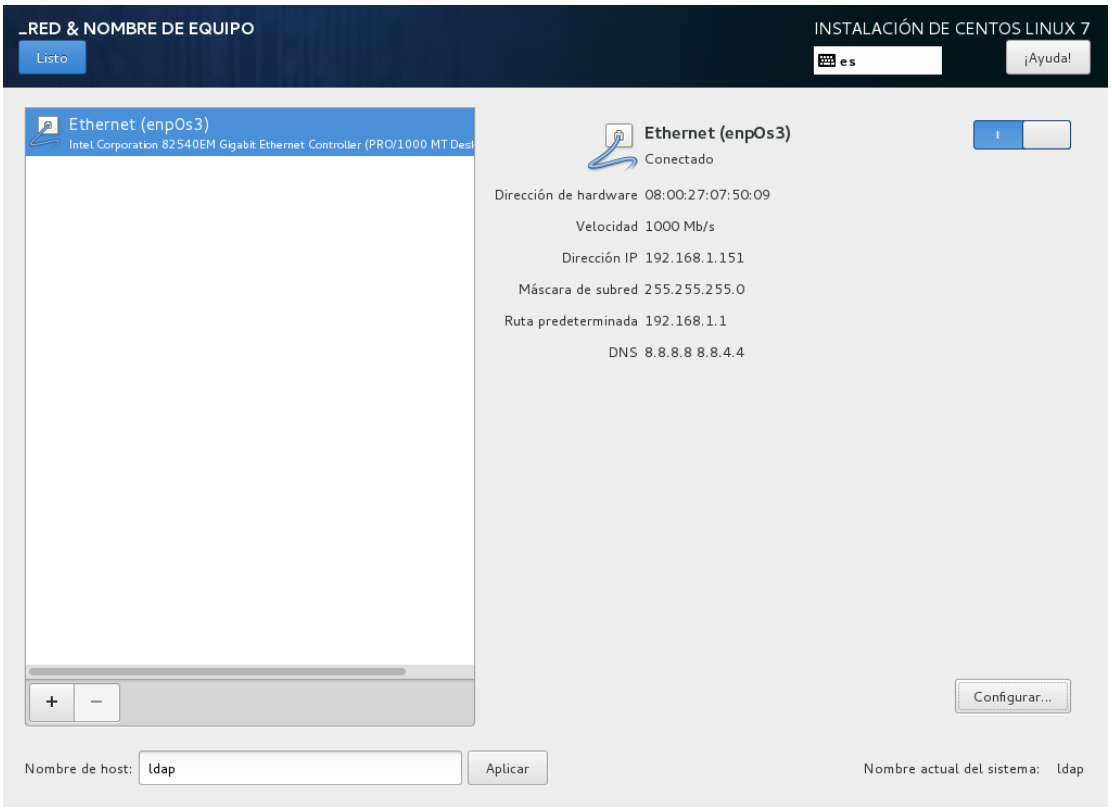


Figura 50 - Configuración de red del servidor LDAP

En la siguiente ventana se introduce la contraseña para el usuario root y se da la opción de crear otros usuarios. Solo se introducirá la contraseña nueva, no son necesarios usuarios.

Una vez terminada la instalación se debería poder acceder por SSH al servidor LDAP, esto no es necesario, pero facilita mucho el utilizar el terminal para copiar y pegar instrucciones. En este caso se ha hecho uso de Putty SSH.

Lo primero que se va a hacer es actualizar los paquetes e instalar nano como editor de ficheros.

```
#  yum -y update

#  yum -y install nano
```

Con esto sistema está al día y preparado para la instalación del servidor LDAP.

A continuación se instala el servidor y cliente LDAP en la máquina. [8]

```
# yum -y install openldap-servers openldap-clients
```

Se crea la nueva base de datos a partir de la que se administra como ejemplo y se cambia de propietario.

```
# cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
# chown ldap. /var/lib/ldap/DB_CONFIG
```

Se inicia el demonio del servidor ldap y se habilita para que se inicie con el sistema durante el arranque.

```
# systemctl start slapd
# systemctl enable slapd
```

Se genera un hash de una contraseña para ser utilizado como contraseña de administrador. El hash de esta contraseña se utilizara cuando se cree el usuario root de la instalación LDAP.

```
# slappasswd
New password :
Re-enter new password :
olcRootPW: {SSHA}l7FL0t4FoXtJP5FuKavgl9eldfg3Yup4
```

Se genera un fichero LDIF para modificar la información del usuario root en la base de datos.

```
# nano chrootpw.ldif
```

```
1  dn: olcDatabase={0}config,cn=config
2  changetype: modify
3  add: olcRootPW
4  olcRootPW: {SSHA}l7FL0t4FoXtJP5FuKavgl9eldfg3Yup4
```

chrootpw.ldif

Una vez se escribe en el atributo olcRootPW el hash de la contraseña que devuelve slappasswd, se procede a modificar la base de datos.

```
# ldapadd -Y EXTERNAL -H ldapi:/// -f chrootpw.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={0}config,cn=config"
```

Se importan los esquemas más básicos.

```
# ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=cosine,cn=schema,cn=config"
# ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
```

```
SASL SSF: 0
adding new entry "cn=nis,cn=schema,cn=config"
#  ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/inetorgperson.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=inetorgperson,cn=schema,cn=config"
```

Para configurar el dominio se hace algo parecido que para establecer la contraseña del usuario root. Se hace un fichero en el que se configura el dominio, se modificaran el acceso, la base del árbol LDAP, el nombre del usuario root del dominio y se añadirá una seguridad mínima.

Lo primero es volver a obtener un hash para la contraseña del usuario root.

```
#  slappasswd
New password :
Re-enter new password :
olcRootPW: {SSHA}lPFHYbw0rvM2V/1QtwQFD15t8FHG1qqv
```

```
#  nano chdomain.ldif
```

```
1   dn: olcDatabase={1}monitor,cn=config
2   changetype: modify
3   replace: olcAccess
4   olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth"
5     read by dn.base="cn=Manager,dc=mydomain,dc=com" read by * none
6
7   dn: olcDatabase={2}hdb,cn=config
8   changetype: modify
9   replace: olcSuffix
10  olcSuffix: dc=mydomain,dc=com
11
12  dn: olcDatabase={2}hdb,cn=config
13  changetype: modify
14  replace: olcRootDN
15  olcRootDN: cn=Manager,dc=mydomain,dc=com
16
17  dn: olcDatabase={2}hdb,cn=config
18  changetype: modify
19  add: olcRootPW
20  olcRootPW: {SSHA}lPFHYbw0rvM2V/1QtwQFD15t8FHG1qqv
21
22  dn: olcDatabase={2}hdb,cn=config
23  changetype: modify
24  add: olcAccess
25  olcAccess: {0}to attrs=userPassword,shadowLastChange by
26    dn="cn=Manager,dc=mydomain,dc=com" write by anonymous auth by self write by * none
27  olcAccess: {1}to dn.base="" by * read
28  olcAccess: {2}to * by dn="cn=Manager,dc=mydomain,dc=com" write by * read
```

chdomain.ldif

En el fichero anterior hay que prestar especial atención a cuál es el nombre del domino y al hash de la contraseña.

Se añaden de la misma forma que los anteriores, utilizando *ladpmodify* para editar la base de datos y cambiar la configuración del servidor LDAP.

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f chdomain.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={1}monitor,cn=config"

modifying entry "olcDatabase={2}hdb,cn=config"

modifying entry "olcDatabase={2}hdb,cn=config"

modifying entry "olcDatabase={2}hdb,cn=config"

modifying entry "olcDatabase={2}hdb,cn=config"
```

Ahora se crea la base del árbol para el dominio LDAP.

```
# nano basedomain.ldif
```

```
1   dn: dc=mydomain,dc=com
2   objectClass: top
3   objectClass: dcObject
4   objectclass: organization
5   o: My Domain
6   dc: mydomain
7
8   dn: cn=Manager,dc=mydomain,dc=com
9   objectClass: organizationalRole
10  cn: Manager
11  description: Directory Manager
```

basedomain.ldif

```
# ldapadd -x -D cn=Manager,dc=mydomain,dc=com -W -f basedomain.ldif
Enter LDAP Password:
adding new entry "dc=mydomain,dc=com"

adding new entry "cn=Manager,dc=mydomain,dc=com"
```

Por último, solo hace falta abrir los puertos del firewall para permitir el servicio LDAP. El protocolo LDAP normalmente hace uso del puerto 389/TCP y el 636/TCP para su versión segura sobre SSL/TLS.

```
# firewall-cmd --add-service=ldap --permanent
success
# firewall-cmd --reload
success
```

Con esto queda configurado el servidor LDAP en CentOS 7. Para comprobar la correcta configuración del servidor se puede hacer uso de JXplorer, que mostrará solamente la base del DN, ya que no se le ha introducido ninguna entrada y está completamente vació, con excepción del usuario root, renombrado como "*cn=Manager,dc=mydomain,dc=com*", tal como se muestra en la siguiente figura.

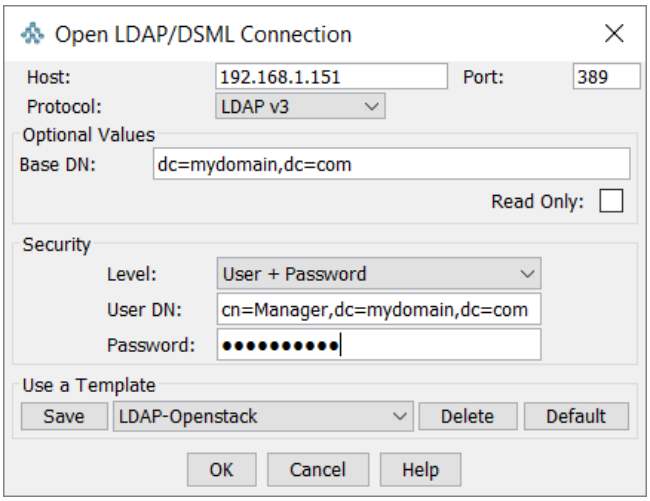


Figura 51 - Conexión con el servidor LDAP

Como se esperaba, lo único que aparece el árbol es el usuario administrador Manager, el resto del árbol está vacío y preparado para ser poblado con usuarios para OpenStack.

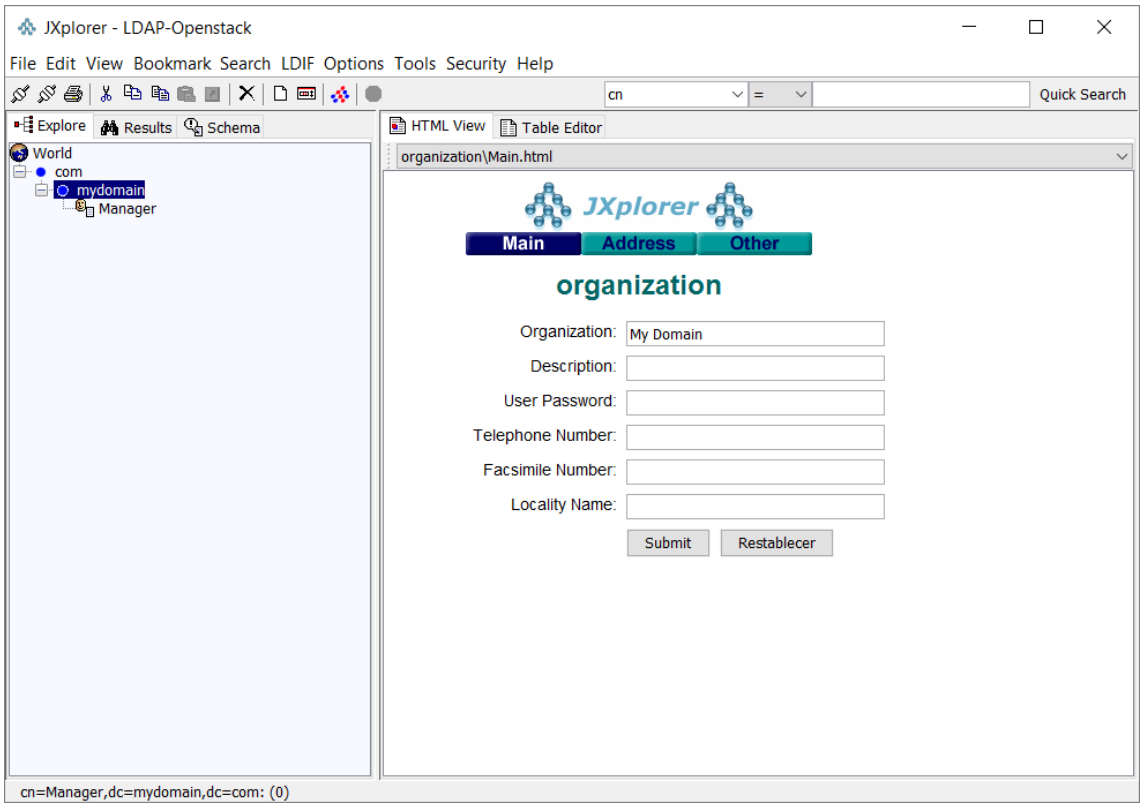


Figura 52 - Árbol visto en JXplorer

También es posible utilizar el paquete OpenLDAP-clients para realizar una petición LDAP, y comprobar así, las entradas en el árbol. La siguiente petición incluye algunas de las opciones más utilizadas para el comando ldapserach. Las opciones más importantes son las siguientes:

- `-b` Especifica donde empieza la búsqueda en el árbol LDAP. El valor especificado debe de ser un DN que exista en la base de datos. Por ejemplo -b "*dc=mydomain,dc=com*".

- `-D` Especifica el nombre de usuario con el que se desea autenticarse, esto es opcional si el servidor soporta autenticación anónima. Un ejemplo -D "*cn=Manager, dc=mydomain,dc=com*"
- `-h` Especifica el hostname o dirección IP de la máquina donde se encuentra instalado el servidor del directorio, si no se define nada en este campo, por defecto utiliza localhost. Un ejemplo -h 192.168.1.151 que es la IP donde está el servidor, en este caso.
- `-w` Se introduce la contraseña asociada con el nombre de usuario especificado con la opción -D. Si no se especifica se utiliza acceso anónimo. Un ejemplo -w telematica. Si es -W, no se escribe, te la pide durante la ejecución.
- `-x` No es muy importante, pero se suele utilizar, sirve para ordenar los resultados en el servidor en vez de en el cliente, lo que normalmente es más rápido.

```
#  ldapsearch -x -h 192.168.1.151 -b "dc=mydomain,dc=com" -D "cn=Manager,dc=mydomain,dc=com" -w
  telematica
# extended LDIF
#
# LDAPv3
# base <dc=mydomain,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# mydomain.com
dn: dc=mydomain,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: My Domain
dc: mydomain

# Manager, mydomain.com
dn: cn=Manager,dc=mydomain,dc=com
objectClass: organizationalRole
cn: Manager
description: Directory Manager

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

Una vez comprobado el correcto funcionamiento del servidor LDAP, se procede a crear los grupos administrativos que se utilizarán en OpenStack, véase usuarios y grupos. Estos grupos tendrán que corresponderse con las clases adecuadas para que tengan disponibles los atributos que precisan los usuarios o grupos de OpenStack. En la Figura 53 se muestra que atributos requieren los distintos parámetros.

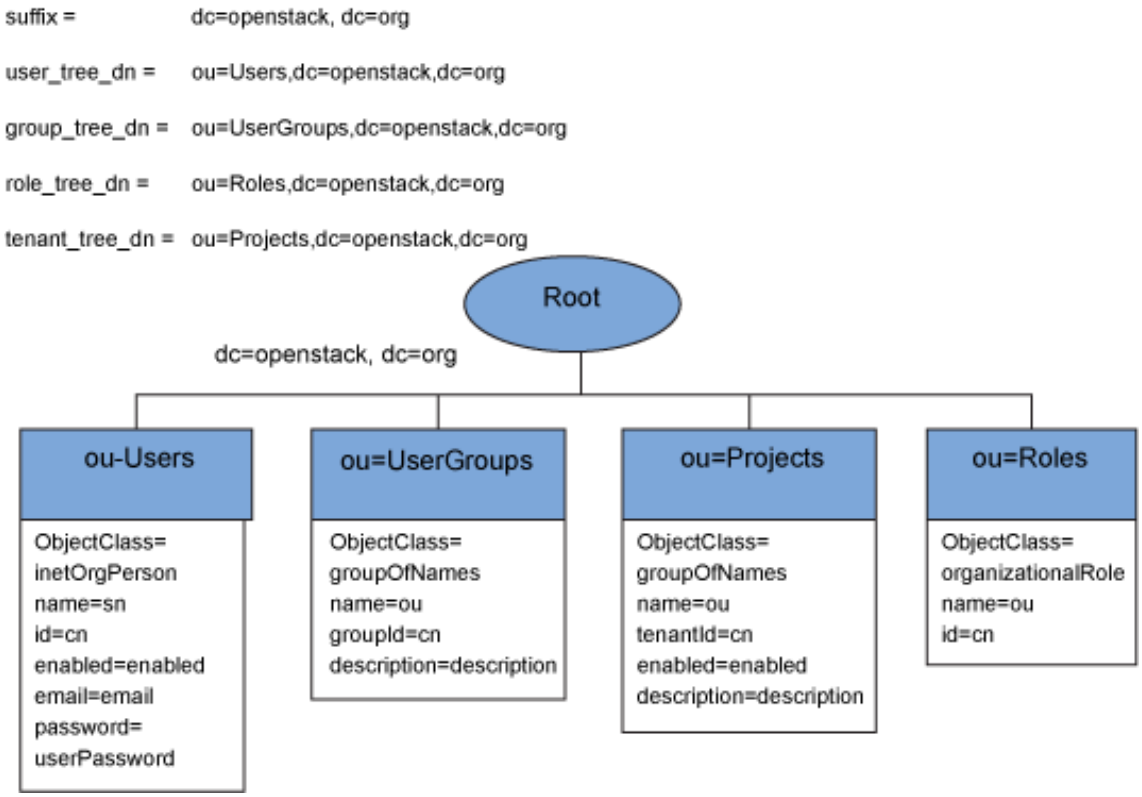


Figura 53 - Grupos administrativos de OpenStack en LDAP

En este trabajo solo se van a utilizar los dos primeros grupos debido a que no se van a definir otros roles y a que en las versiones más modernas de OpenStack no existen los “tenants”, que han pasado a llamarse “projects”. Además, estos son administrados por OpenStack.

En OpenStack se pueden asignar usuarios a un proyecto, pero es mucho más útil añadir un grupo a un proyecto, en vez de administrar el acceso individual de cada usuario desde OpenStack. Asignando un grupo a un proyecto y administrando los miembros de ese grupo desde el propio servidor LDAP, separa los usuarios de la administración de accesos a proyectos. Esto permite una mayor granularidad, haciendo más dinámico la asignación de usuarios a proyectos, pudiéndose administrar desde fuera de OpenStack.

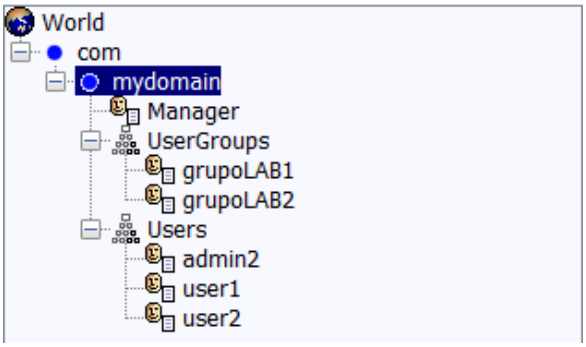


Figura 54 - Usuarios y grupos de OpenStack en LDAP

Se puede observar en los grupos, los atributos de los que dispone y en el atributo member (atributo muti-value obligatorio para la clase groupOfNames) se listan los usuarios. En el atributo se escribe el DN completo de los usuarios que forman parte del grupo y por tanto tienen acceso al proyecto al cual este grupo está asociado.

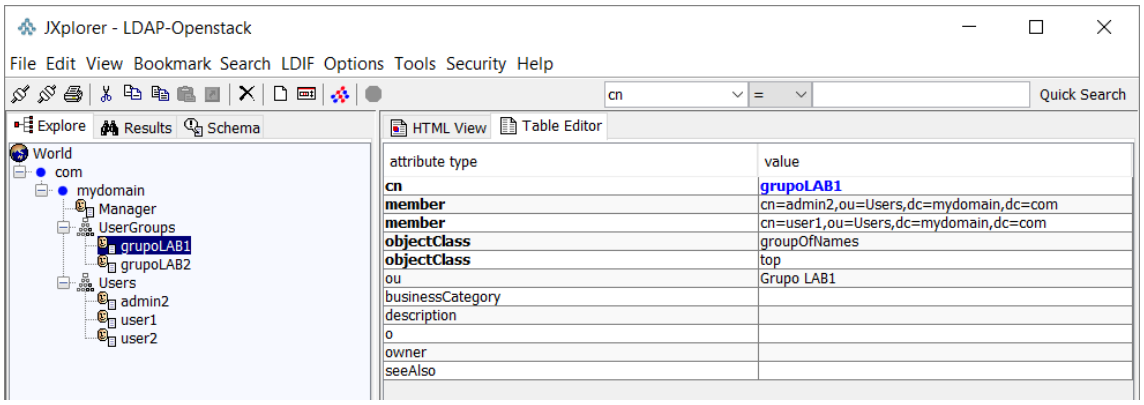


Figura 55 - Grupo del proyecto LAB1

En la unidad organizacional "Users" se almacenan todos los usuarios sin atender a que proyectos tienen acceso, ya que serán asociados a un proyecto a través de los grupos en la unidad organizacional "UserGroups".

Los usuarios utilizan la clase *inetOrgPerson*, y JXplorer añade por defecto, *organizationalPerson*, *Person* y *top*. Aquí hay que prestar atención al *cn* y el *sn*, el *cn* es el identificador (ID) por el cual OpenStack le identificará, en realidad derivará de él un código alfanumérico, y *sn* es el nombre que se utilizará para autentificarse en el OpenStack.

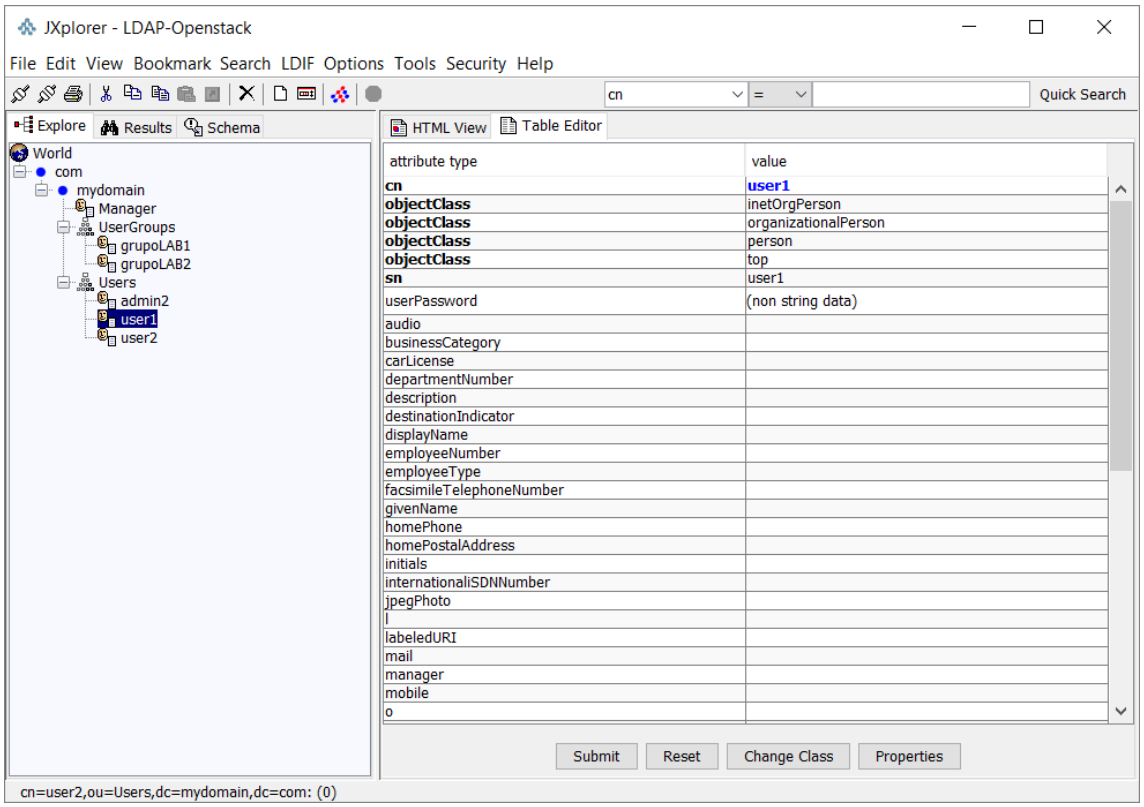


Figura 56 - Usuario OpenStack en LDAP

Para poder autentificarse desde OpenStack hay que proporcionar una contraseña que se ajuste con el hash que queda en el atributo userPassword de los usuarios.

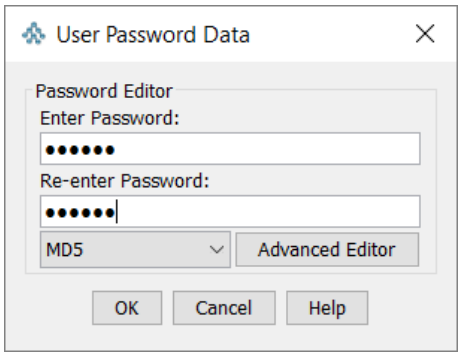


Figura 57 - Edito básico de contraseñas en JXplorer

Como se puede ver en la Figura 57, se utiliza el algoritmo MD5 para autentificar a los usuarios de OpenStack. Aunque la autentificación funciona con otros hashes en LDAP, no funciona en OpenStack.

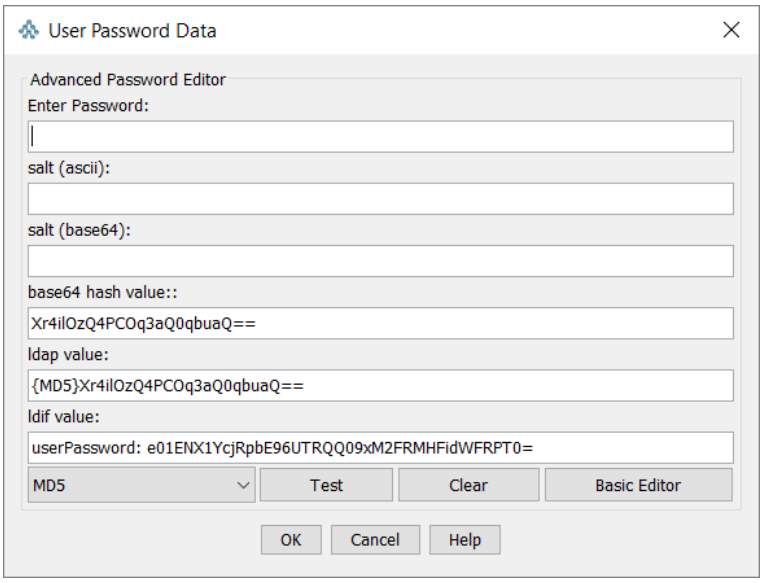


Figura 58 - Editor avanzado de contraseñas en JXplorer

Una de las fortalezas de LDAP, es la facilidad de importar y exportar entradas del directorio. LDAP utiliza ficheros LDIF en los cuales se escribe la información que se va a añadir o modificar en el árbol LDAP. Esto hace que simplemente manteniendo unos ficheros de texto se pueda mover los usuarios de un servidor a otro de forma muy rápida, y además ofrecen una buena solución como copias de respaldo.

Una de las herramientas que proporciona JXplorer está justamente pensada hacer más sencillo el importar y exportar archivos LDIF desde el propio cliente LDAP.

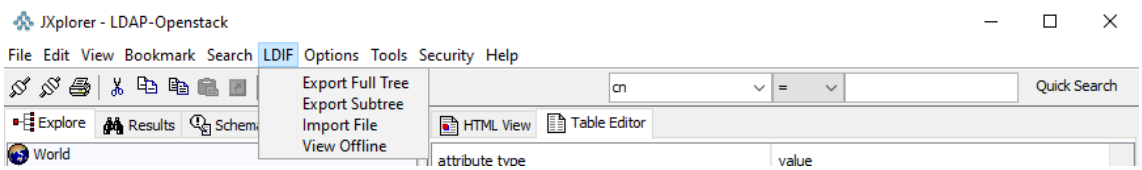


Figura 59 - Herramientas para ficheros LDIF en JXplorer

```
1   version: 1
2   dn: dc=mydomain,dc=com
3   objectClass: dcObject
4   objectClass: organization
5   objectClass: top
6   dc: mydomain
7   o: My Domain
8
9   dn: cn=Manager,dc=mydomain,dc=com
10  objectClass: organizationalRole
11  objectClass: top
12  cn: Manager
13  description: Directory Manager
14
15  dn: ou=Users,dc=mydomain,dc=com
16  objectClass: organizationalUnit
17  objectClass: top
18  ou: Users
19
20  dn: cn=admin2,ou=Users,dc=mydomain,dc=com
21  objectClass: inetOrgPerson
22  objectClass: organizationalPerson
23  objectClass: person
24  objectClass: top
25  cn: admin2
26  mail: alexsuesa@hotmail.com
27  sn: admin2
28  userPassword:: e01ENX1YcjRpbE96UTRQQ09xM2FRMHFidWFRPT0=
29
30  dn: cn=user1,ou=Users,dc=mydomain,dc=com
31  objectClass: inetOrgPerson
32  objectClass: organizationalPerson
33  objectClass: person
34  objectClass: top
35  cn: user1
36  sn: user1
37  userPassword:: e1NIQX01ZW42RzZNZXpScm9UM1hLcWtkUE9tWS9CZlE9
38
39  dn: cn=user2,ou=Users,dc=mydomain,dc=com
40  objectClass: inetOrgPerson
41  objectClass: organizationalPerson
42  objectClass: person
43  objectClass: top
44  cn: user2
45  sn: user2
46  userPassword:: e01ENX1YcjRpbE96UTRQQ09xM2FRMHFidWFRPT0=
47
48  dn: ou=UserGroups,dc=mydomain,dc=com
49  objectClass: organizationalUnit
50  objectClass: top
51  ou: UserGroups
52
53  dn: cn=grupoLAB1,ou=UserGroups,dc=mydomain,dc=com
54  objectClass: groupOfNames
55  objectClass: top
56  cn: grupoLAB1
57  member: cn=admin2,ou=Users,dc=mydomain,dc=com
58  member: cn=user1,ou=Users,dc=mydomain,dc=com
59  ou: Grupo LAB1
60
61  dn: cn=grupoLAB2,ou=UserGroups,dc=mydomain,dc=com
62  objectClass: groupOfNames
63  objectClass: top
64  cn: grupoLAB2
65  member: cn=admin2,ou=Users,dc=mydomain,dc=com
66  member: cn=user2,ou=Users,dc=mydomain,dc=com
67  ou: Grupo LAB2
```

JXplorer permite cambiar la base del dominio, por si se exporta a un servidor diferente.

El DN con el nombre de la base y el usuario administrador, o cualquier entrada que ya exista en el árbol del nuevo server, habrá que eliminarlo para evitar conflictos o utilizar ldapmodify.

Fichero LDIF exportado a través de JXplorer

Si se quiere modificar el control de acceso, que permisos tienen que usuarios para ver que atributos de que entrada, se puede usar un fichero LDIF para este tipo de configuración también.

Por ejemplo:

```
1   dn: olcDatabase={1}hdb,cn=config
2   changetype: modify
3   delete: olcAccess
4   olcAccess: {0}
5   -
6   # Then add a new ACL at position {0}.
7   add: olcAccess
8   olcAccess: {0}to attrs=userPassword,shadowLastChange by self write by anonymous auth by users
    none by * none
```

acl_update.ldif

En el ejemplo anterior el guion en la 5ª línea es importante para separar las ACL y que no salgan todas en una sola línea. [9]

Con la instrucción ldapmodify se modifica la configuración de las listas de control en el servidor LDAP.

```
# ldapmodify -v -n -f acl_update.ldif
```

4.3.1.TLS

Uno de los objetivos de este trabajo es hacer segura la comunicación entre el servicio de identidad de OpenStack (Keystone) y el servidor LDAP en el que confía para realizar una autentificación externa.

Para cumplir con esta seguridad se ha seleccionado TLS, que es un protocolo que hace uso de criptografía asimétrica para ofrecer un canal de comunicaciones seguro a nivel de transporte. TLS hace uso de claves privas y claves públicas, las cuales habrá que generar. Y compartir los certificados entre las dos máquinas. [10]

Lo primero que se necesita es una clave privada. (KEY)

```
# openssl genrsa -out localhost.key 8192
Generating RSA private key, 8192 bit long modulus
..............++
.....................................................................................
.............................++
e is 65537 (0x10001)
```

```
1   -----BEGIN RSA PRIVATE KEY-----
2   MIISKAIBAAKCBAEA0KhHolGU3/fRPcQtQNPYEUCOkvAmoiBv4kLzCfFlNDfujnOt
3   BhcbLTR0B/T+Sw/4ykPgmhYPoLlHOGx8iWBsNHk+oxYlltAjkkRsn2f26GJncuVJ
4   Ct2wiBooJ0CsojfzAI0GpR+PgfbjdXIvcFVYDHWsoOENQjX6CJkUP8i76jbURTxU
5   p2urSJ5r4aUYXsRB3wNqtAYDCQpIESJScX+E3CLUJ8fqqzbBmcfvy4Gsa3FhdN5i
6   X2tFKsevLmKr8IUCZcQ5suCrEEbxoXh59cd2sSPDBwoVhkoauSaYMrIaTo/gQrBW
7   wOJIxYwPlgUoftYDWE2K6eQR4cmPdaRwqGLo9ZfhrswOMECZgltKKl2IAGI45NOc
          . . .
          . . .
96  QAvzH070julNodWHv5FtI5DxCvDovTtev+3D/R/oLpHLXorKWlJ7uMy2Pdn6+EVt
97  eG/S0PRXPTuPq0vWQdB9jp1627N3147D9u10lKqNzU1QAzVkFpOE+qwxQPzQKNy1
98  goA/dRmnsQnWN/j+JgHmHX01mtPcqEgfgu6GWPsl7csRsoLcUS1WV4YMF7c=
99  -----END RSA PRIVATE KEY-----
```

localhost.key

Después se necesita una petición de certificado. (CSR)

```
#  openssl req -new -key localhost.key -out localhost.csr
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) [XX]:ES
State or Province Name (full name) []:Cantabria
Locality Name (eg, city) [Default City]:Santander
Organization Name (eg, company) [Default Company Ltd]:Universidad de Cantabria
Organizational Unit Name (eg, section) []:Master Telecomunicaciones
Common Name (eg, your name or your server's hostname) []:ldap    # server's FQDN
Email Address []:aac61@alumnos.unican.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

En la petición del certificado hay un apartado, “Common Name”, que es muy importante. En este campo hay que escribir el FQDN (Fully Qualified Domain Name) del servidor LDAP, ya que es donde va a intentar conectarse OpenStack cuando intente autentificar usuarios en el dominio LDAP.

```
1   -----BEGIN CERTIFICATE REQUEST-----
2   MIII+TCCBOECAQAwgbMxCzAJBgNVBAYTAkVTMRIwEAYDVQQIDAlDYW50YWJyaWEx
3   EjAQBgNVBAcMCVNhbnRhbmRlcjEhMB8GA1UECgwYVW5pdmVyc2lkYWQgZGUgQ2Fu
4   dGFicmlhMSIwIAYDVQQLDBlNYXN0ZXIgVGVsZWNvbXVuaWNhY2lvbmVzMQ0wCwYD
5   VQQDDARsZGFwMSYwJAYJKoZIhvcNAQkBFhdhYWM2MUBhbHVtbm9zLnVuaWNhbi5l
6   czCCBCIwDQYJKoZIhvcNAQEBBQADggQPADCCBAoCggQBANCoR6JRlN/30T3ELUDT
7   2BFAjpLwJqIgb+JC8wnxZTQ37o5zrQYXGy00dAf0/ksP+MpD4JoWD6C5RzhsfIlg
8   bDR5PqMWJZbQI5JEbJ9n9uhiZ3LlSQrdsIgaKCdArKI38wCNBqUfj4H243VyL3BV
9   WAx1rKDhDUI1+giZFD/Iu+o21EU8VKdrq0iea+GlGF7EQd8DarQGAwkKSBEiUnF/
10  hNwi1CfH6qs2wZnH78uBrGtxYXTeYl9rRSrHry5iq/CFAmXEObLgqxBG8aF4efXH

      . . .

      . . .

45  X25eLj4m9oh2NgfsP1Yuk3PgvcazwVNTeyvCjp9N9dn2xujCWdEzgWyRBuAkrHof
46  Qf/5p836eM0SNCZbCxaIQFakVAZnOGMdKmuf/nQn+2n2MJ3+XTIYxIzFehBh4Q1V
47  /oVpqXoPKStnJfekS8RH6X+FkqZxv7E1jbsaKewDk3RZ3U39N/4bDiMg9ISYa6Hb
48  XNgTDXgdTVI6Pm+qWF16qpCnBfJaL5c4srX8LeRNKYEudNNagKkg9GpYs3EfKCv2
49  RJgm/kybQteS80vcCtp4nJTPQ7HIXEf46UYxwVrChoFTw0nMxWUQwR+u6tLF
50  -----END CERTIFICATE REQUEST-----
```

localhost.csr

Por último, se utiliza la nueva petición de certificado y se crea un certificado autofirmado, firmándolo con la clave privada. En este ejemplo se le ha dado una validez de 3650 días, o lo que es lo mismo, unos 10 años. Para habilitar la comunicación segura entre las dos máquinas, este certificado hay que pasársele a OpenStack para que confíe en el servidor LDAP.

```
# openssl x509 -in localhost.csr -out localhost.crt -req -signkey localhost.key -days 3650

Signature ok

subject=/C=ES/ST=Cantabria/L=Santander/O=Universidad de Cantabria/OU=Master
Telecomunicaciones/CN=ldap/emailAddress=aac61@alumnos.unican.es

Getting Private key
```

Este es el certificado que se crea y que habrá que asociar al servidor LDAP, y que OpenStack confíe en él.

```
1   -----BEGIN CERTIFICATE-----
2   MIIJ5DCCBcwCCQDPIPxiomqNiDANBgkqhkiG9w0BAQUFADCBszELMAkGA1UEBhMC
3   RVMxEjAQBgNVBAgMCUNhbnRhYnJpYTESMBAGA1UEBwwJU2FudGFuZGVyMSEwHwYD
4   VQQKDBhVbml2ZXJzaWRhZCBkZSBDYW50YWJyaWExIjAgBgNVBAsMGU1hc3RlciBU
5   ZWxlY29tdW5pY2FjaW9uZXMxDTALBgNVBAMMBGxkYXAxJjAkBgkqhkiG9w0BCQEW
6   F2FhYzYxQGFsdW1ub3MudW5pY2FuLmVzMB4XDTE3MDcyNDE3MzEzNloXDTI3MDcy
7   MjE3MzEzNlowgbMxCzAJBgNVBAYTAkVTMRIwEAYDVQQIDAlDYW50YWJyaWExEjAQ
8   BgNVBAcMCVNhbnRhbmRlcjEhMB8GA1UECgwYVW5pdmVyc2lkYWQgZGUgQ2FudGFi
9   cmlhMSIwIAYDVQQLDBlNYXN0ZXIgVGVsZWNvbXVuaWNhY2lvbmVzMQ0wCwYDVQQD
10  DARsZGFwMSYwJAYJKoZIhvcNAQkBFhdhYWM2MUBhbHVtbm9zLnVuaWNhbi5lczCC
      . . .
      . . .
50  Jsf9kDNvnThj8xc2TOC5ISocPDMlVBj4+rQ2+d3pD7do8epcffzvFyAEbgd1JpFg
51  D6vWxjcfndYRAcQURPmOv1LYuKvwdcB5yyUECJy4YHAID3+nCTSX8k76W9MdOsQ2
52  O62XUosmiV2G4aajumpkQCYgd10SG6wfjDp+psYD982TjwuLYoMGX/LiSfKlmcof
53  T2eKRszqnHywcguejaUQox2p4DFFxq4uFHPQ1+yRKaf5sn43JTWK/1JWUNJiVZgA
54  sfl0X8enW/+kYgQo0GzTe6YW77TbaFt9bS816Q2dABSBlMjRqEJzIQ==
55  -----END CERTIFICATE-----
```

localhost.crt

Una vez se tiene el certificado, un conjunto de opciones han de ser configuradas tanto en el servidor como en el cliente para habilitar TLS y hacer uso de los certificados. Como mínimo los clientes deben de estar configurados con el nombre del fichero que contiene todas las CA (Certificate Authority) en los que confían. El servidor debe de estar configurado con el certificado de la CA, y también con su propio certificado de servidor y clave privada. [11]

Configuración del servidor

- `TLSCACertificateFile <filename>`

Esta directiva especifica el fichero en formato PEM que contiene los certificados para las CA en las que slapd va a confiar. El certificado para la CA que firmo el certificado del servidor debe de estar incluido en esos certificados. Si la CA firmante no es una CA de alto nivel (CA root), deben estar toda la secuencia de certificados hasta llegar a la CA de alto nivel. Múltiples certificados son simplemente adjuntados al fichero, el orden no importa.

- `TLSCACertificatePath <path>`

Esta directiva especifica la ruta a un directorio que contiene certificados CA individuales separados en diferentes ficheros. Además, este directorio debe de ser específicamente administrado con la utilidad OpenSSL *c_rehash*. OpenSSL intenta localizar los ficheros basandose en un has de los nombres de los archivos y de su numero de serie. En comparación TLSCACertificateFile es mucho más fácil de utilizar.

- `TLSCertificateFile <filename>`

Esta directiva especifica el fichero que contiene el certificado del servidor slapd. En general los certificados son información pública y no requieren de ninguna protección especial.

- `TLSCertificateKeyFile <filename>`

Esta directiva especifica el fichero que contiene la clave privada que concuerda con el certificado almacenado en el fichero TLSCertificateFile. Las claves públicas son información sensible que son normamente son almacenadas de forma encriptada para protejerlas. Pero la actual implentación so soporta claves encriptadas, así que la clave no debe de ser enciptada y el fichero debe ser almacenado cuidadosamente.

- `TLSVerifyClient { never | allow | try | demand }`

Esta directiva especifica que se comprueba en los certificados de los clientes en una conexión TLS entrante, si hay alguna. Esta opción está por defecto establecido en "never" en cuyo caso el servidor nunca pregunta al cliente por su certificado. Cuando está establecido en "allow" el servidor preguntará al cliente por un certificado, si no se manda ninguno la sesión procede con normalidad; si se manda un certificado, pero el servidor es incapaz de verificarlo, el certificado es ignorado y se procede con normalidad, como si ningún certificado hubiese sido provisto. Cuando se establece en "try" se pide el certificado y si ninguno es provisto, la sesión continua con normalidad, pero si el certificado suministrado no puede ser verificado, la sesión es terminada inmediatamente. Por último, si se establece en "demand" se pide un certificado y un certificado válido debe de ser entregado, de otra manera la sesión se termina de forma inmediata.

Para modificar la configuración del servidor LDAP, se hace como en las ocasiones anteriores, se genera un fichero LDIF y se utiliza ldapmodify. [12]

Lo primero se va a copiar los certificados y las claves a las carpetas adecuadas, y se va a establecer la nueva propiedad de los ficheros.

```
#  cp localhost.key localhost.crt /etc/pki/tls/certs/ca-bundle.crt /etc/openldap/certs/

#  chown ldap. /etc/openldap/certs/localhost.key /etc/openldap/certs/localhost.crt
   /etc/openldap/certs/ca-bundle.crt
```

Ahora se crea el fichero LDIF con el que se modificará el servidor LDAP y se activará TLS.

```
1   dn: cn=config
2   changetype: modify
3   add: olcTLSCACertificateFile
4   olcTLSCACertificateFile: /etc/openldap/certs/ca-bundle.crt
5   -
6   replace: olcTLSCertificateFile
7   olcTLSCertificateFile: /etc/openldap/certs/localhost.crt
8   -
9   replace: olcTLSCertificateKeyFile
10  olcTLSCertificateKeyFile: /etc/openldap/certs/localhost.key
```

mod_ssl.ldif

```
#  ldapmodify -Y EXTERNAL -H ldapi:/// -f mod_ssl.ldif

 SASL/EXTERNAL authentication started

 SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth

 SASL SSF: 0

 modifying entry "cn=config"
```

Se modifica las URIs por las que se puede acceder al servidor ldap, y se añade la opción segura de "ldaps:///".

```
#  nano /etc/sysconfig/slapd
```

En la línea 9 se añade el nuevo protocolo.

```
1   # OpenLDAP server configuration
2   # see 'man slapd' for additional information
3
4   # Where the server will run (-h option)
5   # - ldapi:/// is required for on-the-fly configuration using client tools
6   #   (use SASL with EXTERNAL mechanism for authentication)
7   # - default: ldapi:/// ldap:///
8   # - example: ldapi:/// ldap://127.0.0.1/ ldap://10.0.0.1:1389/ ldaps:///
9   SLAPD_URLS="ldapi:/// ldap:/// ldaps:///"
10
11  # Any custom options
12  #SLAPD_OPTIONS=""
13
14  # Keytab location for GSSAPI Kerberos authentication
15  #KRB5_KTNAME="FILE:/etc/openldap/ldap.keytab"
```

slapd

Por último, se reinicia el servidor LDAP. No olvidarse de abrir el puerto correspondiente en el firewall. Con estas configuraciones el lado del servidor debería estar terminado.

```
#  systemctl restart slapd

#  firewall-cmd --add-service=ldaps --permanent

 success

#  firewall-cmd --reload

 success
```

Configuración del cliente

La configuración de este cliente LDAP, solo tiene como propósito el comprobar el correcto funcionamiento de TLS en LDAP.

El verdadero Cliente en este trabajo será Keystone de OpenStack, pero este será configurado más adelante.

```
#  echo "TLS_REQCERT demand" >> /etc/openldap/ldap.conf

#  echo "tls_reqcert demand" >> /etc/nslcd.conf

#  authconfig --enableldaptls –update
```

Con esto, en principio, debería necesitar de un certificado TLS para poder continuar con la sesión LDAP y recuperar información. Para asegurarse de que hace uso del certificado.

Lo primero se comprueba que el certificado está a la vista y es verificado. Para ello se hace uso de la herramienta de OpenSSL *s_client*.

```
#  openssl s_client -connect 192.168.1.151:636 -CAfile /etc/openldap/certs/ca-bundle.crt -verify 5

 verify depth is 5

 CONNECTED(00000003)

 depth=0 C = ES, ST = Cantabria, L = Santander, O = Universidad de Cantabria, OU = Master
 Telecomunicaciones, CN = ldap, emailAddress = aac61@alumnos.unican.es

 verify error:num=18:self signed certificate

 verify return:1

 depth=0 C = ES, ST = Cantabria, L = Santander, O = Universidad de Cantabria, OU = Master
 Telecomunicaciones, CN = ldap, emailAddress = aac61@alumnos.unican.es

 verify return:1

 ---
```

```
Certificate chain

 0 s:/C=ES/ST=Cantabria/L=Santander/O=Universidad de Cantabria/OU=Master
Telecomunicaciones/CN=ldap/emailAddress=aac61@alumnos.unican.es

   i:/C=ES/ST=Cantabria/L=Santander/O=Universidad de Cantabria/OU=Master
Telecomunicaciones/CN=ldap/emailAddress=aac61@alumnos.unican.es

---

Server certificate

-----BEGIN CERTIFICATE-----
MIIJ5DCCBcwCCQDPIPxiomqNiDANBgkqhkiG9w0BAQUFADCBszELMAkGA1UEBhMC
RVMxEjAQBgNVBAgMCUNhbnRhYnJpYTESMBAGA1UEBwwJU2FudGFuZGVyMSEwHwYD
VQQKDBhVbml2ZXJzaWRhZCBkZSBDYW50YWJyaWExIjAgBgNVBAsMGU1hc3RlciBU
ZWxlY29tdW5pY2FjaW9uZXMxDTALBgNVBAMMBGxkYXAxJjAkBgkqhkiG9w0BCQEW
F2FhYzYxQGFsdW1ub3MudW5pY2FuLmVzMB4XDTE3MDcyNDE3MzEzNloXDTI3MDcy
MjE3MzEzNlowgbMxCzAJBgNVBAYTAkVTMRIwEAYDVQQIDAlDYW50YWJyaWExEjAQ
BgNVBAcMCVNhbnRhbmRlcjEhMB8GA1UECgwYVW5pdmVyc2lkYWQgZGUgQ2FudGFi
cmlhMSIwIAYDVQQLDBlNYXN0ZXIgVGVsZWNvbXVuaWNhY2lvbmVzMQ0wCwYDVQQD
DARsZGFwMSYwJAYJKoZIhvcNAQkBFhdhYWM2MUBhbHVtbm9zLnVuaWNhbi5lczCC
        . . .
        . . .
Jsf9kDNvnThj8xc2TOC5ISocPDMlVBj4+rQ2+d3pD7do8epcffzvFyAEbgd1JpFg
D6vWxjcfndYRAcQURPmOv1LYuKvwdcB5yyUECJy4YHAID3+nCTSX8k76W9MdOsQ2
O62XUosmiV2G4aajumpkQCYgd10SG6wfjDp+psYD982TjwuLYoMGX/LiSfKlmcof
T2eKRszqnHywcguejaUQox2p4DFFxq4uFHPQ1+yRKaf5sn43JTWK/1JWUNJiVZgA
sfl0X8enW/+kYgQo0GzTe6YW77TbaFt9bS816Q2dABSBlMjRqEJzIQ==
-----END CERTIFICATE-----

subject=/C=ES/ST=Cantabria/L=Santander/O=Universidad de Cantabria/OU=Master
Telecomunicaciones/CN=ldap/emailAddress=aac61@alumnos.unican.es

issuer=/C=ES/ST=Cantabria/L=Santander/O=Universidad de Cantabria/OU=Master
Telecomunicaciones/CN=ldap/emailAddress=aac61@alumnos.unican.es

---

No client certificate CA names sent

Server Temp Key: ECDH, secp521r1, 521 bits

---

SSL handshake has read 3862 bytes and written 441 bytes

---

New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 8192 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 0AE69F9E7F4DCF49B5DF447AE0A14364BD6762FF75D672884378059AA988BD9D
    Session-ID-ctx:
    Master-Key:
FB6A6742048649EEB5CBDA886F91A3E2395FA743792B3F3832AE63B1926E0447BF6187D53B9F3D317202036CA7A6216D
    Key-Arg   : None
    Krb5 Principal: None
    PSK identity: None
    PSK identity hint: None
    Start Time: 1500933398
    Timeout   : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
```

Como se puede ver, el certificado es el mismo que se creó y firmó anteriormente. La herramienta de OpenSSL *s_client* permite recuperar toda la información asociada a un certificado, y en este caso lo identifica como autofirmado. También da información de cual es protocolo de la sesión SSL/TLS y sobre la cadena de entidades certificadoras en caso de haberla.

Así se puede comprobar que efectivamente el certificado está disponible en el puerto 636 del servidor LDAP. Pero si se intenta lanzar una query ldap para hacer una búsqueda en el directorio se recibe un error.

```
#  ldapsearch -x -H ldaps://192.168.1.151 -b "dc=mydomain,dc=com" -D
   "cn=Manager,dc=mydomain,dc=com" -w telematica

 ldap_sasl_bind(SIMPLE): Can't contact LDAP server (-1)
```

Se puede comprobar que si se usa el protocolo no seguro “ldap:///” sí que funciona el servidor, por lo que hay un problema en la configuración para el protocolo “ldaps:///”.

```
#  ldapsearch -x -H ldap://192.168.1.151 -b "dc=mydomain,dc=com" -D
   "cn=Manager,dc=mydomain,dc=com" -w telematica

 # extended LDIF

 #

 # LDAPv3

 # base <dc=mydomain,dc=com> with scope subtree

 # filter: (objectclass=*)

 # requesting: ALL

      . . .

      . . .

 # search result

 search: 2

 result: 0 Success


 # numResponses: 10

 # numEntries: 9
```

Para encontrar que quiere decir el anterior error se puede utilizar el debugger de ldapsearch. Este debugger puede ser activado con la opción -d y el número indica en nivel del debugger. En este trabajo se ha habilitado simplemente la opción 1. [13] [14]

LEVEL	DESCRIPTION
-1	enable all debugging
0	no debugging
1	trace function calls
2	debug packet handling
4	heavy trace debugging
8	connection management
16	print out packets sent and received
32	search filter processing
64	configuration file processing
128	access control list processing
256	stats log connections/operations/results
512	stats log entries sent
1024	print communication with shell backends
2048	print entry parsing debugging

Tabla 1 - Niveles de debug en LDAP

Se pueden habilitar múltiples debuggers especificando en la opción de “debug” una para cada nivel que se desee(ej: -d1 -d64). O teninedo en cuenta que los niveles son aditivos, se puede hacer la cuenta y especidicar solo una opcion de debug (ej: -d65). Se puede entender como que cada bit representa uno de los niveles

1	0000 0000 0001
64	0000 0100 0000
65	0000 0100 0001

Tabla 2 – Suma “binaria”

```
#  ldapsearch -x -H ldaps://192.168.1.151 -b "dc=mydomain,dc=com" -D
  "cn=Manager,dc=mydomain,dc=com" -w telematica -d1

ldap_url_parse_ext(ldaps://192.168.1.151)

ldap_create

ldap_url_parse_ext(ldaps://192.168.1.151:636/??base)

ldap_sasl_bind

ldap_send_initial_request

ldap_new_connection 1 1 0

ldap_int_open_connection

ldap_connect_to_host: TCP 192.168.1.151:636

ldap_new_socket: 3

ldap_prepare_socket: 3

ldap_connect_to_host: Trying 192.168.1.151:636

ldap_pvt_connect: fd: 3 tm: -1 async: 0

attempting to connect:

connect success

TLS: certdb config: configDir='/etc/openldap' tokenDescription='ldap(0)' certPrefix='cacerts'
keyPrefix='cacerts' flags=readOnly

TLS: cannot open certdb '/etc/openldap', error -8018:Unknown PKCS #11 error.

TLS: could not get info about the CA certificate directory /etc/openldap/cacerts - error -
5950:File not found.

TLS: certificate [E=aac61@alumnos.unican.es,CN=ldap,OU=Master Telecomunicaciones,O=Universidad
de Cantabria,L=Santander,ST=Cantabria,C=ES] is not valid - error -8172:Peer's certificate issuer
has been marked as not trusted by the user..

TLS: error: connect - force handshake failure: errno 2 - moznss error -8172

TLS: can't connect: TLS error -8172:Peer's certificate issuer has been marked as not trusted by
the user..

ldap_err2string

ldap_sasl_bind(SIMPLE): Can't contact LDAP server (-1)
```

El primer error que hay que solucionar es que el certificado de la CA no le encuentra, esto es debido a que por defecto el cliente utiliza la directiva `TLS_CACERTDIR <path>` que es la equivalente a `TLSCACertificatePath <path>` en el servidor. En este caso no se utiliza esta directiva en el servidor, si no que se utiliza un archivo directamnete, en vez de una ruta.

```
#  nano /etc/openldap/ldap.conf
```

Se sustituye esa directiva por `TLS_CACERT <filename>` que es equivalente a `TLSCACertificateFile <filename>` en el lado del servidor. Como valor para esta directiva se le da el path completo hata el archivo, no el directorio, que contiene el certificado.

```
1   #
2   # LDAP Defaults
3   #
4
5   # See ldap.conf(5) for details
6   # This file should be world readable but not world writable.
7
8   #BASE   dc=example,dc=com
9   #URI    ldap://ldap.example.com ldap://ldap-master.example.com:666
10
11  #SIZELIMIT      12
12  #TIMELIMIT      15
13  #DEREF          never
14
15  #TLS_CACERTDIR /etc/openldap/cacerts
16
17  # Turning this off breaks GSSAPI used with krb5 when rdns = false
18  SASL_NOCANON    on
19  TLS_REQCERT demand
20  TLS_CACERT /etc/openldap/certs/localhost.crt
```

ldap.conf

Se pueden hacer dos cosas. La primera, simplemente añadir el certificado, autofirmado a `TLS_CACERT` y la segunda, anexar el certificado a la lista de certificados, una lista de las CA en las que se confían, en el fichero ca-bundle.crt. En este caso se ha escogido la primera opción.

Si se vuelve a ejecutar la búsqueda ldapsearch con el debugger, se puede observar otro error.

```
#  ldapsearch -x -H ldaps://192.168.1.151 -b "dc=mydomain,dc=com" -D
   "cn=Manager,dc=mydomain,dc=com" -w telematica -d1

ldap_url_parse_ext(ldaps://192.168.1.151)

ldap_create

ldap_url_parse_ext(ldaps://192.168.1.151:636/??base)

ldap_sasl_bind

ldap_send_initial_request

ldap_new_connection 1 1 0

ldap_int_open_connection

ldap_connect_to_host: TCP 192.168.1.151:636

ldap_new_socket: 3

ldap_prepare_socket: 3

ldap_connect_to_host: Trying 192.168.1.151:636

ldap_pvt_connect: fd: 3 tm: -1 async: 0

attempting to connect:

connect success

TLS: loaded CA certificate file /etc/openldap/certs/localhost.crt.

TLS: certificate [E=aac61@alumnos.unican.es,CN=ldap,OU=Master Telecomunicaciones,O=Universidad
de Cantabria,L=Santander,ST=Cantabria,C=ES] is valid

TLS certificate verification: subject: E=aac61@alumnos.unican.es,CN=ldap,OU=Master
Telecomunicaciones,O=Universidad de Cantabria,L=Santander,ST=Cantabria,C=ES, issuer:
E=aac61@alumnos.unican.es,CN=ldap,OU=Master Telecomunicaciones,O=Universidad de
Cantabria,L=Santander,ST=Cantabria,C=ES, cipher: AES-256-GCM, security level: high, secret key
bits: 256, total key bits: 256, cache hits: 0, cache misses: 0, cache not reusable: 0

TLS: hostname (192.168.1.151) does not match common name in certificate (ldap).

TLS: can't connect: TLS error -8157:Certificate extension not found..

ldap_err2string

ldap_sasl_bind(SIMPLE): Can't contact LDAP server (-1)
```

El certificado ahora es válido. El error esta vez tiene que ver con que el hostname (192.168.1.151) que se usa en la dirección de la instrucción no coincide con el FQDN que se encuentra en el certificado (Common Name=ldap).

Esto tiene facil solución si se usa el fichero /etc/hosts.

```
# nano /etc/hosts
```

En este fichero se añade una nueva entrada con la dirección IP del servidor LDAP y el “common name” utilizado durante la creación del certificado (ldap).

```
1   127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
2   ::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
3   192.168.1.151 ldap
```

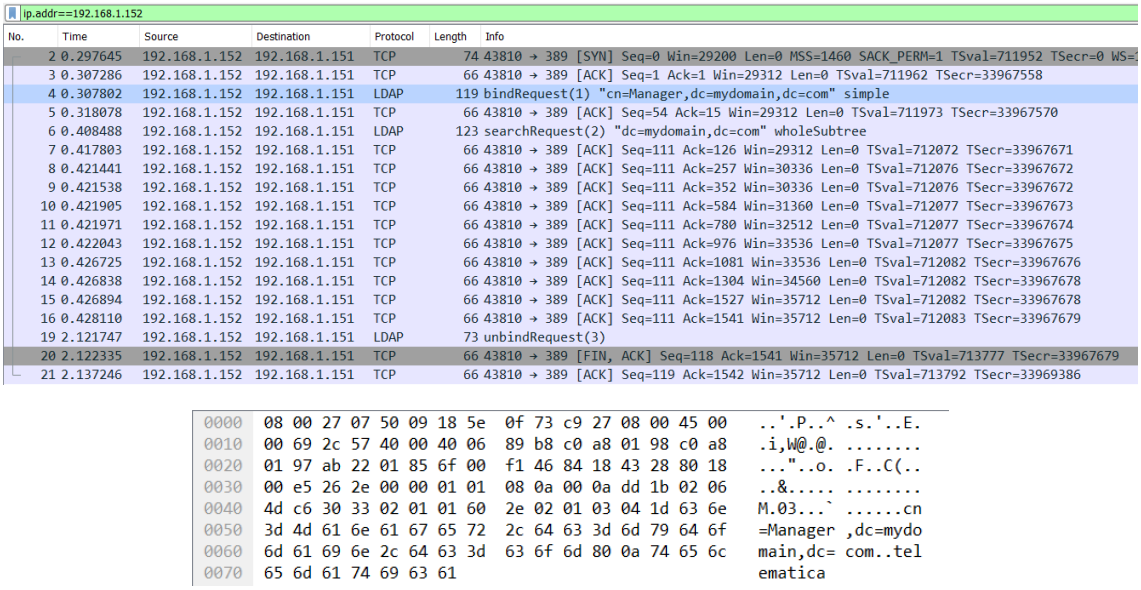
hosts

Por último, se puede comprobar, que se han solucionado todos los errores, haciendo una búsqueda con TLS y se puede comparar con una búsqueda sin TLS.

```
# ldapsearch -x -H ldaps://ldap -b "dc=mydomain,dc=com" -D "cn=Manager,dc=mydomain,dc=com" -w
  telematica
# extended LDIF
#
# LDAPv3
# base <dc=mydomain,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# mydomain.com
dn: dc=mydomain,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: My Domain
dc: mydomain

# Manager, mydomain.com
dn: cn=Manager,dc=mydomain,dc=com
objectClass: organizationalRole
cn: Manager
description: Directory Manager

   . . .
   . . .

# search result
search: 2
result: 0 Success

# numResponses: 10
# numEntries: 9
```

La diferencia entre usar TLS o no, es muy sencilla de ver con un analizador de protocolos.

Sin TLS



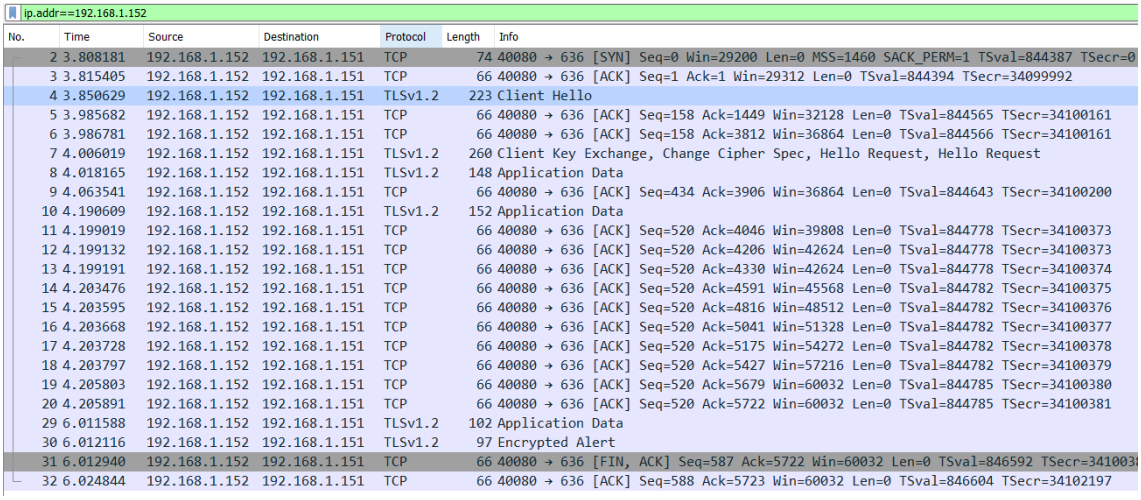
ip.addr==192.168.1.152

No.	Time	Source	Destination	Protocol	Length	Info
2	0.297645	192.168.1.152	192.168.1.151	TCP	74	43810 → 389 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=711952 TSecr=0 WS=
3	0.307286	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=711962 TSecr=33967558
4	0.307802	192.168.1.152	192.168.1.151	LDAP	119	bindRequest(1) "cn=Manager,dc=mydomain,dc=com" simple
5	0.318078	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=54 Ack=15 Win=29312 Len=0 TSval=711973 TSecr=33967570
6	0.408488	192.168.1.152	192.168.1.151	LDAP	123	searchRequest(2) "dc=mydomain,dc=com" wholeSubtree
7	0.417803	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=126 Win=29312 Len=0 TSval=712072 TSecr=33967671
8	0.421441	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=257 Win=30336 Len=0 TSval=712076 TSecr=33967672
9	0.421538	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=352 Win=30336 Len=0 TSval=712076 TSecr=33967672
10	0.421905	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=584 Win=31360 Len=0 TSval=712077 TSecr=33967673
11	0.421971	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=780 Win=32512 Len=0 TSval=712077 TSecr=33967674
12	0.422043	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=976 Win=33536 Len=0 TSval=712077 TSecr=33967675
13	0.426725	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=1081 Win=33536 Len=0 TSval=712082 TSecr=33967676
14	0.426838	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=1304 Win=34560 Len=0 TSval=712082 TSecr=33967678
15	0.426894	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=1527 Win=35712 Len=0 TSval=712082 TSecr=33967678
16	0.428110	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=111 Ack=1541 Win=35712 Len=0 TSval=712083 TSecr=33967679
19	2.121747	192.168.1.152	192.168.1.151	LDAP	73	unbindRequest(3)
20	2.122335	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [FIN, ACK] Seq=118 Ack=1541 Win=35712 Len=0 TSval=713777 TSecr=33967679
21	2.137246	192.168.1.152	192.168.1.151	TCP	66	43810 → 389 [ACK] Seq=119 Ack=1542 Win=35712 Len=0 TSval=713792 TSecr=33969386

```
0000  08 00 27 07 50 09 18 5e  0f 73 c9 27 08 00 45 00   ..'.P..^ .s.'..E.
0010  00 69 2c 57 40 00 40 06  89 b8 c0 a8 01 98 c0 a8   .i,W@.@. ........
0020  01 97 ab 22 01 85 6f 00  f1 46 84 18 43 28 80 18   ..."..o. .F..C(..
0030  00 e5 26 2e 00 00 01 01  08 0a 00 0a dd 1b 02 06   ..&..... ........
0040  4d c6 30 33 02 01 01 60  2e 02 01 03 04 1d 63 6e   M.03...` ......cn
0050  3d 4d 61 6e 61 67 65 72  2c 64 63 3d 6d 79 64 6f   =Manager ,dc=mydo
0060  6d 61 69 6e 2c 64 63 3d  63 6f 6d 80 0a 74 65 6c   main,dc= com..tel
0070  65 6d 61 74 69 63 61                              ematica
```

Figura 60 - Captura de paquetes sin TLS

Con TLS



ip.addr==192.168.1.152

No.	Time	Source	Destination	Protocol	Length	Info
2	3.808181	192.168.1.152	192.168.1.151	TCP	74	40080 → 636 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=844387 TSecr=0
3	3.815405	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=844394 TSecr=34099992
4	3.850629	192.168.1.152	192.168.1.151	TLSv1.2	223	Client Hello
5	3.985682	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=158 Ack=1449 Win=32128 Len=0 TSval=844565 TSecr=34100161
6	3.986781	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=158 Ack=3812 Win=36864 Len=0 TSval=844566 TSecr=34100161
7	4.006019	192.168.1.152	192.168.1.151	TLSv1.2	260	Client Key Exchange, Change Cipher Spec, Hello Request, Hello Request
8	4.018165	192.168.1.152	192.168.1.151	TLSv1.2	148	Application Data
9	4.063541	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=434 Ack=3906 Win=36864 Len=0 TSval=844643 TSecr=34100200
10	4.190609	192.168.1.152	192.168.1.151	TLSv1.2	152	Application Data
11	4.199019	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=4046 Win=39808 Len=0 TSval=844778 TSecr=34100373
12	4.199132	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=4206 Win=42624 Len=0 TSval=844778 TSecr=34100373
13	4.199191	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=4330 Win=42624 Len=0 TSval=844778 TSecr=34100374
14	4.203476	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=4591 Win=45568 Len=0 TSval=844782 TSecr=34100375
15	4.203595	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=4816 Win=48512 Len=0 TSval=844782 TSecr=34100376
16	4.203668	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=5041 Win=51328 Len=0 TSval=844782 TSecr=34100377
17	4.203728	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=5175 Win=54272 Len=0 TSval=844782 TSecr=34100378
18	4.203797	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=5427 Win=57216 Len=0 TSval=844782 TSecr=34100379
19	4.205803	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=5679 Win=60032 Len=0 TSval=844785 TSecr=34100380
20	4.205891	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=520 Ack=5722 Win=60032 Len=0 TSval=844785 TSecr=34100381
29	6.011588	192.168.1.152	192.168.1.151	TLSv1.2	102	Application Data
30	6.012116	192.168.1.152	192.168.1.151	TLSv1.2	97	Encrypted Alert
31	6.012940	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [FIN, ACK] Seq=587 Ack=5722 Win=60032 Len=0 TSval=846592 TSecr=341003
32	6.024844	192.168.1.152	192.168.1.151	TCP	66	40080 → 636 [ACK] Seq=588 Ack=5723 Win=60032 Len=0 TSval=846604 TSecr=34102197

```
0000  08 00 27 07 50 09 18 5e  0f 73 c9 27 08 00 45 00   ..'.P..^ .s.'..E.
0010  00 d1 e4 a8 40 00 40 06  d0 fe c0 a8 01 98 c0 a8   ....@.@. ........
0020  01 97 9c 90 02 7c 3d fc  06 c6 66 44 53 be 80 18   .....|=. ..fDS...
0030  00 e5 43 59 00 00 01 01  08 0a 00 0c e2 8e 02 08   ..CY.... ........
0040  53 18 16 03 01 00 98 01  00 00 94 03 03 85 3e 50   S....... ......>P
0050  19 5e eb bf af 50 a5 79  07 14 4e eb 21 61 03 e7   .^...P.y ..N.!a..
0060  5f fe d0 ac ce 89 2e 69  22 da 3c 78 26 00 00 36   _......i ".<x&..6
0070  c0 2c c0 0a c0 2b c0 09  c0 30 c0 14 c0 2f c0 13   .,...+.. .0.../..
0080  00 9f 00 39 00 38 00 6b  00 9e 00 33 00 32 00 67   ...9.8.k ...3.2.g
```

Figura 61 - Captura de paquetes con TLS

Como se puede ver, utilizar una conexión sin seguridad no es adecuado. Con un simple analizador de protocolos como Wireshark, se puede obtener la contraseña del usuario que se use para hacer el binding, con el árbol LDAP. Sin embarco con TLS se crean un túnel encriptado, por donde pasan las peticiones del LDAP, protegiendo las credenciales de los usuarios.

4.4. Configuración de OpenStack para que use LDAP como sistema autentificador

Para hacer uso de LDAP como sistema autenticador, se va a habilitar el multi-dominio en OpenStack. Este multi-dominio permite mantener separados en dominios diferentes proyectos, grupos, usuarios y roles. El dominio no se hará usando una base de datos, que es como funciona por defecto, si no que se usará un sistema de directorios LDAP, un protocolo diseñado para este tipo de funcionalidades. En este caso solo se va a mantener grupos y usuarios en el nuevo dominio. [15]

Suponiendo que ya se disponga de un servidor LDAP configurado adecuadamente, con usuarios y grupos preparados para usarlos en OpenStack, se procede con la configuración. [16]

Lo primero que se hace es ejecutar este procedimiento en el Controller, si se trata de un entorno de alta disponibilidad con múltiples Controller se ejecuta en cada uno de ellos.

```
#  setsebool -P authlogin_nsswitch_use_ldap=on
 Full path required for exclude: net:[4026532459].
```

Las respuestas de los comandos pueden incluir mensajes como el de arriba. Pueden ser ignorados.

Se crea el directorio para los dominios. En este directorio se encuentras los nuevos dominios que se creen.

```
#  mkdir /etc/keystone/domains/
#  chown keystone /etc/keystone/domains/
```

Se configura el servicio de identidad para que utiliza múltiples “Backends”. En esta parte se va a hacer uso de *crudini* para modificar el archivo de configuración de Keystone.

```
#  yum -y install crudini
#  crudini --set /etc/keystone/keystone.conf identity domain_specific_drivers_enabled true
#  crudini --set /etc/keystone/keystone.conf identity domain_config_dir /etc/keystone/domains
#  crudini --set /etc/keystone/keystone.conf assignment driver sql
```

Se habilitan los dominios múltiples en el dashboard. Para ello se añaden las siguientes líneas al archivo /etc/openstack-dashboard/local_settings.

```
#  nano /etc/openstack-dashboard/local_settings
```

```
66  OPENSTACK_API_VERSIONS = {
67    'identity': 3,
68
69  }
70
71
72  # Set this to True if running on multi-domain model. When this is enabled, it
73  # will require user to enter the Domain name in addition to username for login.
74  OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
75
76
77  # Overrides the default domain used when running on single-domain model
78  # with Keystone V3. All entities will be created in the default domain.
79  OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'Default'
```

local_settings

Hay que asegurarse de que usa la versión 3 ('identity': 3).

Se reinicia el servicio httpd para aplicar los cambios.

```
#  systemctl restart httpd.service
```

Figura 62 - Login de OpenStack con el multi-dominio activado

Una vez configurado el multi-dominio se procede con la configuración del "Backend" adicional.

Se crea un dominio Keystone para la integración de LDAP.

```
#  openstack domain create mydomain --description "Dominio LDAP"
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Dominio LDAP                     |
| enabled     | True                             |
| id          | f745d33ee24842a1bc816636c6a6a7ba |
| name        | mydomain                         |
+-------------+----------------------------------+
```

Se crea un fichero de configuración para el nuevo dominio. Para añadir el "Backend" para LDAP, se introducen los ajustes en un fichero llamado keystone.*mydomain*.conf (donde mydomain es el nombre de mi domino). Este fichero está localizado en el directorio que se ha creado para los dominios (/etc/keystone/domains/).

```
#  nano /etc/keystone/domains/keystone.mydomain.conf
```

```
1   [identity]
2   driver = ldap
3
4   [ldap]
5   url = ldaps://ldap
6   suffix = dc=mydomain,dc=com
7   query_scope = sub
8   user = cn=Manager,dc=mydomain,dc=com
9   password = telematica
10  use_dumb_member = False
```

```
11
12  user_tree_dn = ou=Users,dc=mydomain,dc=com
13
14  user_objectclass = inetOrgPerson
15
16  user_id_attribute = cn
17  #user_id_attribute = uidNumber
18  user_name_attribute = sn
19  user_mail_attribute = mail
20  user_pass_attribute = userPassword
21  user_enabled_attribute = enabled
22
23  group_tree_dn = ou=UserGroups,dc=mydomain,dc=com
24  group_objectclass = groupOfNames
25  group_id_attribute = cn
26  group_name_attribute = ou
27  group_member_attribute = member
28  group_desc_attribute = description
29
30  user_allow_create = fasle
31  user_allow_update = false
32  user_allow_delete = false
33  group_allow_create = false
34  group_allow_update = false
35  group_allow_delete = false
36
37  use_tls = false
38  tls_cacertfile = /etc/keystone/domains/ca-ldap.crt
```

keystone.mydomain.conf

En este fichero existen bastantes directivas que explicar. La primera directiva, en la sección [identity] básicamente especifica los drives que se van a utilizar en este "Backend".

En la sección [ldap], se especifica, la URI donde dirigirse, y como se puede observar, está haciendo uso del protocolo seguro de LDAP (ldaps), por lo que necesita como hostname el FQDN que se especifica en el atributo Common Name (CN) del certificado.

Después, se especifica la base del dominio LDAP, y que puede se puede ver (Sub, se pueden ver las entradas, por debajo de la base). También se declara el usuario que se conectará con el servidor LDAP, para administrar las cuentas de usuario y grupos, pero como se puede ver más abajo suele ser habitual que no se puedan crear o modificar las entradas desde OpenStack.

Por último, se especifica la ruta al archivo que contiene el certificado del servidor LDAP, para así saber en quien confiar, duran el intercambio de llaves.

Ahora, se puede importar el certificado del servidor LDAP. En este caso se ha decidido copiar el certificado a través de SSH en un fichero llamado ca-ldap.crt. En este fichero no pude darse que se use el protocolo ldaps:// y que la directiva use_tls tenga valor true, es uno o lo otro. Pero la directiva para la ruta del certificado (cacertfile) funciona para ambos casos.

```
# nano /etc/keystone/domains/ca-ldap.crt
```

```
1   -----BEGIN CERTIFICATE-----
2   MIIJ5DCCBcwCCQDPIPxiomqNiDANBgkqhkiG9w0BAQUFADCBszELMAkGA1UEBhMC
3   RVMxEjAQBgNVBAgMCUNhbnRhYnJpYTESMBAGA1UEBwwJU2FudGFuZGVyMSEwHwYD
4   VQQKDBhVbml2ZXJzaWRhZCBkZSBDYW50YWJyaWExIjAgBgNVBAsMGU1hc3RlciBU
5   ZWxlY29tdW5pY2FjaW9uZXMxDTALBgNVBAMMBGxkYXAxJjAkBgkqhkiG9w0BCQEW
6   F2FhYzYxQGFsdW1ub3MudW5pY2FuLmVzMB4XDTE3MDcyNDE3MzEzNloXDTI3MDcy
7   MjE3MzEzNlowgbMxCzAJBgNVBAYTAkVTMRIwEAYDVQQIDAlDYW50YWJyaWExEjAQ
8   BgNVBAcMCVNhbnRhbmRlcjEhMB8GA1UECgwYVW5pdmVyc2lkYWQgZGUgQ2FudGFi
9   cmlhMSIwIAYDVQQLDBlNYXN0ZXIgVGVsZWNvbXVuaWNhY2lvbmVzMQ0wCwYDVQQD
10  DARsZGFwMSYwJAYJKoZIhvcNAQkBFhdhYWM2MUBhbHVtbm9zLnVuaWNhbi5lczCC
```

```
          . . .
          . . .
50  Jsf9kDNvnThj8xc2TOC5ISocPDMlVBj4+rQ2+d3pD7do8epcffzvFyAEbgd1JpFg
51  D6vWxjcfndYRAcQURPmOv1LYuKvwdcB5yyUECJy4YHAID3+nCTSX8k76W9MdOsQ2
52  O62XUosmiV2G4aajumpkQCYgd10SG6wfjDp+psYD982TjwuLYoMGX/LiSfKlmcof
53  T2eKRszqnHywcguejaUQox2p4DFFxq4uFHPQ1+yRKaf5sn43JTWK/1JWUNJiVZgA
54  sfl0X8enW/+kYgQo0GzTe6YW77TbaFt9bS816Q2dABSBlMjRqEJzIQ==
55  -----END CERTIFICATE-----
```

ca-ldap.crt

Al igual que con el cliente LDAP, para poder acceder a la URI ldaps://ldap, es necesario declarar el hostname LDAP en el archivo de hosts del sistema.

```
#  nano /etc/hosts
```

```
1   127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
2   ::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
3   192.168.1.151 ldap
```

hosts

Se cambia a quién le pertenece el archivo de configuración del nuevo dominio, para dárselo a Keystone. Se reinicia el servidor httpd otra vez.

```
#  chown keystone /etc/keystone/domains/keystone.mydomain.conf

#  systemctl restart httpd.service
```

Se le concede al usuario administrador *admin* acceso al dominio. Esto no le concede ningún permiso a la cuenta *admin* en el servidor LDAP real. En este caso el termino se refiere al uso de OpenStack del dominio Keystone.

```
#  openstack role add --domain mydomain --user admin admin
```

Se puede comprobar que todo funciona correctamente intentando obtener una lista con los usuarios en el dominio LDAP. Estos usuarios deberían ser los que aparecen en la Figura 54.

```
#  openstack user list --domain mydomain

 +------------------------------------------------------------------+--------+
 | ID                                                               | Name   |
 +------------------------------------------------------------------+--------+
 | bad1e26805f0d2289980df388f74679da7085b9006155b96467d2c1d88efb12a | admin2 |
 | c94e02a7caf67f44f31d7d44d2ae704f12d0a3298c1fec61003b489a50b8f643 | user1  |
 | e3af157f28b597ecd46948fc6bfaed7d889caeb783fd137d6178e3dcd4e12cf7 | user2  |
 +------------------------------------------------------------------+--------+
```

Se configura Compute para que utilice Keystone v3 en vez de la versión por defecto (v2.0). Esto se hace para poder utilizar la funcionalidad del multi-dominio.

En cada nodo Compute hay que ajustar el valor el valor “keystone_authtoken”.

```
#  crudini --set /etc/nova/nova.conf keystone_authtoken auth_version v3
```

Se reinician los servicios en el Controller para aplicar los cambios.

```
#  systemctl restart openstack-nova-api.service openstack-nova-cert.service openstack-nova-
   conductor.service openstack-nova-consoleauth.service openstack-nova-novncproxy.service
   openstack-nova-scheduler.service
```

Y se reinicia este servicio en cada nodo Compute., pero aquí solo hay uno.

```
#  systemctl restart openstack-nova-compute.service
```

También se actualiza la versión de Keystone para el servicio de almacenamiento de bloques (Cinder).

```
#  nano /etc/cinder/cinder.conf
```

```
3782  [keystone_authtoken]
3783
3784  #
3785  # From keystonemiddleware.auth_token
3786  #
3787
3788  # Complete "public" Identity API endpoint. This endpoint should not be an
3789  # "admin" endpoint, as it should be accessible by all end users.
3790  # Unauthenticated clients are redirected to this endpoint to authenticate.
3791  # Although this endpoint should  ideally be unversioned, client support in the
3792  # wild varies.  If you're using a versioned v2 endpoint here, then this  should
3793  # *not* be the same endpoint the service user utilizes  for validating tokens,
3794  # because normal end users may not be  able to reach that endpoint. (string
3795  # value)
3796  #auth_uri = <None>
3797  auth_uri = http://192.168.1.150:5000/v3
3798
3799  # API version of the admin Identity API endpoint. (string value)
3800  auth_version = v3
```

cinder.conf

Y se reinician los servicios cinder.

```
#  systemctl restart openstack-cinder-api.service

#  systemctl restart openstack-cinder-backup.service

#  systemctl restart openstack-cinder-scheduler.service

#  systemctl restart openstack-cinder-volume.service
```

Se le concede acceso a un grupo en LDAP a un proyecto.

Tal y como se están haciendo las cosas en este trabajo, en el directorio LDAP debe de haber un grupo LDAP por proyecto. En concreto debería haber 2 grupos, *grupoLAB1* y *grupoLAB2*, que se corresponden con los proyectos *LAB1* y *LAB2* respectivamente. Como ejemplo se va a conceder a el grupo *grupoLAB1* acceso al proyecto *LAB1*, pero es exactamente igual para los demás casos.

Lo primero que se necesita son los proyectos, para crearlos se puede utilizar el siguiente comando.

```
#  openstack project create LAB1 --description "Laboratorio 1"
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Laboratorio 1                    |
| domain_id   | default                          |
| enabled     | True                             |
| id          | e2b58aba537547b78543c7d433988510 |
| is_domain   | False                            |
| name        | LAB1                             |
| parent_id   | default                          |
+-------------+----------------------------------+
```

Para las entidades que están en el dominio LDAP, no se puede usar su nombre en los comandos, hay que usar los identificadores ID. En el caso de que sea el dominio “default” sí que se puede usar los nombres como en el comando anterior donde se le conceden permisos al usuario *admin*, aunque es recomendable usar ID si hay conflictos.

Se obtiene la lista de grupos en el dominio LDAP y se coge el ID del grupo que interesa.

```
# openstack group list --domain mydomain

+----------------------------------------------------------------------+------------+
| ID                                                                   | Name       |
+----------------------------------------------------------------------+------------+
| b09e724e327f9c3aa04bebf3b38fd23b2fd2573689176e46a8f8024941cff888 | Grupo LAB1 |
| 35a0f440fb66e3ffb2bf50989dadb329a9b20ca6f2ed3e766638e504b45e2b63 | Grupo LAB2 |
+----------------------------------------------------------------------+------------+
```

Se obtiene la lista de roles que se pueden asignar a usuarios y grupos. Nos quedamos con _member_.

```
# openstack role list

+----------------------------------+----------+
| ID                               | Name     |
+----------------------------------+----------+
| 80c01f1ffdc04ac385d92027d6f37328 | admin    |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
+----------------------------------+----------+
```

Con el siguiente comando se le concede los permisos definidos en el rol escogido al grupo, identificado por el ID, en un proyecto.

```
# openstack role add --project LAB1 --group
  b09e724e327f9c3aa04bebf3b38fd23b2fd2573689176e46a8f8024941cff888 _member_
```

En este momento todo debería funcionar correctamente. Se intenta hacer un login desde el dashboard. Pero se produce un error debido a un Bug en el código. En el master ya ha sido arreglado, por lo que es posible que en una futura actualización se parchee el código.

Para arreglar este Bug, se comprueban los logs de Keystone.

```
# nano /var/log/keystone/keystone.log
```

```
4093  2017-07-06 16:02:02.517 8391 INFO keystone.token.persistence.backends.sql [-] Total expired
      tokens removed: 0
4094  2017-07-06  16:02:09.831  6844  INFO  keystone.common.wsgi  [req-c189ace2-98b7-46e3-b34a-
      567aa409f0bb - - - - -] POST http://192.168.1.150:5000/v3/auth/tokens
4095  2017-07-06  16:02:10.079  6844  ERROR  keystone.common.wsgi  [req-c189ace2-98b7-46e3-b34a-
      567aa409f0bb - - - - -] 'options'
4096  2017-07-06 16:02:10.079 6844 ERROR keystone.common.wsgi Traceback (most recent call last):
4097  2017-07-06 16:02:10.079 6844 ERROR keystone.common.wsgi    File "/usr/lib/python2.7/site-
      packages/keystone/common/wsgi.py", line 228, in __call__
4098  2017-07-06 16:02:10.079 6844 ERROR keystone.common.wsgi     result = method(req, **params)
4099  2017-07-06 16:02:10.079 6844 ERROR keystone.common.wsgi    File "/usr/lib/python2.7/site-
      packages/keystone/auth/controllers.py", line 132, in authenticate_for_token
4100  2017-07-06  16:02:10.079  6844  ERROR  keystone.common.wsgi        auth_context['user_id'],
      method_names_set):
4101  2017-07-06 16:02:10.079 6844 ERROR keystone.common.wsgi    File "/usr/lib/python2.7/site-
      packages/keystone/auth/core.py", line 377, in check_auth_methods_against_rules
4102  2017-07-06  16:02:10.079  6844  ERROR  keystone.common.wsgi              mfa_rules  =
      user_ref['options'].get(ro.MFA_RULES_OPT.option_name, [])
4103  2017-07-06 16:02:10.079 6844 ERROR keystone.common.wsgi KeyError: 'options'
4104  2017-07-06 16:02:10.079 6844 ERROR keystone.common.wsgi
```

keystone.log

La Autenticación de usuarios LDAP falla al comprobar la regla MFA. Este error surge porque en la línea remarcada en rojo *user_ref* espera siempre tener un atributo opciones y esto no está presente para los usuarios LDAP, por lo tanto, falla. [17]

Este es un bug identificado con como BUG 1662762. Y ya ha sido arreglado, por lo que en la página web de OpenStack ofrecen los cambios que hay que hacer en el código para que funcione la autenticación a través de LDAP.

Para arreglarlo hay que modificar unos archivos del código escritos en Python, este fichero define cómo comportarse en el caso de un "Backend" LDAP. [18]

```
#  nano /usr/lib/python2.7/site-packages/keystone/identity/backends/ldap/core.py
```

```
            . . .
306     def mask_enabled_attribute(self, values):
307         value = values['enabled']
308         values.setdefault('enabled_nomask', int(self.enabled_default))
309         if value != ((values['enabled_nomask'] & self.enabled_mask) !=
310                      self.enabled_mask):
311             values['enabled_nomask'] ^= self.enabled_mask
312         values['enabled'] = values['enabled_nomask']
313         del values['enabled_nomask']
314
315     def create(self, values):
316         if 'options' in values:
317             values.pop('options')  # can't specify options
318         if self.enabled_mask:
319             orig_enabled = values['enabled']
320             self.mask_enabled_attribute(values)
321         elif self.enabled_invert and not self.enabled_emulation:
322             orig_enabled = values['enabled']
323             if orig_enabled is not None:
324                 values['enabled'] = not orig_enabled
325             else:
326                 values['enabled'] = self.enabled_default
327         values = super(UserApi, self).create(values)
328         if self.enabled_mask or (self.enabled_invert and
329                                  not self.enabled_emulation):
330             values['enabled'] = orig_enabled
331         values['options'] = {}  # options always empty
332         return values
333
334     def get(self, user_id, ldap_filter=None):
335         obj = super(UserApi, self).get(user_id, ldap_filter=ldap_filter)
336         obj['options'] = {}  # options always empty
337         return obj
338
339     def get_filtered(self, user_id):
340         user = self.get(user_id)
341         return self.filter_attributes(user)
342
343     def get_all(self, ldap_filter=None, hints=None):
344         objs = super(UserApi, self).get_all(ldap_filter=ldap_filter,
345                                             hints=hints)
346         for obj in objs:
347             obj['options'] = {}  # options always empty
348         return objs
349
350     def get_all_filtered(self, hints):
351         query = self.filter_query(hints, self.ldap_filter)
352         return [self.filter_attributes(user)
353                 for user in self.get_all(query, hints)]
354
355     def filter_attributes(self, user):
356         return base.filter_user(common_ldap.filter_entity(user))
357
358     def is_user(self, dn):
359         """Return True if the entry is a user."""
360         # NOTE(blk-u): It's easy to check if the DN is under the User tree,
361         # but may not be accurate. A more accurate test would be to fetch the
362         # entry to see if it's got the user objectclass, but this could be
```

```
363         # really expensive considering how this is used.
364
365         return common_ldap.dn_startswith(dn, self.tree_dn)
366
367     def update(self, user_id, values, old_obj=None):
368         if old_obj is None:
369             old_obj = self.get(user_id)
370         # don't support updating options
371         if 'options' in old_obj:
372             old_obj.pop('options')
373         if 'options' in values:
374             values.pop('options')
375         values = super(UserApi, self).update(user_id, values, old_obj)
376         values['options'] = {}  # options always empty
377         return values
378
379
380 class GroupApi(common_ldap.BaseLdap):
381     DEFAULT_OU = 'ou=UserGroups'
382     DEFAULT_STRUCTURAL_CLASSES = []
383     DEFAULT_OBJECTCLASS = 'groupOfNames'
384     DEFAULT_ID_ATTR = 'cn'
385     DEFAULT_MEMBER_ATTRIBUTE = 'member'
386     NotFound = exception.GroupNotFound
387     options_name = 'group'
388     attribute_options_names = {'description': 'desc',
            . . .
```

core.py

Se reinicia el servicio httpd una última vez.

```
#  systemctl restart httpd.service
```

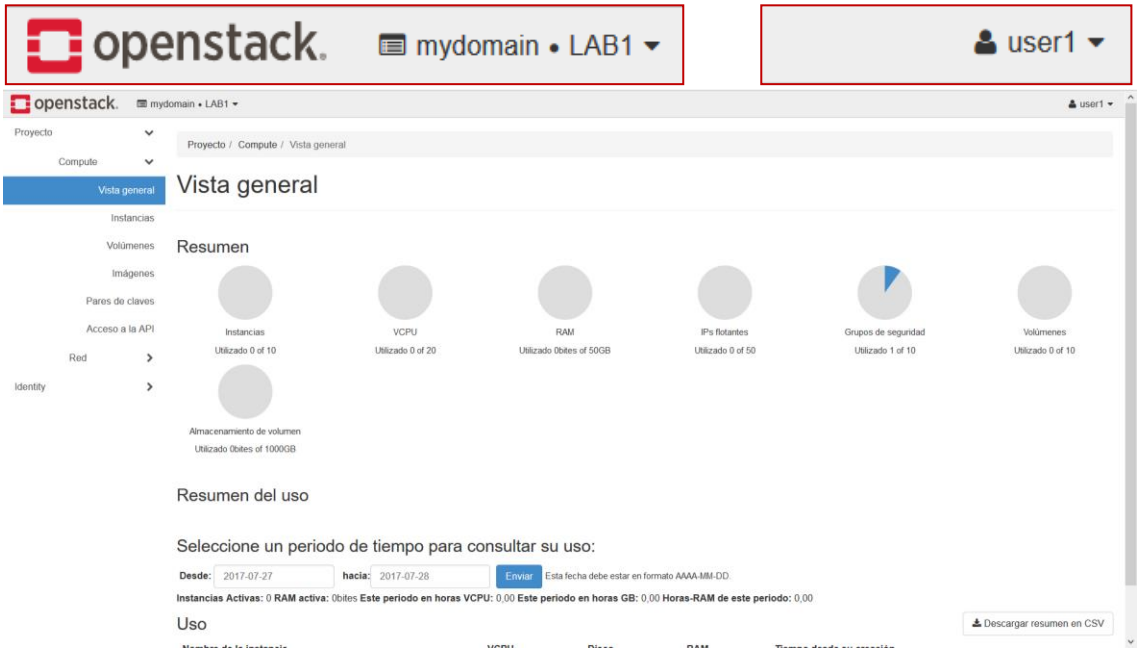


Figura 63 - Acceso al dominio LDAP de OpenStack con un usuario en LDAP

4.5. Laboratorio virtual

Uno de los objetivos de este trabajo era, conseguir levantar un laboratorio virtual con solo la ejecución de un script o algo igual de sencillo.

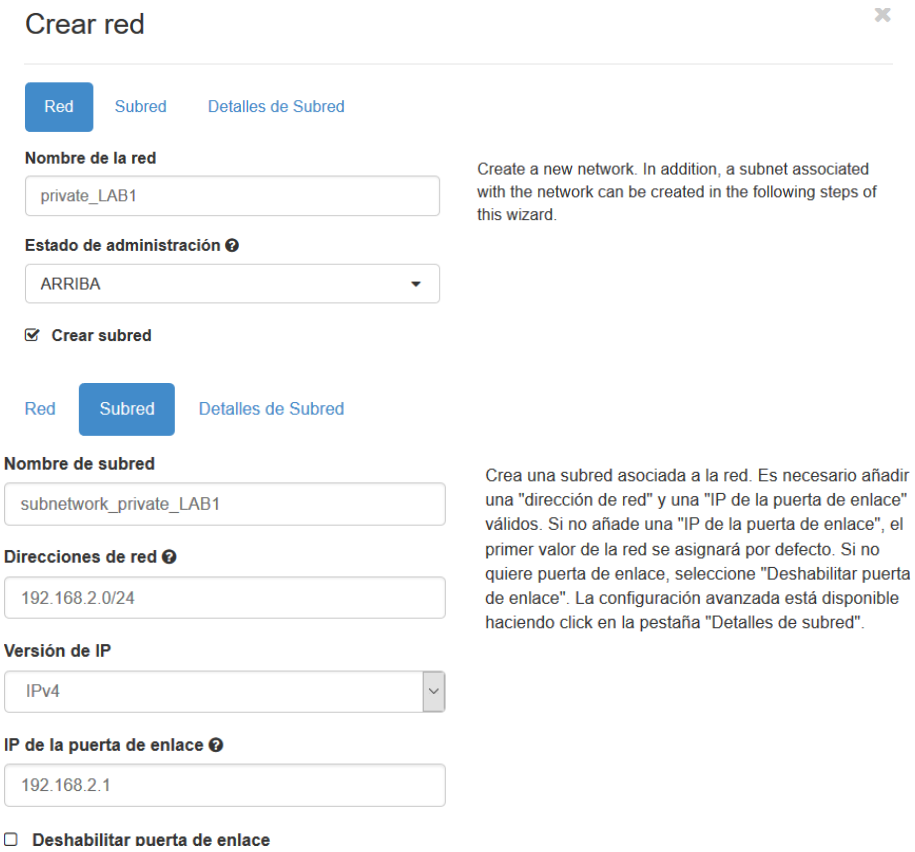
El laboratorio virtual de prueba será simplemente un par de máquinas cirros, en uno de los proyectos que se han creado antes.

Lo primero que se hace es crear una red privada para el proyecto, y lo conectamos a la red externa, ya que queremos poder encender la VM del proyecto y acceder a ellas por SSH.

Se crea un router y se conecta la red externa (external_network).

Figura 64 - Crear router entre la red externa y la red privada

Se crea una red privada para el proyecto, en este caso el proyecto es LAB1. También se puede definir las subred y detalles, como si hay DHCP, el pool de direcciones o DNS.



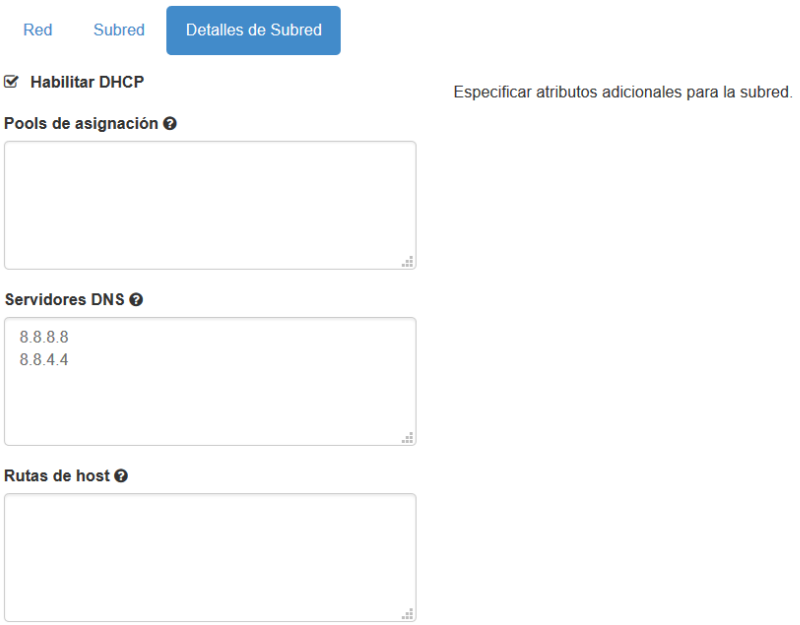


Figura 65 - Creación de una red privada para el proyecto LAB1

Para conectar la nueva red privada a la red externa, se hace a través del router que se ha creado antes. Para ello se le añade una nueva interfaz.

Figura 66 - Agregar interfaz al router

Figura 67 - Configuración de la nueva interfaz

Al añadir la nueva interfaz se selección la subred que se le quiere conectar, en este caso la subred que se ha creado para la red privada del proyecto LAB1. Y se le conecta como dirección IP la que se ha definido como Gateway, para así tener conectividad con otras redes.

Si todo va bien, debería quedar configurado algo parecido a esto.

Figura 68 - Vista general del router R1 del proyecto LAB1

Se crean un par de instancias para simular las VM que se tienen que levantar desde el script.

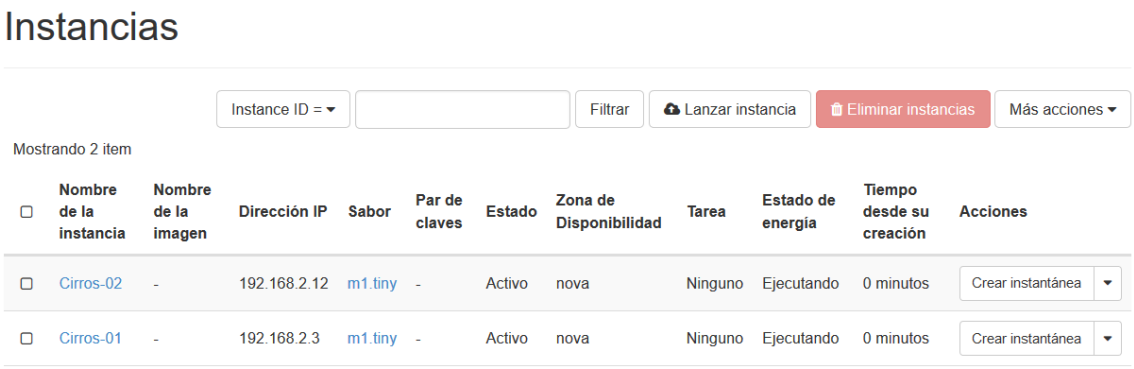


Figura 69 - Máquinas Virtuales del laboratorio virtual

En este momento en las VM solo se puede hace un ping para probar la conexión desde la red privada hacia las redes externas. No se puede hacer desde la red externa a la red privada del proyecto.

Para solucionar este problema hay que abrir los puertos ICMP y SSH, que son los protocolos que se quieren utilizar, al router R1_LAB1.

En el apartado de Red > Grupos de seguridad es donde se configurar los puertos del router. Por defecto hay un grupo de reglas llamado “default”. Interesa añadir las reglas ICMP y SSH con dirección entrante y así poder manejar las VM desde la red externa. Se pueden modificar las reglas, y añadir “ALL ICMP” y “SSH”, quedando algo tal que así.

Administrar Reglas de Grupo de Seguridad: default (b1b1bc6f-a7f1-4a87-b20b-e4dc369e8bbb)

+ Agregar regla | Eliminar Reglas

Mostrando 6 item

Dirección	Tipo Ethernet	Protocolo IP	Rango de puertos	Prefijo de IP Remota	Grupo de Seguridad Remoto	Acciones
Entrante	IPv6	Cualquier	Cualquier	-	default	Eliminar Regla
Saliente	IPv4	Cualquier	Cualquier	0.0.0.0/0	-	Eliminar Regla
Entrante	IPv4	Cualquier	Cualquier	-	default	Eliminar Regla
Saliente	IPv6	Cualquier	Cualquier	::/0	-	Eliminar Regla
Entrante	IPv4	ICMP	Cualquier	0.0.0.0/0	-	Eliminar Regla
Entrante	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	Eliminar Regla

Figura 70 - Grupo de seguridad default

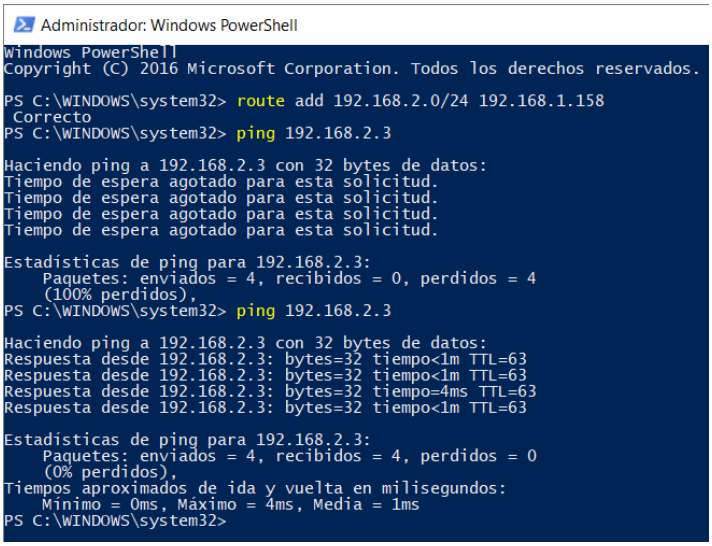


Figura 71 - PING a la VM en el proyecto LAB1

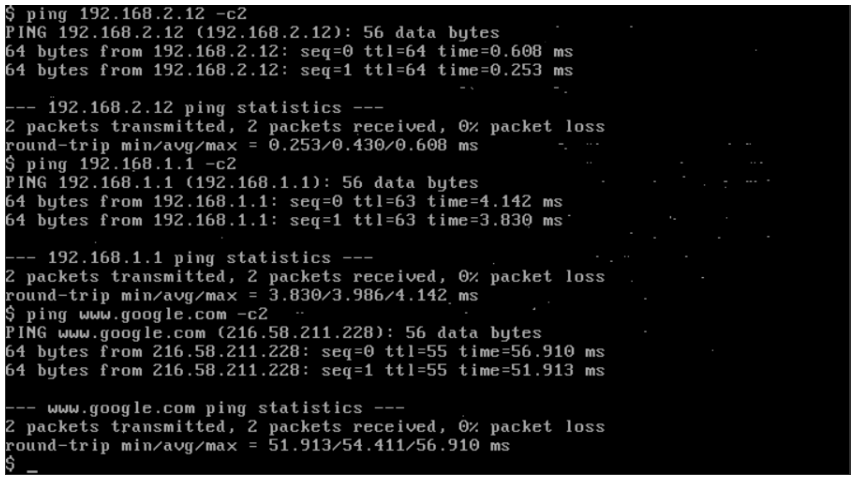


Figura 72 - Pruebas de PING desde VM en el proyecto LAB1

Como se puede ver la Figura 71, para poder dirigirse a la red privada del proyecto LAB1, hay que añadir una ruta donde el Gateway es la dirección de la interfaz externa en el router R1_LAB1, esta configuración e puede ver en la Figura 68.

Por supuesto, de la misma forma que PING, habiendo añadido la regla de SSH, es posible controlar la VM desde la red externa.

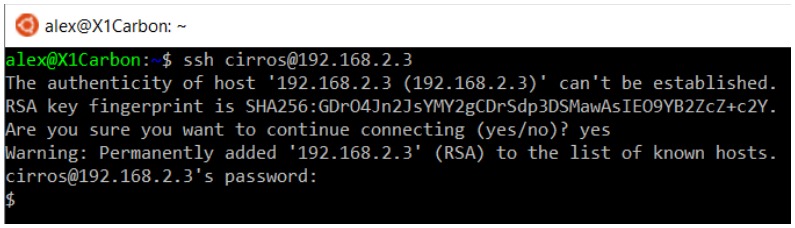


Figura 73 - SSH a la VM en el proyecto LAB1

Para lanzar las máquinas hay que tener acceso a OpenStack CLI, que está en el Controller (192.168.1.150). Se puede usar SSH a al Controller e ir pasándole los comandos a ejecutar. El problema son las credenciales, para no usar en todos los proyectos la cuenta de administrador hay que usar unas credenciales válidas para el proyecto, en formato "RC File". Las credenciales se pueden obtener desde el dashboard.

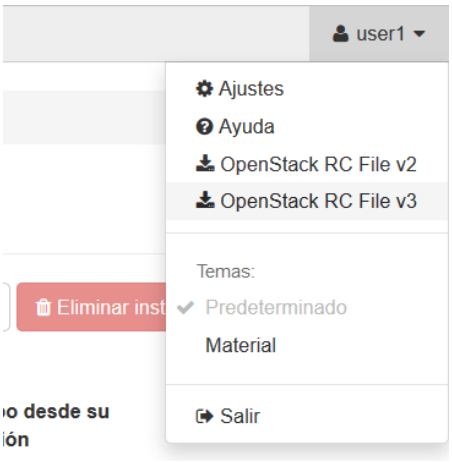


Figura 74 - Credenciales en formato "RC File"

```
1   #!/usr/bin/env bash
2   export OS_AUTH_URL=http://192.168.1.150:5000/v3
3
4   # With the addition of Keystone we have standardized on the term **project**
5   # as the entity that owns the resources.
6   export OS_PROJECT_ID=e2b58aba537547b78543c7d433988510
7   export OS_PROJECT_NAME="LAB1"
8   export OS_USER_DOMAIN_NAME="mydomain"
9   if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset OS_USER_DOMAIN_NAME; fi
10
11  # unset v2.0 items in case set
12  unset OS_TENANT_ID
13  unset OS_TENANT_NAME
14
15  # In addition to the owning entity (tenant), OpenStack stores the entity
16  # performing the action as the **user**.
17  export OS_USERNAME="user1"
18
19  # With Keystone you pass the keystone password.
20  echo "Please enter your OpenStack Password for project $OS_PROJECT_NAME as user $OS_USERNAME:
    "
```

```
21  read -sr OS_PASSWORD_INPUT
22  export OS_PASSWORD=$OS_PASSWORD_INPUT
23
24  # If your configuration has multiple regions, we set that information here.
25  # OS_REGION_NAME is optional and only valid in certain environments.
26  export OS_REGION_NAME="RegionOne"
27  # Don't leave a blank variable, unset it if it was empty
28  if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
29
30  export OS_INTERFACE=public
31  export OS_IDENTITY_API_VERSION=3
```

LAB1-openrc.sh

Se crea un script que contendrá las credenciales para OpenStack y a continuación ejecutará los comandos OpenStack CLI [19], para lanzar las VM. El código en el script ha sufrido algunas optimizaciones, para acelerar el lanzamiento de los servers (VM), ya que los comandos OpenStack en el CLI a través de SSH, pueden ser muy lentos.

```
1   #!/usr/bin/env bash
2
3   # To use an OpenStack cloud you need to authenticate against the Identity
4   # service named keystone, which returns a **Token** and **Service Catalog**.
5   # The catalog contains the endpoints for all services the user/tenant has
6   # access to - such as Compute, Image Service, Identity, Object Storage, Block
7   # Storage, and Networking (code-named nova, glance, keystone, swift,
8   # cinder, and neutron).
9   #
10  # *NOTE*: Using the 3 *Identity API* does not necessarily mean any other
11  # OpenStack API is version 3. For example, your cloud provider may implement
12  # Image API v1.1, Block Storage API v2, and Compute API v2.0. OS_AUTH_URL is
13  # only for the Identity API served through keystone.
14  export OS_AUTH_URL=http://192.168.1.150:5000/v3
15
16  # With the addition of Keystone we have standardized on the term **project**
17  # as the entity that owns the resources.
18  export OS_PROJECT_ID=e2b58aba537547b78543c7d433988510
19  export OS_PROJECT_NAME="LAB1"
20  export OS_USER_DOMAIN_NAME="mydomain"
21  if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset OS_USER_DOMAIN_NAME; fi
22
23  # unset v2.0 items in case set
24  unset OS_TENANT_ID
25  unset OS_TENANT_NAME
26
27  # In addition to the owning entity (tenant), OpenStack stores the entity
28  # performing the action as the **user**.
29  export OS_USERNAME="user1"
30
31  # With Keystone you pass the keystone password.
32  #echo "Enter your OpenStack Password for project $OS_PROJECT_NAME as user $OS_USERNAME: "
33  #read -sr OS_PASSWORD_INPUT
34  #export OS_PASSWORD=$OS_PASSWORD_INPUT
35  export OS_PASSWORD=secret
36
37  # If your configuration has multiple regions, we set that information here.
38  # OS_REGION_NAME is optional and only valid in certain environments.
39  export OS_REGION_NAME="RegionOne"
40  # Don't leave a blank variable, unset it if it was empty
41  if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
42
43  export OS_INTERFACE=public
44  export OS_IDENTITY_API_VERSION=3
45
46  #una de las principales cosas es obtener la IP del router del projecto y añadir la ruta para
    hacer ssh.
47  ID_router=$(openstack router list -f json --column ID --noindent | cut -d "\"" -f 4)
```

```
48  IP_router=$(openstack router show $ID_router -f value --column external_gateway_info --
    noindent | cut -d "\"" -f 16) # <--IP del router
49  #ID_subnetwork=$(openstack network list -f value --column Subnets) #Solo para redes con un
    solo router en el proyecto.
50  #IP_subnetwork=$(openstack subnet show $ID_subnetwork -f value --column cidr)
51  IP_subnetwork=$(openstack subnet list -f value --column Subnet)
52  # No hace falta comprobar si ya existe la ruta, te avisa que ya existe y no la añade.
53  #echo "Ruta añadida en OpenStack." No hace falta.
54  echo ""
55  echo "IP del Gateway para la red privada del laboratorio 1: $IP_router"
56  echo "Necesitas añadir una ruta a tu máguina (sudo):"
57  echo ""
58  echo "route del -net $IP_subnetwork"
59  echo "route add -net $IP_subnetwork gw $IP_router"
60  echo ""
61
62  i_instancias=$(openstack server list -f value --column ID | wc -l) # <- Número de instancias
    en el proyecto
63  Lista_ID_server=$(openstack server list -f value --column ID) # <-- Una lista con los server,
    y asi no ejecutamos varias veces el comando
64  for ((i=1;i<=$i_instancias;i++))
65  do
66  echo "Instancia $i/$i_instancias"
67  ID_server=$(echo "$Lista_ID_server" | awk '{if ( NR >= '$i' && NR <= '$i' ) { print $0 }}')
68  server=$(openstack server show $ID_server -f yaml) #Se guarda toda la informacion del server
    y nos ahorramos instrucciones openstack.
69  openstack server start $ID_server & # <-En un hilo a parte, no nos importa el resultado ni
    tiene variables de entorno.
70  #Name_server=$(openstack server show $ID_server -f value --column name)
71  Name_server=$(echo "$server"|grep -w "name:" | cut -d ":" -f 2)
72  #IP_server=$(openstack server show $ID_server --column addresses -f value | cut -d "=" -f 2)
73  IP_server=$(echo "$server"|grep -w "addresses:" | cut -d "=" -f 2)
74  echo "Dirección IP de la instancia $Name_server: $IP_server"
75  echo ""
76  done
```

LAB1-Start.sh

Se establece una conexión SSH con el Controller y se le pasa el script LAB1-Start.sh para que lo ejecute. El script devuelve básicamente dos cosas, las nuevas rutas que necesitan para acceder a las VM del proyecto.

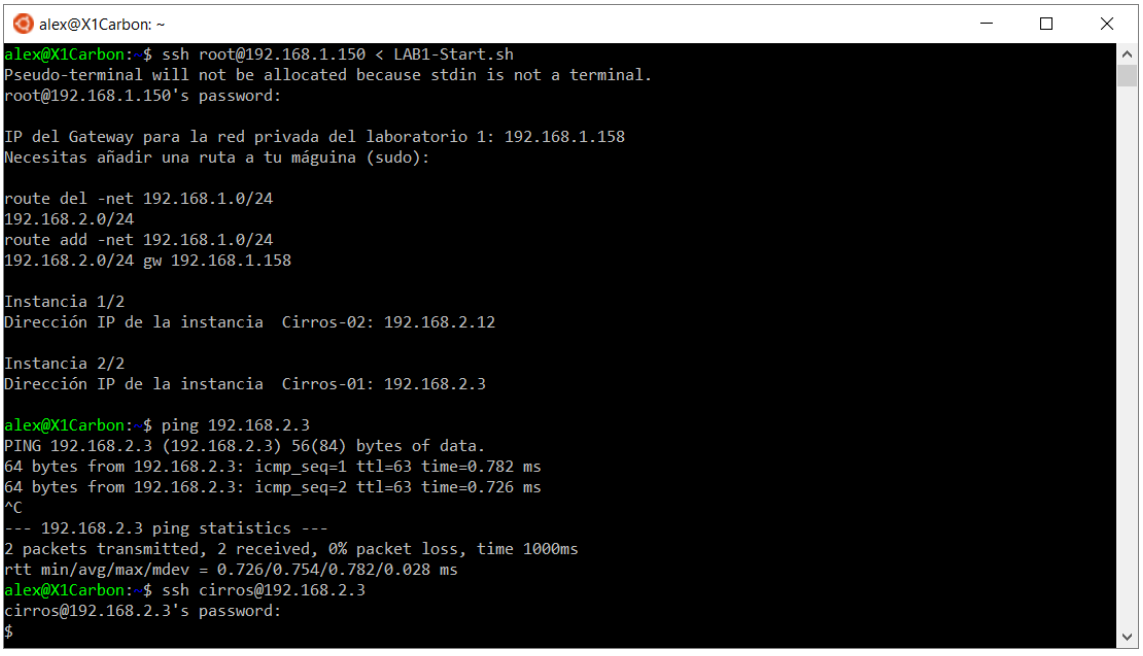


Figura 75 - Lanzamiento del laboratorio virtual desde la red externa (Windows con WSL)

```
#  alex@ALEX-VAIO:~/TFM$ ssh root@192.168.1.150 < LAB1-openrc.sh
 Pseudo-terminal will not be allocated because stdin is not a terminal.
 root@192.168.1.150's password:

 IP del Gateway para la red privada del laboratorio 1: 192.168.1.158
 Necesitas añadir una ruta a tu máguina (sudo):

 route del -net 192.168.2.0/24
 route add -net 192.168.2.0/24 gw 192.168.1.158

 Instancia 1/2
 Dirección IP de la instancia cirros-002: 192.168.2.12

 Instancia 2/2
 Dirección IP de la instancia cirros-001: 192.168.2.3

#  alex@ALEX-VAIO:~/TFM$ sudo -i
 [sudo] password for alex:
#  root@ALEX-VAIO:~# route del -net 192.168.2.0/24
#  root@ALEX-VAIO:~# route add -net 192.168.2.0/24 gw 192.168.1.158
#  root@ALEX-VAIO:~# exit
 logout
#  alex@ALEX-VAIO:~/TFM$ ssh cirros@192.168.2.3
 cirros@192.168.2.3's password:
 $
```

Lanzamiento desde el terminal de una máquina con Ubuntu

Para los demás proyectos el procedimiento es el mismo. En el caso de la red privada del proyecto LAB2, se ha decidido utilizar 192.168.3.0/24 como CIDR.

5. Ejemplo práctico

Como ejemplo de aplicación práctica de un laboratorio virtual en la nube, se va a montar un laboratorio con dos máquinas. Este laboratorio está centrado en la seguridad de las redes y sistemas en la nube, para ellos una de las máquinas será un atacante (Attacker) y la otra máquina será una víctima (Victim). Ambas máquinas se crean en proyectos del dominio LDAP, *mydomain.*

En este ejemplo en concreto cada una de las máquinas estará en un proyecto distinto teniendo que atravesar 2 routers externos, el router de cada proyecto, para poder verse el uno al otro.

El sistema atacante será una VM con una distribución de Kali-Linux instalada, que es una distribución que se centra en la seguridad ofensiva y viene cargada con muchas herramientas y aplicaciones para comprometer la seguridad y el análisis.

Mientras el sistema victima contará con una distribución de Metasploitable, una versión de Ubuntu Linux que es intencionalmente vulnerable, diseñado para probar herramientas de seguridad y demostrar vulnerabilidades comunes.

5.1. Kali Linux

Es una distribución basada en Debian que está diseñada principalmente para auditoría y seguridad informática. En la distribución se incluyen muchos programas para escanear puertos, capturador de paquetes y analizador de protocolos, la suite Aircrack-ng, entre otros.

Para instalarlo se pude hacer uso de la ISO que proveen, o desde máquinas virtuales prefabricadas para montarlas de nuevo en virtualizadores, como VMWare o VirtualBox. En este caso se va a utilizar una máquina prefabricada, porque el instalar desde el ISO, en el caso de Kali-Linux puede ser complicado si no se hace desde CD o no se tiene disquetera donde montar el disco, como en OpenStack.

Se descarga un archivo OVA de la página web de Offensive Security, en el apartado para las VM prefabricadas se selecciona la pestaña para VirtualBox. https://www.kali.org/downloads/

En principio OpenStack puede aceptar archivos OVA, pero en este caso falla la subida a Glance.

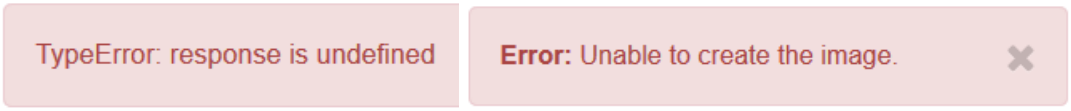


Figura 76 - Error al importar la imagen OVA

Se descomprime el archivo con algún programa como 7z, y entre los archivos descomprimidos deberían aparecer dos archivos un OVF, que no tiene mucho tamaño y parecen más bien metadatos o una plantilla de la máquina virtual, conteniendo información como UUID o identificando el nombre del disco duro virtual y su capacidad, entre otras cosas. El segundo archivo es el que interesa, se trata de un archivo en formato VMDK (Virtual Machine DisK). Este es el disco virtual que contiene la instalación de Kali Linux y es esto fichero el que se importa en OpenStack, que también acepta VMDK.

Entre los datos que se puede ver en el fichero OVF, uno de ellos es la capacidad del disco, esto es necesario a la hora de crear un volumen con suficiente capacidad en OpenStack.

Disk ovf:capacity="42949672960" o lo que es lo mismo 40 GiB.

Debido a las limitaciones hardware, se creará un sabor adicional con suficiente espacio de almacenamiento, CPU y RAM. El principal problema es la RAM.

Estas dos cosas, cargar la imagen y crear el sabor, requieren de permisos de administración, por lo que se hace desde el usuario *admin* del dominio “Default”. Es probable que no se necesite el nuevo sabor, ya que como capacidad de almacenamiento se va a adjuntar el volumen de Kali.

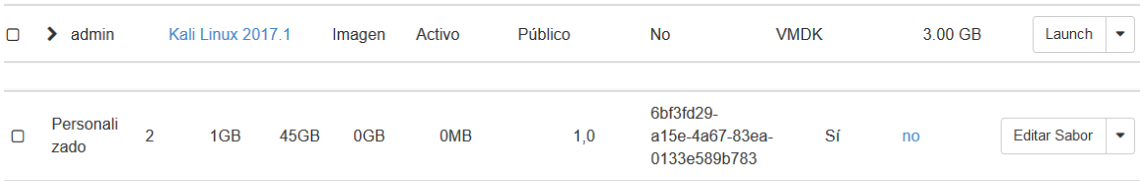


Figura 77 - Nueva Imagen y nuevo sabor para Kali Linux

Ya que no se crea automáticamente y a hay que hacerlo a mano. La instancia se crea a partir del volumen. Esto ya es en el usuario LDAP, *user1*, para el proyecto LAB1.

Lo primero es crear un volumen de por lo menos 40 GiB y cargar la imagen de Kali Linux en él.

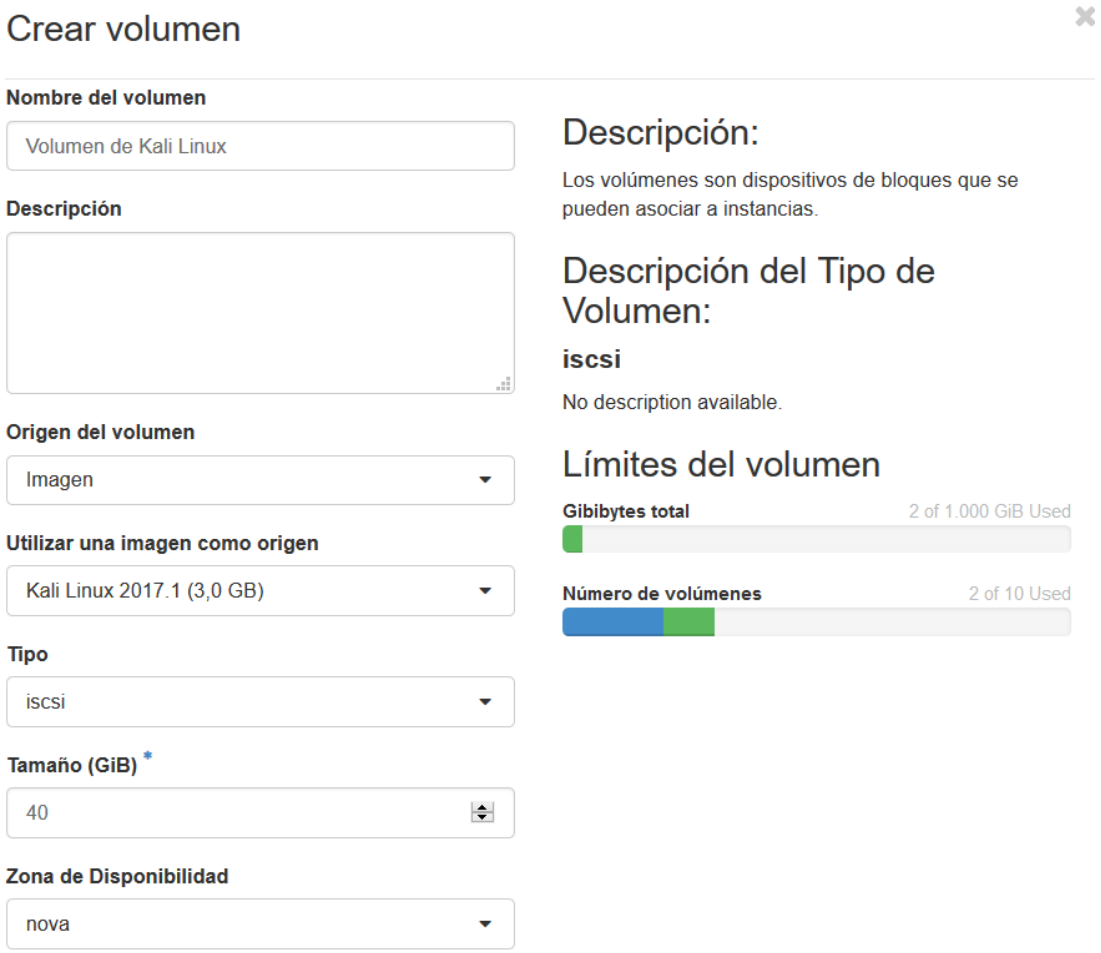


Figura 78 - Creación de volumen para Kali Linux

Como es un volumen grande y tiene mucho que descomprimir de la imagen VMDK, puede tardar un rato. Y después el volumen se ejecuta como instancia.

Figura 79 - El volumen se ejecuta como instancia

Esto convierte en arrancable el volumen y lo asocia con la instancia que se cree en el menú que aparece. Pero durante la secuencia de arranque surge un problema.

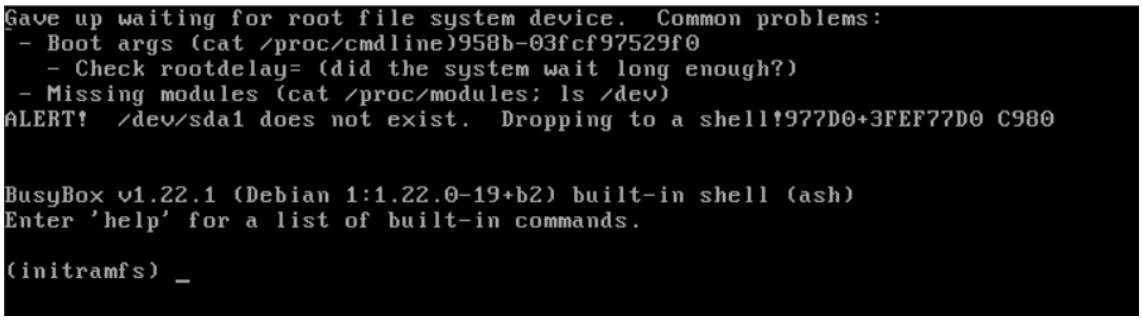


Figura 80 - Fallo durante el boot de Kali Linux

Este error advierte de que no existe la partición */dev/sda1*. Esto es debido a que la imagen tiene montado el sistema de archivos sobre la ruta el dispositivo */dev/sda* y OpenStack monta los volúmenes en otro dispositivo, */dev/vda*.

☐	Nombre	Descripción	Tamaño	Estado	Tipo	Asociado a
☐	Volumen de Kali Linux	-	40GiB	En-uso	iscsi	/dev/vda on Kali Linux

Figura 81 - Asociación del volumen Kali Linux en OpenStack

Para arreglar esto se pueden hacer varias cosas, como utilizar una imagen preparada para OpenStack como la ofrecida en *tuxfixer.com* [20], a cambio de utilizar versiones que no están completamente actualizadas y que no provienen de la página oficial, u otras opciones como asociar el volumen a una máquina que funcione y editar los ficheros de arranque, pero estos no deberían ser modificados a mano. Lo más sencillo es reiniciar la máquina y arrancar a mano el Sistema Operativo. [21]

Figura 82 - GRUB 2 de Kali Linux

El GRUB solo sale durante unos pocos segundos y muchas veces se acaba antes de que arranque una consola noVNC donde poder parar la cuenta atrás y arrancar el editor del BOOT o una línea de comandos. Esto se puede solventar reiniciando la máquina en frío sin cerrar la pestaña donde estaba la consola noVNC y mientras está reiniciando actualizar continuamente la página hasta que conecte con el "handshake" y aparezca el GRUB. Inmediatamente después parar la cuenta atrás pulsando una tecla. La línea de comandos se abre pulsando la tecla "c".

En este caso es más sencillo desde la línea de comandos si se sabe hacer.

```
grub> set root=(hd0,1)
grub> linux /boot/vmlinuz-4.9.0-kali3-amd64 root=/dev/vda1
grub> initrd /boot/initrd.img-4.9.0-kali3-amd64
grub> boot
```

La primera línea establece la partición en la que la raíz del sistema de ficheros se encuentra.

La segunda línea le dice a GRUB donde está el kernel que se quiere usar. Se puede empezar escribiendo */boot/vmli* y utilizar la tecla TAB para autocompletar el resto. Seguido en la misma línea se escribe la localización de la raíz del sistema de ficheros, en el dispositivo, esto se puede ver en la Figura 81 y añadiendo un "1" indicando la primera partición. Esto es importante para evitar que el kernel no entre en pánico.

La tercera línea configura el fichero initrd, que tiene que ser la misma versión que el kernel.

La cuarta línea arranca el sistema.

El sistema arranca correctamente, entendiendo correctamente que esta vez, si es capaz de identificar el dispositivo */dev/vda1*. Pero rápidamente se queda trabado en un proceso en el arranque. [22]

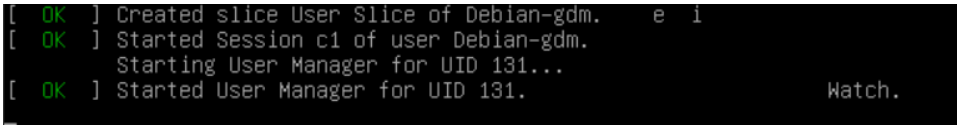


Figura 83 - Se queda trabado intentado arrancar "User Manager for UID 131"

Para arreglar esto basta con seguir los siguientes pasos para desactivar Wayland. La forma más sencilla es cambiar de TTY, pero no siempre funciona. Otra forma es modificar la imagen que se sube a OpenStack desde VirtualBox. La que se va a utilizar en este trabajo es posiblemente la más sencilla en caso de que no funcione el cambio de TTY, esto es montar el volumen en una máquina que funcione y editar los ficheros desde allí.

Se ha montado una VM con CentOS 7 minimal, pero esto no es muy importante, lo que interesa es que se pueda la montar el volumen en el sistema. Se puede usar una instancia que ya exista o se puede crear una temporalmente.

Primero, se adjunta el volumen con el sistema de ficheros de Kali Linux, a la instancia donde se van a modificar los ficheros. En este caso el volumen ha quedado en */dev/vdb*.

Se monta el volumen con los ficheros de Kali.

```
#  mkdir temp
#  mount /dev/vdb1 temp/
#  ll temp/
```

```
total 100
-rw-r--r--.   1 root root     0 abr 16 03:46 0
drwxr-xr-x.   2 root root  4096 abr 22 11:23 bin
drwxr-xr-x.   3 root root  4096 abr 22 11:23 boot
drwxr-xr-x.   4 root root  4096 abr 22 11:12 dev
drwxr-xr-x. 180 root root 12288 ago  3 22:06 etc
drwxr-xr-x.   2 root root  4096 abr  5 09:44 home
lrwxrwxrwx.   1 root root    33 abr 22 11:12 initrd.img -> boot/initrd.img-4.9.0-kali3-amd64
lrwxrwxrwx.   1 root root    33 abr 22 11:12 initrd.img.old -> boot/initrd.img-4.9.0-kali3-amd64
drwxr-xr-x.  18 root root  4096 abr 22 11:26 lib
drwxr-xr-x.   2 root root  4096 abr 22 11:12 lib64
drwx------.   2 root root 16384 abr 22 11:12 lost+found
drwxr-xr-x.   3 root root  4096 abr 16 03:21 media
drwxr-xr-x.   2 root root  4096 abr 16 03:21 mnt
drwxr-xr-x.   3 root root  4096 abr 22 11:12 opt
drwxr-xr-x.   2 root root  4096 abr  5 09:44 proc
drwxr-xr-x.  15 root root  4096 abr 22 12:18 root
drwxr-xr-x.   2 root root  4096 abr 22 11:23 run
drwxr-xr-x.   2 root root  4096 abr 22 11:26 sbin
drwxr-xr-x.   3 root root  4096 abr 22 11:12 srv
drwxr-xr-x.   2 root root  4096 abr  5 09:44 sys
drwxrwxrwt.   9 root root  4096 ago  3 22:07 tmp
drwxr-xr-x.  12 root root  4096 abr 22 11:18 usr
drwxr-xr-x.  13 root root  4096 abr 22 11:18 var
lrwxrwxrwx.   1 root root    30 abr 22 11:18 vmlinuz -> boot/vmlinuz-4.9.0-kali3-amd64
lrwxrwxrwx.   1 root root    30 abr 22 11:18 vmlinuz.old -> boot/vmlinuz-4.9.0-kali3-amd64
```

Se abre un terminal y se modifica el archivo */etc/gdm3/daemon.conf*. Como en este caso el volumen se ha montado en */root/temp/* la ruta completa del fichero a modificar es la siguiente:

```
#  nano /root/temp/etc/gdm3/daemon.conf
```

```
1   # GDM configuration storage - modified by kali-root-login
2   #
3   # See /usr/share/gdm/gdm.schemas for a list of available options.
4
5   [daemon]
6   # Enabling automatic login
7   #  AutomaticLoginEnable = true
8   #  AutomaticLogin = root
9   WaylandEnable = false
10  # Enabling timed login
11  #  TimedLoginEnable = true
12  #  TimedLogin = user1
13  #  TimedLoginDelay = 10
14
15  # Reserving more VTs for test consoles (default is 7)
16  #  FirstVT = 9
17
18  [security]
19  AllowRoot = true
20
21  [xdmcp]
22
23  [greeter]
24  # Only include selected logins in the greeter
25  # IncludeAll = false
26  # Include = user1,user2
27
28  [chooser]
29
30  [debug]
31  # More verbose logs
32  # Additionally lets the X server dump core if it crashes
33  #  Enable = true
```

daemon.conf

Se apaga la instancia, se desmonta el volumen y se vuelve a crear la instancia de Kali Linux.

Siguiendo los mismos pasos que antes, se puede arrancar el sistema desde el GRUB2. La diferencia es que esta vez sí arranca el sistema en vez de quedarse trabado, y usando las credenciales por defecto (usuario *root* y contraseña *toor*) para Kali Linux se puede acceder a la cuanta de administración.

Figura 84 - Kali Linux del proyecto LAB1 en OpenStack

Lo primero que se debe hacer es actualizar las instrucciones de la secuencia de arranque, para no tener que iniciar manualmente Kali Linux. Para esto basta con actualizar el GRUB.

```
root@kali:~# update-grub
Generating grub configuration file ...
Found background image: /usr/share/images/desktop-base/desktop-grub.png
Found linux image: /boot/vmlinuz-4.9.0-kali3-amd64
Found initrd image: /boot/initrd.img-4.9.0-kali3-amd64
done
```

Figura 85 - Actualizar arranque (GRUB) de Kali Linux

Ahora funciona correctamente la VM de Kali Linux en OpenStack.

Tarda mucho en cerrar la máquina porque algunos servicios no responden. Estos servicios están relacionados con los que impedían el arranque del sistema y se tuvo que desactivar Wayland. Para acelerar el proceso de cierre se pueden modificar los siguientes parámetros para los tiempos de *start* y *stop* en el fichero */etc/systemd/system.conf*. [23]

```
      . . .
34  DefaultTimeStart=10
35  DefaultTimeStop=10
      . . .
```

system.conf

5.2. Metasploitable

Uno de los problemas que se encuentran al aprender a utilizar un framework de explotación, es configurar los objetivos de escaneo y ataque. Por suerte, el equipo de Metasploit ha alanzado una máquina virtual vulnerable (Metasploitable), para atender este problema.

Metasploitable es una máquina virtual Linux basada en Ubuntu que es intencionalmente vulnerable. Esta VM se puede utilizar para realizar capacitación en seguridad, probar herramientas de seguridad y practicar técnicas comunes de pruebas de penetración. La máquina virtual se ejecutará en un entorno virtual, como VMware o VirtualBox, pero en este trabajo se ejecutará sobre QEMU en OpenStack. Por defecto esta máquina nunca debe ser expuesta a redes hostiles.

La descarga está disponible en *sourceforge.net* en el siguiente enlace:

- `https://sourceforge.net/projects/metasploitable/files/Metasploitable2/metasploitable-linux-2.0.0.zip/download`

Igual que en los casos anteriores, lo primero que se hace es subir la imagen de la VM a Glance. La forma más sencilla es a través del dashboard, pero si se tiene que descargar se puede hacer como con la imagen de cirros al final del capítulo 4.3, pero en pasos diferentes ya que Metasploitable vine comprimido en un ZIP. Las imágenes se suben desde el proyecto admin, donde se encuentran todas las imágenes y no desde el propio proyecto.

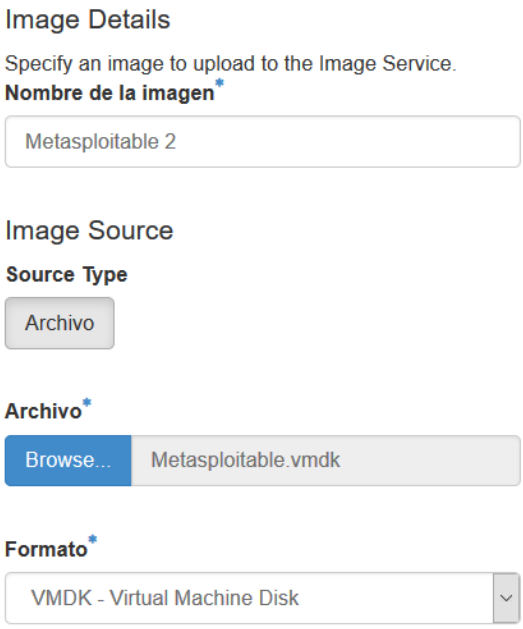


Figura 86 - Subir Imagen a OpenStack

	Owner	Nombre	Tipo	Estado	Visibilidad	Protegido	Disk Format	Tamaño	
☐	admin	CentOS 7 minimal	Imagen	Activo	Público	No	ISO	680.00 MB	Launch
☐	admin	cirros image	Imagen	Activo	Público	No	QCOW2	12.67 MB	Launch
☐	admin	Kali Linux	Imagen	Activo	Público	No	VMDK	3.00 GB	Launch
☐	admin	Metasploitable 2	Imagen	Activo	Público	No	VMDK	1.81 GB	Launch

Figura 87 - Imágenes en OpenStack

Una vez subida la imagen, se cambia al proyecto LAB2 y se crea un volumen con la imagen VMDK de Metasploitable. El volumen que se crea es de 8GiB.

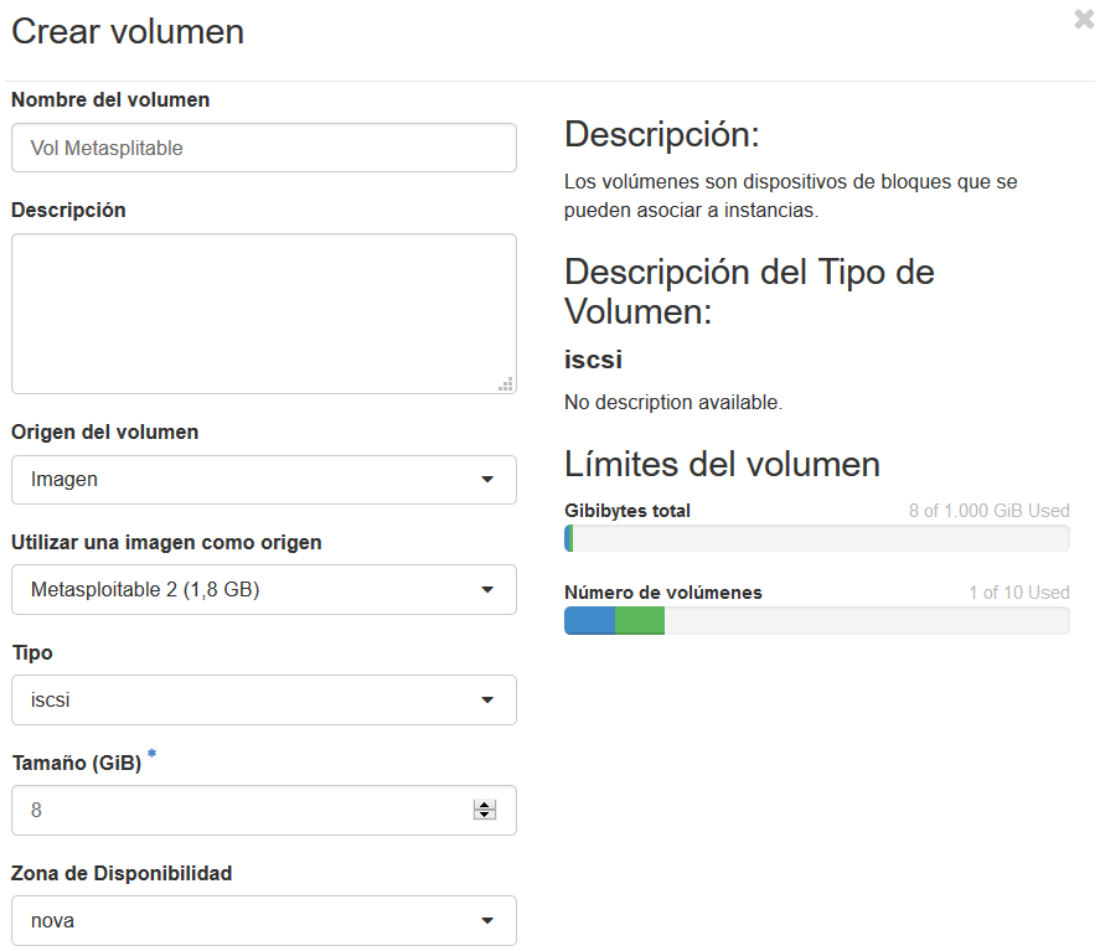


Figura 88 - Se crea el volumen para Metasploitable

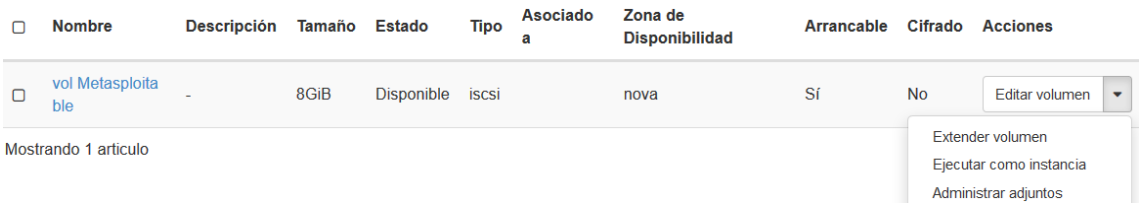


Figura 89 - Se crea una instancia del nuevo volumen

Se usa un sabor “m1.tiny” con 1 vCPU, 512 MB de RAM y un disco de 1GB, esto no importa porque se le está adjuntando un volumen de 8GiB con el filesystem.

En esta nueva instancia que se acaba de crear, se da el mismo problema que en Kali Linux. El sistema está intentando encontrar el sistema de ficheros en un dispositivo y una partición que no existen */dev/sda1*. El volumen de Metasploitable se ha adjuntado en la ruta */dev/vda1*.

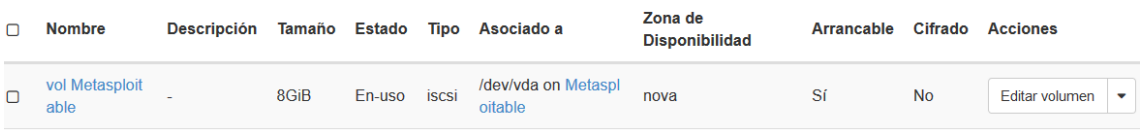


Figura 90 - Volumen de Metasploitable adjuntado en la ruta /dev/vda

Afortunadamente, después de fallar en arrancar la instancia, inicia una Shell de mantenimiento desde la que se puede arrancar la máquina virtual. Para arrancarla basta con pulsar `Ctrl`+D.

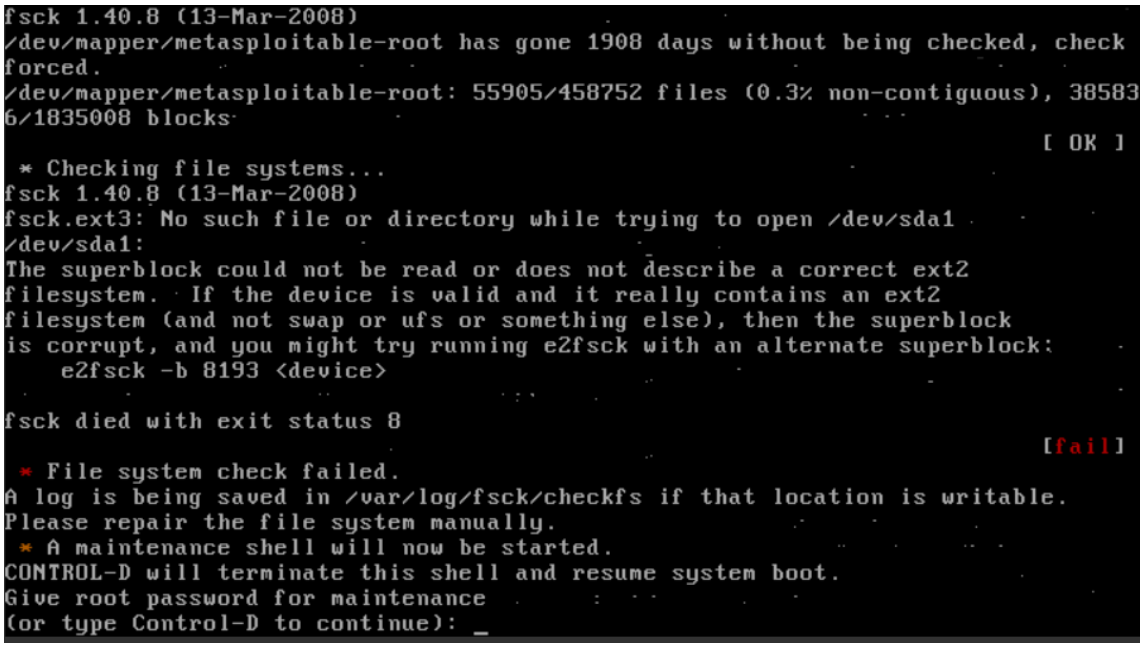


Figura 91 - Fallo al arrancar Metasploitable y Shell de mantenimiento

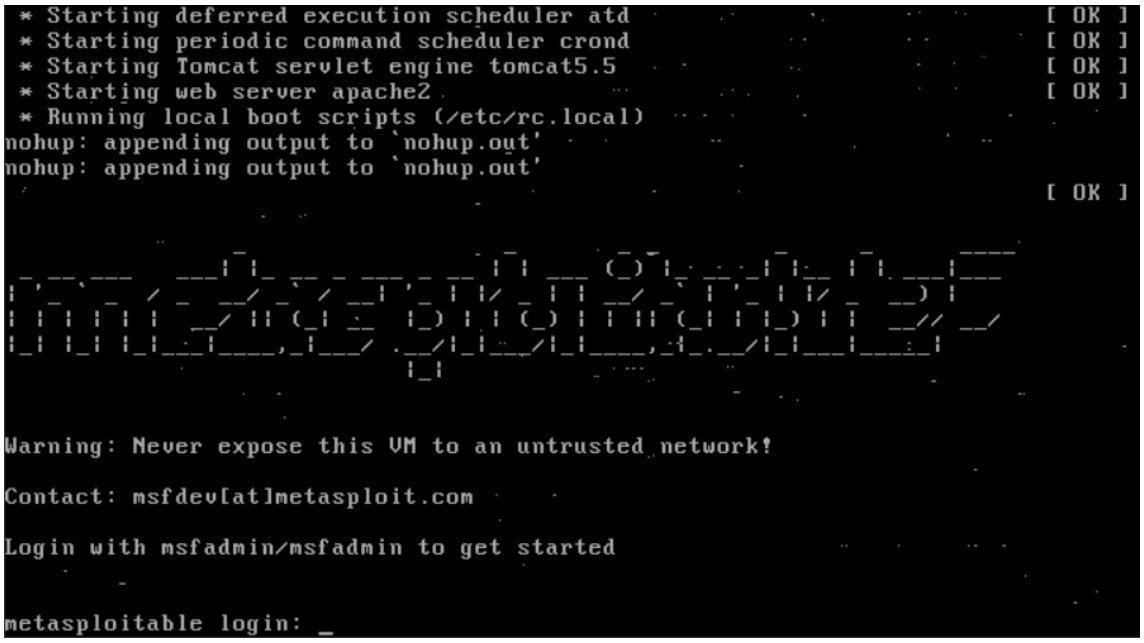


Figura 92 - Metasploitable arrancado a través de la Shell de mantenimiento

Las credenciales por defecto de Metasploitable son *msfadmin* como usuario y contraseña, tal y como se puede leer en la Figura 92.

Para arreglar el inicio de esta máquina virtual lo más fácil es modificar en el fichero */etc/fstab* cuál es el dispositivo del arranque.

```
$   sudo nano /etc/fstab
$   [sudo] password for msfadmin:
```

```
1   # /etc/fstab: static file system information.
2   #
3   # <file system> <mount point>   <type>  <options>       <dump>  <pass>
4   proc            /proc           proc    defaults        0       0
5   # /dev/mapper/metasploitabale-root
6   UUID=59bd36ce-2d78-44fe-843f-a4ca5fcafad1 /               ext3    relatime,errors=remount-ro
    0       1
7   /dev/vda1 /boot           ext3    relatime        0       2
8   /dev/scd0       /media/cdrom0   udf,iso9660 user,noauto,exec,utf8 0       0
9   /dev/fd0        /media/floppy0  auto    rw,user,noauto,exec,utf8 0       0
```

fstab

5.3. Conectividad

Una vez se tienen las instancias atacante y víctima, hay que probar la conectividad entre ambas instancias. Como cada instancia está en una red privada diferente, no saben cómo llegar a la red de la otra VM.

```
root@kali:~# ping -c 3 192.168.3.6
PING 192.168.3.6 (192.168.3.6) 56(84) bytes of data.
From 80.58.67.122 icmp_seq=1 Packet filtered
From 80.58.67.122 icmp_seq=2 Packet filtered
From 80.58.67.122 icmp_seq=3 Packet filtered

--- 192.168.3.6 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2002ms
```

Figura 93 - PING a Metasploitable desde Kali Linux

```
msfadmin@metasploitable:~$ ping -c 3 192.168.2.13
PING 192.168.2.13 (192.168.2.13) 56(84) bytes of data.
From 80.58.67.122 icmp_seq=1 Packet filtered
From 80.58.67.122 icmp_seq=2 Packet filtered
From 80.58.67.122 icmp_seq=3 Packet filtered

--- 192.168.2.13 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 1998ms
```

Figura 94 - PING a Kali Linux desde Metasploitable

Ninguna de las máquinas tiene conectividad con la otra. Esto es debido a que no saben por dónde ir a la otra red privada, por lo que hay que añadir rutas.

Así están las redes en este momento.

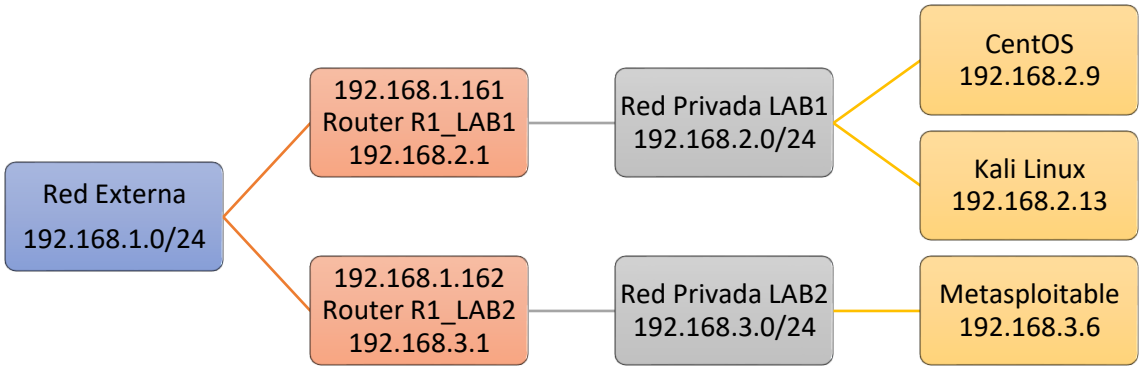


Figura 95 - Configuración de las redes

Para añadir las rutas hay que dirigirse al proyecto y dentro del apartado de “Red” se escoge “Routers”. En el caso del proyecto LAB1, el router es R1_LAB1 y se abre para modificar “Rutas

estáticas". Al cual se le añade la nueva ruta, indicando el CIDR de la red privada del proyecto LAB2 y la IP externa del router R1_LAB2.

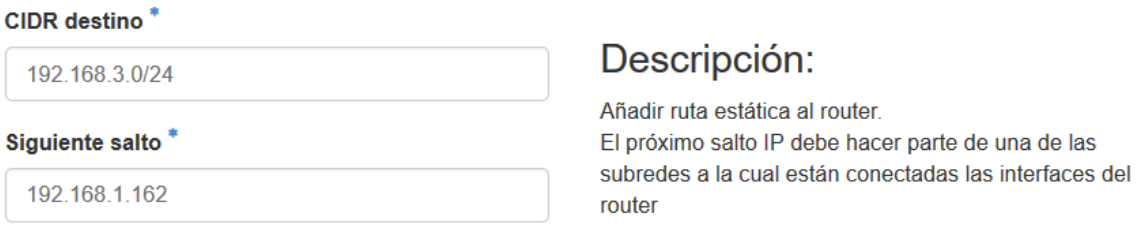


Figura 96 - Ruta estática del proyecto LAB1 al proyecto LAB2

Se cambia al proyecto LAB2 y se repite la operación, pero para una ruta al proyecto LAB1.

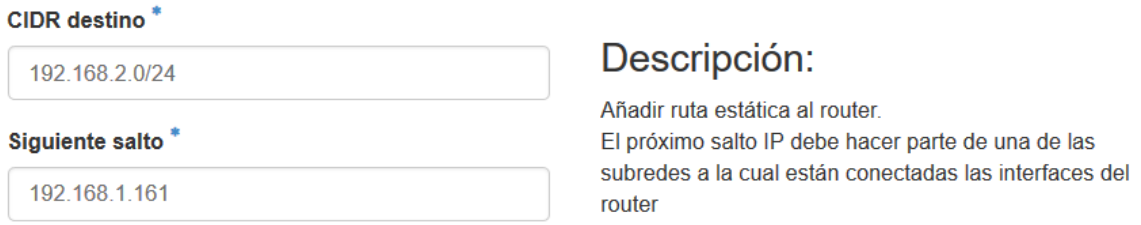


Figura 97 - Ruta estática del proyecto LAB2 al proyecto LAB1

Con las nuevas rutas establecidas ya se puede hacer PING para probar la conectividad.

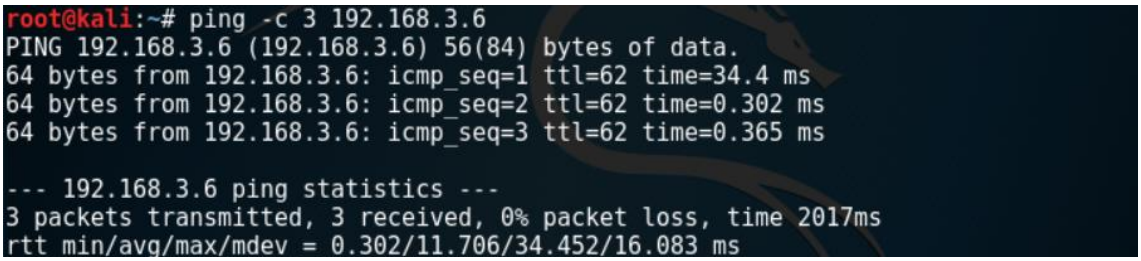


Figura 98 - PING a Metasploitable desde Kali Linux

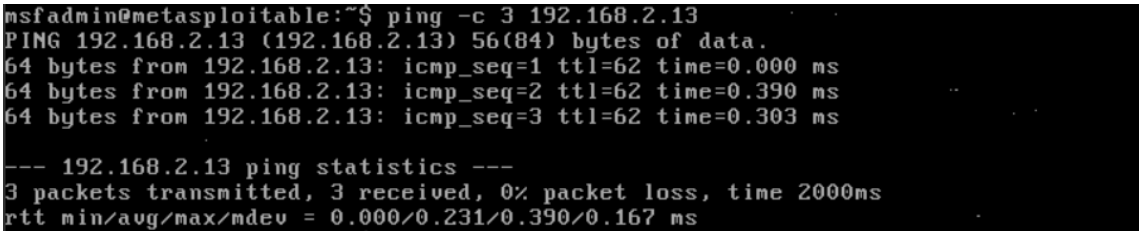


Figura 99 - PING a Kali Linux desde Metasploitable

Con esto ya están preparados los laboratorios virtuales.

Otra opción habría sido configurarlo todo en el mismo proyecto, creando más redes privadas y routers para conectar las redes privadas.

Después de comprobar la conectividad de las máquinas se puede automatizar de la misma forma que en ejemplo del capítulo 4.5. En este caso cada máquina está en un proyecto diferente por lo que habrá que descargarse los ficheros "RC file" de cada uno de los proyectos. Hay que cambiar a mitad de ejecución el proyecto que se está utilizando, pero si el usuario es el mismo no hace falta volver a escribir lo mismo dos veces. El script quedaría algo así:

```
#!/usr/bin/env bash

# To use an OpenStack cloud you need to authenticate against the Identity
# service named keystone, which returns a **Token** and **Service Catalog**.
# The catalog contains the endpoints for all services the user/tenant has
# access to - such as Compute, Image Service, Identity, Object Storage, Block
# Storage, and Networking (code-named nova, glance, keystone, swift,
# cinder, and neutron).
#
# *NOTE*: Using the 3 *Identity API* does not necessarily mean any other
# OpenStack API is version 3. For example, your cloud provider may implement
# Image API v1.1, Block Storage API v2, and Compute API v2.0. OS_AUTH_URL is
# only for the Identity API served through keystone.
export OS_AUTH_URL=http://192.168.1.150:5000/v3

# With the addition of Keystone we have standardized on the term **project**
# as the entity that owns the resources.
export OS_PROJECT_ID=fe703fe959be4a84b132036a88c3c58f
export OS_PROJECT_NAME="LAB1"
export OS_USER_DOMAIN_NAME="mydomain"
if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset OS_USER_DOMAIN_NAME; fi

# unset v2.0 items in case set
unset OS_TENANT_ID
unset OS_TENANT_NAME

# In addition to the owning entity (tenant), OpenStack stores the entity
# performing the action as the **user**.
export OS_USERNAME="user1"

# With Keystone you pass the keystone password.
#echo "Enter your OpenStack Password for project $OS_PROJECT_NAME as user $OS_USERNAME: "
#read -sr OS_PASSWORD_INPUT
#export OS_PASSWORD=$OS_PASSWORD_INPUT
export OS_PASSWORD=secret

# If your configuration has multiple regions, we set that information here.
# OS_REGION_NAME is optional and only valid in certain environments.
export OS_REGION_NAME="RegionOne"
# Don't leave a blank variable, unset it if it was empty
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi

export OS_INTERFACE=public
export OS_IDENTITY_API_VERSION=3

#una de las principales cosas es obtener la IP del router del projecto y añadir la ruta para hacer ssh.
ID_router=$(openstack router list -f json --column ID --noindent | cut -d "\"" -f 4)
IP_router=$(openstack router show $ID_router -f value --column external_gateway_info --noindent | cut -d "\"" -f 16) # <--IP del router
#ID_subnetwork=$(openstack network list -f value --column Subnets) #Solo para redes con un solo router en el proyecto.
#IP_subnetwork=$(openstack subnet show $ID_subnetwork -f value --column cidr)
IP_subnetwork=$(openstack subnet list -f value --column Subnet)
# No hace falta comprobar si ya existe la ruta, te avisa que ya existe y no la añade.
#echo "Ruta añadida en OpenStack." No hace falta.
echo ""
echo "IP del Gateway para la red privada del laboratorio 1: $IP_router"
echo "Necesitas añadir una ruta a tu máguina (sudo):"
echo ""
echo "route del -net $IP_subnetwork"
echo "route add -net $IP_subnetwork gw $IP_router"
echo ""

i_instancias=$(openstack server list -f value --column ID | wc -l) # <- Número de instancias en el proyecto
Lista_ID_server=$(openstack server list -f value --column ID) # <-- Una lista con los server, y asi no ejecutamos varias veces el comando
for ((i=1;i<=$i_instancias;i++))
```

```
65  do
66  echo "Instancia $i/$i_instancias"
67  ID_server=$(echo "$Lista_ID_server" | awk '{if ( NR >= '$i' && NR <= '$i' ) { print $0 }}')
68  server=$(openstack server show $ID_server -f yaml) #Se guarda toda la informacion del server
    y nos ahorramos instrucciones openstack.
69  openstack server start $ID_server & # <-En un hilo a parte, no nos importa el resultado ni
    tiene variables de entorno.
70  #Name_server=$(openstack server show $ID_server -f value --column name)
71  Name_server=$(echo "$server"|grep -w "name:" | cut -d ":" -f 2)
72  #IP_server=$(openstack server show $ID_server --column addresses -f value | cut -d "=" -f 2)
73  IP_server=$(echo "$server"|grep -w "addresses:" | cut -d "=" -f 2)
74  echo "Dirección IP de la instancia $Name_server: $IP_server"
75  echo ""
76  done
77
78  #Laboratorio 2
79  export OS_PROJECT_ID=0e794263f4b0408dbe5440af5550a1fc
80  export OS_PROJECT_NAME="LAB2"
81
82  #una de las principales cosas es obtener la IP del router del projecto y añadir la ruta para
    hacer ssh.
83  ID_router=$(openstack router list -f json --column ID --noindent | cut -d "\"" -f 4)
84  IP_router=$(openstack router show $ID_router -f value --column external_gateway_info --
    noindent | cut -d "\"" -f 16) # <--IP del router
85  #ID_subnetwork=$(openstack network list -f value --column Subnets) #Solo para redes con un
    solo router en el proyecto.
86  #IP_subnetwork=$(openstack subnet show $ID_subnetwork -f value --column cidr)
87  IP_subnetwork=$(openstack subnet list -f value --column Subnet)
88  # No hace falta comprobar si ya existe la ruta, te avisa que ya existe y no la añade.
89  #echo "Ruta añadida en OpenStack." No hace falta.
90  echo ""
91  echo "IP del Gateway para la red privada del laboratorio 2: $IP_router"
92  echo "Necesitas añadir una ruta a tu máguina (sudo):"
93  echo ""
94  echo "route del -net $IP_subnetwork"
95  echo "route add -net $IP_subnetwork gw $IP_router"
96  echo ""
97
98  i_instancias=$(openstack server list -f value --column ID | wc -l) # <- Número de instancias
    en el proyecto
99  Lista_ID_server=$(openstack server list -f value --column ID) # <-- Una lista con los server,
    y asi no ejecutamos varias veces el comando
100 for ((i=1;i<=$i_instancias;i++))
101 do
102 echo "Instancia $i/$i_instancias"
103 ID_server=$(echo "$Lista_ID_server" | awk '{if ( NR >= '$i' && NR <= '$i' ) { print $0 }}')
104 server=$(openstack server show $ID_server -f yaml) #Se guarda toda la informacion del server
    y nos ahorramos instrucciones openstack.
105 openstack server start $ID_server & # <-En un hilo a parte, no nos importa el resultado ni
    tiene variables de entorno.
106 #Name_server=$(openstack server show $ID_server -f value --column name)
107 Name_server=$(echo "$server"|grep -w "name:" | cut -d ":" -f 2)
108 #IP_server=$(openstack server show $ID_server --column addresses -f value | cut -d "=" -f 2)
109 IP_server=$(echo "$server"|grep -w "addresses:" | cut -d "=" -f 2)
110 echo "Dirección IP de la instancia $Name_server: $IP_server"
111 echo ""
112 done
```

Ejemplo_practico.sh

El script está compuesto de dos partes, accede a un proyecto en cada parte y enciendo todas las máquinas. Esto puede ser lo que se desea o no, pero con las instrucciones que hay en el script se debería poder identificar las máquinas de un proyecto. Puede que el script hay que modificarlo para ajustarlo a la funcionalidad que se busque, pero la forma de comunicarse con OpenStack, la forma de lanzar los comandos CLI y la forma de recibir las respuestas ya están solucionado en el script propuesto.

En cuanto a lanzar las máquinas a través de SSH, es igual que en el capítulo 4.5.

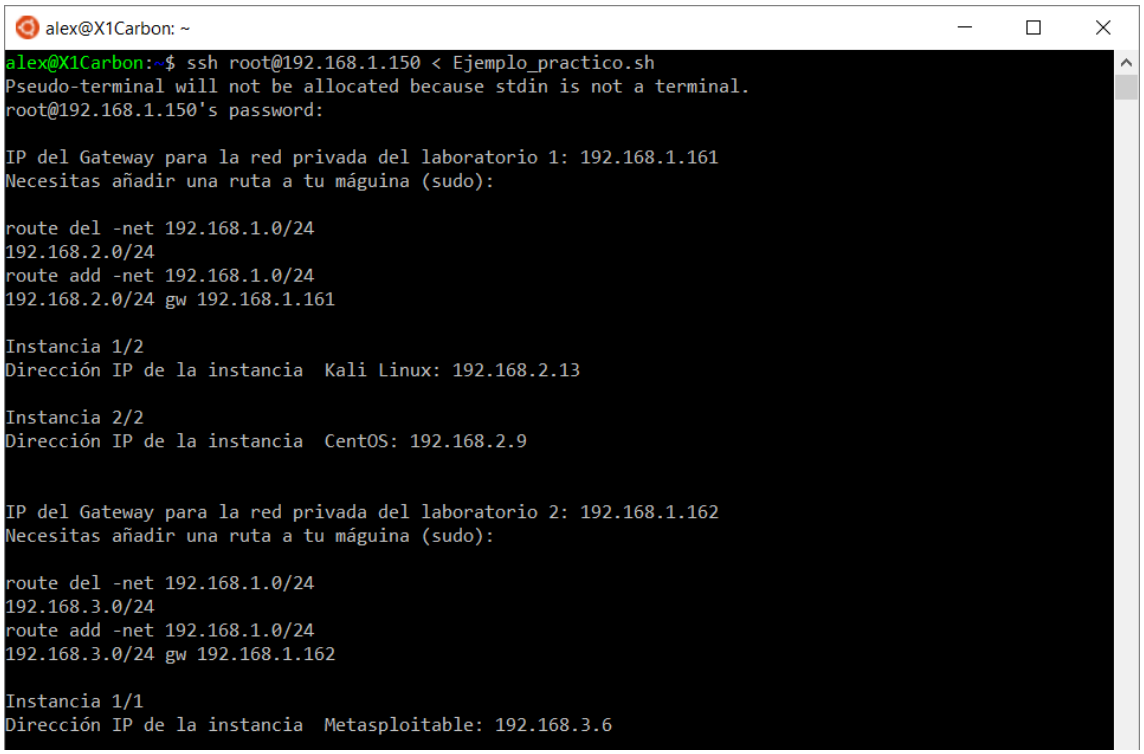


Figura 100 – [Ejemplo práctico] Lanzamiento del laboratorio virtual (Windows con WSL)

```
#  alex@ALEX-VAIO:~$ ssh root@192.168.1.150 < Ejemplo_practico.sh
 Pseudo-terminal will not be allocated because stdin is not a terminal.
 root@192.168.1.150's password:

 IP del Gateway para la red privada del laboratorio 1: 192.168.1.161
 Necesitas añadir una ruta a tu máquina (sudo):

 route del -net 192.168.1.0/24
 192.168.2.0/24
 route add -net 192.168.1.0/24
 192.168.2.0/24 gw 192.168.1.161

 Instancia 1/2
 Dirección IP de la instancia  Kali Linux: 192.168.2.13

 Instancia 2/2
 Dirección IP de la instancia  CentOS: 192.168.2.9


 IP del Gateway para la red privada del laboratorio 2: 192.168.1.162
 Necesitas añadir una ruta a tu máquina (sudo):

 route del -net 192.168.1.0/24
 192.168.3.0/24
 route add -net 192.168.1.0/24
 192.168.3.0/24 gw 192.168.1.162

 Instancia 1/1
 Dirección IP de la instancia  Metasploitable: 192.168.3.6
```

Lanzamiento desde el terminal de una máquina con Ubuntu

Conclusiones y líneas futuras

Para terminar, cabe destacar algunas ideas que resultan fundamentales para poner en valor este trabajo. Lo primero de todo hay que tener en cuenta el tipo de despliegue de nubes privadas que se facilita. OpenStack permite generar múltiples nubes privadas que son gestionadas mediante un sistema de administración de identidades centralizado (Keystone). La gestión individualizada de cada nube privada requiere dar acceso a cada propietario, lo que repercute en una degradación de la seguridad del sistema global, al obligar mantener múltiples roles de administración activos, todos sobre un único punto de fallo. La posibilidad de utilizar LDAP como sistema autenticador para cada nube permite descentralizar la gestión en sí de la nube, aumentando la superficie de ataque aislando físicamente los diferentes roles de administrador, y en definitiva, compartimentando los sistemas de acceso.

El control de acceso basado en LDAP es una alternativa muy interesante no solo por su sencillez, sino por la adaptación del mecanismo de acceso en función de las características de cada proyecto. Permite establecer desde sistemas simples de acceso por contraseña, como sistemas basados en dos o tres factores de autentificación, o la fortificación mediante el uso de certificados en uno o ambos extremos, pudiendo incluso implementar nuestra propia entidad certificadora.

Aprovechar la opción del multi-dominio, disponible pero no activada por defecto en OpenStack, permite el uso de diferentes “backends”, tal como se ha indicado anteriormente, pero también abre la posibilidad de establecer esquemas mucho más complejos para la fortificación del control de acceso, por ejemplo, definiendo diferentes estratos de servidores en función del nivel de seguridad requerido, o estableciendo configuraciones redundantes que permitan una rápida respuesta ante fallos.

El uso de LDAP permite establecer configuraciones de seguridad adaptadas a cada proyecto, pero además, dichas configuraciones pueden ser exportadas a diferentes proyectos de forma extremadamente fácil, mediante la simple importación y exportación de dichos esquemas, lo cual repercute directamente en la creación de backups que permitan restablecer automáticamente todos los sistemas.

Además, en este trabajo se ha demostrado cómo el uso de un servidor de autenticación independiente a OpenStack, permite llevar hasta los usuarios funciones y servicios que de otra forma solo se podrían hacer desde OpenStack. Es el caso del arranque de proyectos completos, que en este trabajo se han denominado laboratorios virtuales. Por defecto, OpenStack aunque permite a los usuarios de un Proyecto el arranque y ejecución de cada elemento, no permite la automatización de dicho proceso, salvo que se le diera a los usuarios específicamente el rol de administrador de OpenStack, para por ejemplo, acceder a la consola de comandos con privilegios suficientes para la ejecución de scripts. El servidor LDAP, una vez garantizada la identidad del usuario, puede gestionar directamente con OpenStack la ejecución de scripts en modo administrador, que automatizan todo el proceso de arranque y configuración de los citados Laboratorios Virtuales.

Sin duda LDAP es una tecnología con futuro en el ámbito de la autenticación en las nubes, tal como se ha demostrados en este trabajo. Sin embargo, quedan abiertas todavía bastantes líneas de desarrollo como, por ejemplo:

El esquema de autenticación propuesto es hasta cierto punto simplista, ya que no deja de ser un sistema básico de login más contraseña. Como ya se ha indicado anteriormente, LDAP

permite actuar como servidor de certificados, es decir, autoridad certificadora. Un primer desarrollo sería la implementación del sistema de autentificación basada en certificados, tanto para los usuarios, como para un sistema de certificación mutua, incluyendo a los diferentes servidores.

El proceso de automatización de proyectos debe ser refinado. El sistema propuesto hace uso de scripts que todavía deben ser ejecutados desde una ventana de comandos de OpenStack. Una mejora evidente es la posibilidad de realizar dichas operaciones de forma directa, a través de una aplicación de gestión, sin intervención directa del usuario, el cual solo accedería a una interfaz por ejemplo, tipo web, a través de la cual poder acceder directamente a los proyectos (Laboratorios Virtuales) una vez arrancados.

Por último, puesto que la aplicación directa de los denominados Laboratorios Virtuales es en la docencia, sería deseable el desarrollo de herramientas para la generación y gestión de dichos laboratorios virtuales. Estas herramientas incluirían interfaces sencillos para la creación de proyectos, la parametrización de los Laboratorios para que cada alumno genere un escenario relativamente distinto al de sus compañeros y la gestión en tiempo real de los proyectos abiertos por los usuarios.

Referencias

La mayor de la información utilizada en este trabajo procede de guías y artículos online, por lo que en esta bibliografía no aparecerán libros publicados sino (que los hay, pero no han sido de mucha ayuda) que aparecen enlaces a las páginas web con dicha información.

En muchas ocasiones no bastaba con seguir lo que se explicaba en una página web, y ha habido que utilizar el conocimiento conjunto que se obtenía entre múltiples referencias. También se han investigado muchas opciones que más tarde se han desechado, y no han sido utilizadas en el trabajo.

[1] G. Čandrlić, «GlobalDots,» 19 03 2013. [En línea]. Available: https://globaldots.com/cloud-computing-types-of-cloud7. [Último acceso: 30 09 2017].

[2] R. t. Docs, «Read the Docs,» [En línea]. Available: https://media.readthedocs.org/pdf/openstack-xenserver/latest/openstack-xenserver.pdf. [Último acceso: 30 09 2017].

[3] OpenStack, «Wiki - OpenStack,» [En línea]. Available: https://wiki.openstack.org/wiki/XenServer/XenAndXenServer. [Último acceso: 30 09 2017].

[4] OpenStack, «Docs - OpenStack,» [En línea]. Available: https://docs.openstack.org/devstack/latest/. [Último acceso: 30 09 2017].

[5] A.-X. s. c. h. j. a. t. M. L. J. m. B. J. G. R. A. s. y o. , «GitHub,» 27 10 2011. [En línea]. Available: https://github.com/citrix-openstack/devstack/blob/master/tools/xen/README.md. [Último acceso: 30 09 2017].

[6] Annie-XIE, «GitHub,» 17 01 2017. [En línea]. Available: https://github.com/citrix-openstack/qa/commit/d3e2fedb9564e05295fb449be52620ee0fb05e39. [Último acceso: 30 09 2017].

[7] R. Project, «RDO Project,» [En línea]. Available: https://www.rdoproject.org/networking/neutron-with-existing-external-network/. [Último acceso: 30 09 2017].

[8] S. World, «Server World,» 15 04 2015. [En línea]. Available: https://www.server-world.info/en/note?os=CentOS_7&p=openldap&f=1. [Último acceso: 30 09 2017].

[9] W. E. Quixote, «ServerFault,» 11 04 2015. [En línea]. Available: https://serverfault.com/questions/681259/how-to-properly-insert-a-set-of-olcaccess-attributes-to-the-configuration-of-an. [Último acceso: 30 09 2017].

[10] pico.dev, «Blog Bitix,» 21 02 2014. [En línea]. Available: https://picodotdev.github.io/blog-bitix/2014/02/generar-y-convertir-claves-y-certificados-con-openssl/. [Último acceso: 30 09 2017].

[11] OpenLDAP, «OpenLDAP,» [En línea]. Available: https://www.openldap.org/doc/admin21/tls.html. [Último acceso: 30 09 2017].

[12] S. World, «Server World,» 18 03 2015. [En línea]. Available: https://www.server-world.info/en/note?os=CentOS_7&p=openldap&f=4. [Último acceso: 30 09 2017].

[13] c. wang, «OpenLDAP,» 08 08 2003. [En línea]. Available: http://www.openldap.org/lists/openldap-software/200308/msg00254.html. [Último acceso: 30 09 2017].

[14] OpenLDAP, «OpenLDAP,» [En línea]. Available: https://www.openldap.org/doc/admin23/runningslapd.html. [Último acceso: 30 09 2017].

[15] HEIG-Cloud, «HEIG-Cloud,» 17 12 2015. [En línea]. Available: HEIG-Cloud. [Último acceso: 30 09 2017].

[16] R. Hat, «Red Hat,» [En línea]. Available: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/integrate_with_identity_service/sec-active-directory. [Último acceso: 30 09 2017].

[17] D. K. K. <dikonoor@xxxxxxxxxx>, «Launchpad,» 08 02 2017. [En línea]. Available: https://lists.launchpad.net/yahoo-eng-team/msg61334.html. [Último acceso: 30 09 2017].

[18] OpenStack, «Review - OpenStack,» [En línea]. Available: https://review.openstack.org/#/c/437402/4/keystone/identity/backends/ldap/core.py. [Último acceso: 30 09 2017].

[19] OpenStack, «Docs - OpenStack,» [En línea]. Available: https://docs.openstack.org/user-guide/cli-cheat-sheet.html. [Último acceso: 15 08 2017].

[20] G. Juszczak, «Tuxfixer,» 20 05 2016. [En línea]. Available: http://www.tuxfixer.com/download-kali-linux-64bit-openstack-kvm-qcow2-image/. [Último acceso: 30 09 2017].

[21] C. Schroder, «Linux.com,» 12 06 2014. [En línea]. Available: https://www.linux.com/learn/how-rescue-non-booting-grub-2-linux. [Último acceso: 30 09 2017].

[22] crtman, «Forum - kali.org,» 01 11 2016. [En línea]. Available: https://forums.kali.org/showthread.php?33190-Boot-stuck-with-message-quot-started-user-manager-with-uid-132-Dependency-failed-for-dev-quot. [Último acceso: 30 09 2017].

[23] Crash, «Forum - kali.org,» 26 08 2016. [En línea]. Available: https://forums.kali.org/showthread.php?32498-Delay-90-seconds-on-shutdown. [Último acceso: 30 09 2017].