ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

MECANISMOS PARA LA AUTOMATIZACION DE LABORATORIOS VIRTUALES EN INFRAESTRUCTURAS DE CLOUD OPENSTACK

(Mechanisms for the automation of virtual laboratories in cloud infrastructures OpenStack)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Jonatan Ruiz Alvarado

Abril - 2019

Índice

Tabla de ilustraciones

Resumen

La implantación de soluciones basadas en nubes (Cloud), están modificando muchos de los procesos productivos que conocemos hasta ahora. En el ámbito de la educación los beneficios que aporta la migración de sistemas físicos hacia sistemas virtuales repercuten directamente en la mejora de las clases prácticas mediante la elaboración de laboratorios virtuales, con prestaciones muy superiores a las de los laboratorios físicos existentes. La motivación de este proyecto es la existencia de una infraestructura de nube privada que quiere ser configurada para la provisión de laboratorios virtuales a utilizar dentro de las asignaturas impartidas por el Grupo de Telemática de la Universidad de Cantabria. En un TFG anterior, se comprobó la viabilidad de utilizar este tipo de tecnologías mediante la implementación de una prueba de concepto desarrollada en un PC. El objetivo de este trabajo es la definición de un sistema que pueda ser integrado en la infraestructura de Cloud existente, con el fin de permitir automatizar la creación y uso de laboratorios virtuales, en definitiva, la base para la definición de un sistema de gestión de laboratorios virtuales.

Abstract

The implementation of solutions based on clouds are modifying many of the production processes we know until now. In the field of education, the benefits of migrating from physical systems to virtual systems have a direct impact on the improvement of practical classes through the development of virtual laboratories, with features far superior to those of existing physical laboratories. The motivation of this project is the existence of a private cloud infrastructure that wants to be configured for the provision of virtual laboratories to be used within the subjects taught by the Telematics Group of the University of Cantabria. In a previous TFG, the viability of using this type of technology was verified through the implementation of a proof of concept developed on a PC. The objective of this work is the implementation of this system in the existing Cloud infrastructure, in order to specify different mechanisms that allow automating the creation and use of virtual laboratories.

1. Introducción y objetivos

El uso de Cloud ha ganado popularidad en los últimos tiempos, convirtiendo al Cloud Computing como la tendencia de mayor crecimiento en el mercado TI. Una de las razones fundamentales de este crecimiento es el ahorro de costes, ya que se reducen las inversiones tanto en software como en hardware, frente al uso de equipos físicos en entornos de trabajo tradicionales.

Si bien las aplicaciones de estos sistemas pueden circunscribirse a casi cualquier dominio, una de las grandes posibilidades de uso radica en el ámbito educativo. La posibilidad de crear laboratorios de redes y sistemas virtuales, en los que se emulan configuraciones físicas que pueden resultar inasumibles económicamente, permite acercar entornos de trabajo reales, sobre los cuales los alumnos pueden llevar a cabo ejercicios prácticos. Así, estos sistemas permiten replicar sistemas y situaciones que pueden ser accedidas por el alumno remotamente, haciendo más flexible el aprendizaje del mismo, incluso durante el desarrollo de las clases teóricas, pero especialmente, durante la fase de estudio y trabajo personalizado.

Dentro de los entornos de computación distribuida, OpenStack es uno de los software de código abierto más utilizados, ya que permite el despliegue de nubes privadas de una forma muy flexible, en el que el usuario puede desplegar sus propias aplicaciones, servicios o sistemas, mediante la definición de servidores y redes especializados, y todo a través de interfaces remotos amigables.

1.1 Motivación y objetivos

Este trabajo no es sino la evolución de los estudios y resultados obtenidos en dos desarrollos anteriores. En el primer trabajo, realizado en la finalización de su Máster por Martín Pereira Diéguez, y de título "Implementación de un entorno Cloud en las infraestructuras del Laboratorio de Aplicaciones Telemáticas", se evaluaba el despliegue de una infraestructura de Cloud basada en OpenStack. Básicamente el trabajo define tres soluciones para la implantación de OpenStack, de las cuales solo una, a la vista de los resultados obtenidos, fue implementada físicamente en los Laboratorios Docentes del Grupo de Ingeniería Telemática de la Universidad de Cantabria. Por otra parte, en el segundo Trabajo Fin de Máster, realizado por Alejandro Abascal Crespo, y titulado "Aplicación de LDAP como método de autenticación en entornos de Cloud Privada", aunque el objetivo principal del trabajo era el uso del protocolo LDAP (Lightweigth Directory Access Protocol) como método de autenticación en entornos Cloud OpenStack, planteó la posibilidad del uso de scripts mediante los cuales realizar la automatización en la creación de proyectos.

De acuerdo con los resultados de ambos trabajos, es necesario estudiar y comprobar el nivel de automatización que puede llevarse a cabo sobre los sistemas OpenStack desplegados. Para ello, se plantea un desarrollo portable del sistema OpenStack, basado en la versión PackStack, sobre el cual desarrollar el principal objetivo del trabajo: definición de un entorno de ejecución de comandos que permita automatizar el proceso de creación y arranque de laboratorios virtuales. Una vez arrancados, los proyectos generados deberán presentar todas las condiciones suficientes para que los alumnos puedan desarrollar sus prácticas docentes en base a los enunciados establecidos.

Como objetivo secundario, y de acuerdo con las observaciones realizadas en uno de los trabajos de referencia, según las cuales la automatización requiere la ejecución directa sobre ventanas de comandos, se pretende analizar dicha restricción y buscar alguna alternativa basada en servicios web.

1.2 Organización del documento

Este trabajo se compone de cinco apartados que se han estructurado de la siguiente forma:

Tras esta primera parte de introducción, en el capítulo 2 se habla sobre los conceptos teóricos necesarios para la realización del trabajo. Se exponen de una manera breve, centrándose en la indicación de sus características principales.

En el capítulo 3 se desarrolla el despliegue del sistema portable, basado en un servidor OpenStack sobre sistema operativo CentOS. Se incluyen detalles sobre el proceso de instalación y configuración PackStack, así como una explicación de los principales comandos de OpenStack necesarios para la creación y automatización del laboratorio virtual.

En el capítulo 4 se exponen, además de los motivos para la creación de un servicio de automatización, su creación mediante el uso de Apache y PHP, en forma de servicio web.

Para finalizar, en el capítulo 5 se hace un balance del trabajo y se aportan posibles ideas para su mejora.

2. Conceptos teóricos

En este apartado se va a proceder a explicar los principios de varias tecnologías usadas durante el desarrollo de este trabajo.

2.1 Computación en la nube

La Nube o Cloud se puede definir como una red de ordenadores que permite ofrecer servicios de computación a los clientes. Para ello se ofrece el acceso a una serie de recursos virtuales, los cuales no tienen por qué estar ligados a una sola máquina física. Esto permite que, dependiendo de la necesidad de un servicio, se le darán o quitaran recursos dinámicamente para satisfacer sus necesidades específicas en cada momento.

El tipo de nube se clasifica en función de diferentes parámetros. Así, por ejemplo, dependiendo del modelo de despliegue, existen hasta cuatro tipos de nube:

- Publica: Su infraestructura está abierta al público en general. Su proveedor es una empresa que ofrece sus recursos a cambio de un pago. Ejemplos de estas empresas son Amazon Web Service, Google App Engine, Azure, etc.
- Privada: La infraestructura solo sirve a una organización, la cual puede estar o no gestionada por la propia organización, al igual que su alojamiento, que puede estar en instalaciones de terceros.
- Comunitaria: La infraestructura es compartida por un conjunto de organizaciones.
- Hibrida: Combinación de una cloud publica y una privada. Es decir, ciertas funcionalidades del cloud son privadas y otras son accesibles por el público en general.

Otra clasificación atiende al modelo de servicio ofertado:

- Software as a Service (SaaS): Los consumidores tienen acceso a las aplicaciones web que ofrece la empresa, como pueden ser el web mail. Estas aplicaciones son accesibles a través de un navegador web, eliminando la necesidad de instalar software adicional en el equipo de los clientes.
- Platform as a Service (PaaS): El proveedor ofrece a sus clientes un entorno de desarrollo junto a una serie de herramientas y servicios para el desarrollo de

sus aplicaciones. Los consumidores crean sus propias aplicaciones y se las ofrecen a terceros.

- Infraestructure as a Service (IaaS): El proveedor ofrece sus servidores, donde el consumidor es capaz de desplegar y ejecutar software. El consumidor tiene control sobre los sistemas operativos, el almacenamiento y las aplicaciones implementadas, y control limitado de determinados componentes de red, como puede ser el firewall de host.

En la figura 1 se pueden ver los diferentes modelos de servicio, indicándose a qué elementos tienen acceso los consumidores, y cuáles el proveedor de servicio.

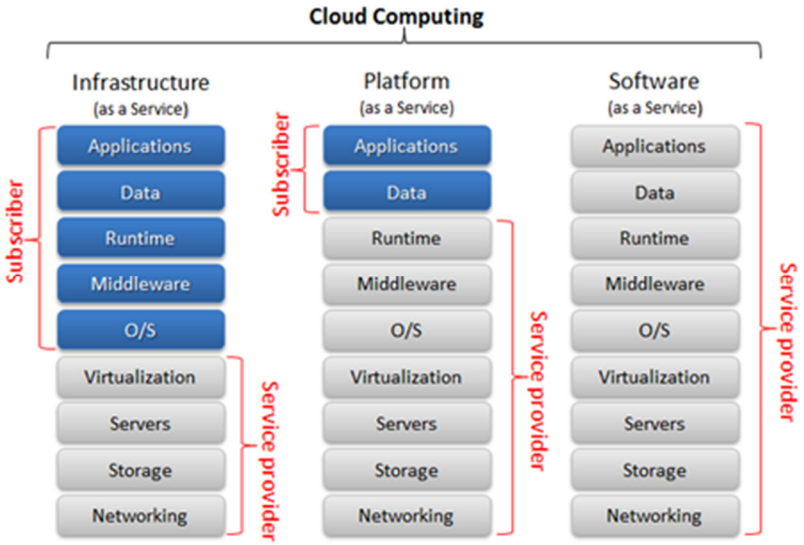


Figura 1- Modelo de servicio

2.1.1Virtualización

La virtualización es el mecanismo mediante el cual se crea una versión virtual de un dispositivo o recurso, como por ejemplo un servidor, un dispositivo de almacenamiento, una red o un sistema operativo. Para que un equipo físico pueda comportarse como varios equipos independientes, las características de cada hardware físico deben recrearse mediante el uso de software especializado, normalmente denominado Hypervisor.

Los objetivos fundamentales de la virtualización son:

- Proteger contra fallos: Las máquinas virtuales no están ligadas a un servidor en específico, es decir que si un servidor falla las máquinas virtuales localizadas en dicho servidor pueden migrar a otro.

- Aumentar la eficiencia del uso de recursos: Al permitir que un servidor físico pueda ejecutar más de una instancia al mismo tiempo, hace que los recursos hardware se aprovechen mejor.
- Mejora la flexibilidad: Las máquinas virtuales se pueden configurar fácilmente gracias a software de gestión ahorrando tiempo y por lo tanto dinero a la empresa.
- Mejora de la seguridad: Las máquinas virtuales son independientes entre sí, por lo que si una de ellas tiene un virus el resto de las máquinas no se verán afectaras.

Por su parte, el hipervisor es encargado de implementar una capa de adaptación entre el hardware físico del host y el sistema operativo de la máquina virtual. A su vez, esta capa software proporciona un sistema de comunicación específico entre el hardware virtual y el resto del software, posibilitando así que estos se comuniquen con el hardware real. Un equipo donde un hipervisor ejecuta una o más máquinas virtuales se denomina equipo host o anfitrión, y a cada máquina virtual se la denomina equipo invitado. El hipervisor gestiona así la ejecución de los diferentes sistemas operativos huéspedes. De esta forma, las máquinas virtuales pueden tener diversidad de sistemas operativos compartiendo los recursos de hardware virtualizados, es decir, pueden existir instancias ejecutándose a la vez de Linux, Windows y macOS. sobre una sola máquina física.

A su vez, los hipervisores se pueden clasificar en dos tipos:

- Hipervisores de tipo 1: También conocidos como bare-metal, el hipervisor se ejecuta directamente sobre el hardware físico. Esto hace que el hipervisor sea independiente del sistema operativo. Su tarea principal es gestionar los recursos hardware entre los diferentes sistemas operativos. Cualquier problema en una de las máquinas virtuales no afecta al resto de las máquinas. Ejemplos de hipervisores de tipo 1 son Microsoft Hyper-V, Citrix Xen Server, VMWare ESXi-Server. En la figura 3 se puede apreciar la arquitectura del hipervisor de tipo 1.

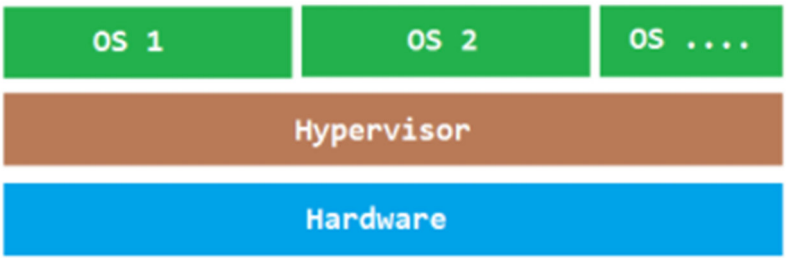


Figura 2 - Arquitectura de Hipervisor tipo 1

- Hipervisores de tipo 2: También llamados hosted, el hipervisor se ejecuta sobre el sistema operativo, así que al contrario que el tipo 1 este es completamente dependiente del sistema operativo sobre el que se ejecuta. Ejemplos de hipervisores de tipo 2 son VirtualBox y VMware. En la figura número 4 se puede apreciar la arquitectura del hipervisor de tipo 2.

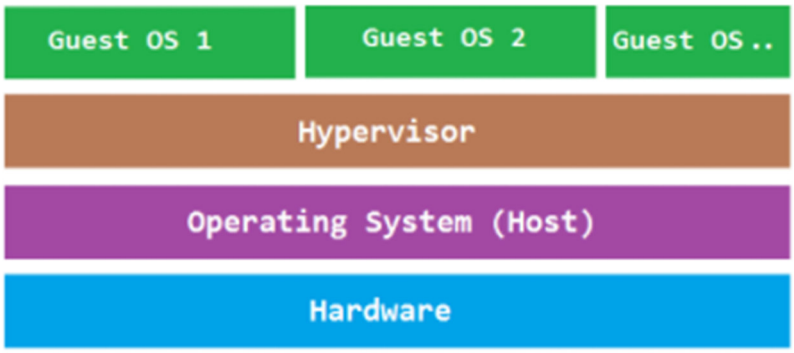


Figura 3 - Arquitectura de Hipervisor tipo 2

2.1.2 Sistema Operativo de Nube: OpenStack

Un sistema operativo de nube es un tipo de sistema operativo diseñado para operar en entornos de computación en nube y virtualización. Un sistema operativo en la nube gestiona el funcionamiento, la ejecución y los procesos de las máquinas virtuales, los servidores virtuales y la infraestructura virtual, así como los recursos de hardware y software de back-end.

Un sistema operativo en la nube gestiona principalmente la operación de una o más máquinas virtuales dentro de un entorno virtualizado. Dependiendo del entorno virtual y los servicios en la nube en uso, la funcionalidad de los sistemas operativos en la nube varía.

Por ejemplo, un sistema operativo en la nube desarrollado para ser utilizado dentro de un entorno informático específico administrará los procesos y subprocesos de una sola máquina o de una agrupación de máquinas virtuales y servidores. De manera similar, un sistema operativo en la nube liviano podría proporcionar a los usuarios finales aplicaciones y servicios preinstalados, a los que se accede a través de un navegador de Internet.

Entre las soluciones comerciales existentes cabe destacar Microsoft Windows Azure y Google Chrome OS, mientras que entre las soluciones Opensource la más utilizada es OpenStack.

OpenStack es un sistema operativo en nube que controla grandes grupos de recursos de computación, almacenamiento y redes a través de un centro de datos, todo ello gestionado a través de un panel de control que proporciona a los administradores el control al tiempo que permite a sus usuarios aprovisionar recursos a través de una interfaz web. Desde el punto de vista del despliegue, Openstack ofrece un modelo de "infraestructura como servicio" (IaaS), por medio del cual se ponen a disposición de los clientes servidores y redes virtuales.

OpenStack está a su vez compuesto por una serie de proyectos de software libre, de los cuales ahora se indican únicamente los elementos utilizados en este. En concreto, los proyectos utilizados son:

- Keystone: Proporciona autenticación de cliente de acceso a la API (Application Program Interface), además de la detección de servicios de identidad. Soporta protocolos relacoinados con dicha tarea, como son LDAP, OAuth, OpenID Connect, SAML y SQL.
- Neutron: Proporciona conectividad de red como servicio a otros proyectos de OpenStack, como por ejemplo Nova, el cual utiliza la API de Neutron para solicitar la conexión de las máquinas virtuales a la red determinada. Permite la creación de redes, subredes, enrutadores, cortafuegos, y redes privadas virtuales. Cuenta con una arquitectura modular que se integra con múltiples tecnologías de proveedores de red externos.
- Glance: Almacena y gestiona imágenes de los discos de las máquinas virtuales.

- Nova: Proporciona una forma de aprovisionar servidores virtuales. Nova soporta la creación de máquinas virtuales y servidores baremetal. Nova se ejecuta como un conjunto de daemons encima de los servidores Linux existentes para proporcionar ese servicio. Para el uso de nova es necesario que los tres servicios anteriores estén previamente activos:
- Horizon: Es la implementación del panel de control de OpenStack, que es extensible y proporciona una interfaz de usuario basada en web para los servicios de OpenStack.
- Cinder: Es un servicio de almacenamiento en bloque para OpenStack.. Virtualiza la gestión de dispositivos de almacenamiento y proporciona a los usuarios finales una API de autoservicio para solicitar y consumir esos recursos, sin necesidad de saber dónde se implementa realmente su almacenamiento o en qué tipo de dispositivo se lleva a cabo.

Con tal cantidad de componentes, el proceso de instalación de OpenStack puede ser bastante complejo, por lo que existe, además de abundante información técnica, multitud de tutoriales y ejemplos de configuración de diferentes soluciones basadas en OpenStack, así como implementaciones desarrolladas a medida por algunas empresas, y que son ofrecidas a la comunidad añadiendo ofertas de mantenimiento. Por otro lado, la complejidad del proceso también ha hecho que aparezcan algunas soluciones que permiten la elaboración de "Pruebas de Concepto", es decir, instalaciones simplificadas de OpenStack con ciertas limitaciones, pero suficientes para comprobar el funcionamiento de la nube. Ese es el caso de DevStack. o del sistema utilizado en este trabajo, denominado PackStack, y que ya fue utilizado anteriormente en los trabajos de referencia.

2.2 Puppet

Puppet es una herramienta de gestión de configuración open-source utilizada por los administradores de sistemas y los ingenieros para construir y configurar infraestructuras complejas sobre la marcha. Es una herramienta desarrollada con Ruby que incluye el acceso a máquinas físicas o virtuales. Puppet es de hecho una de las tecnologías utilizadas por PackStack para la instalación de OpenStack.

Puppet sigue una arquitectura maestro-esclavo. El maestro contiene las funciones de configuración y el esclavo es el nodo remoto sobre el que se van a aplicar los procesos de configuración. El esclavo le envía al maestro información, como puede ser la dirección IP, sistema operativo, si es una máquina virtual, etc. El maestro utiliza esa información para compilar un catálogo que define cómo debe configurarse el Esclavo. Así, el catálogo es el documento que describe cómo quiere que sea configurado cada recurso, gestionado por Puppet, en un Esclavo. Para finalizar, el esclavo informa al maestro que la configuración está completa. El intercambio de información entre el maestro y el esclavo se realiza de forma segura mediante el uso de SSL. Todo este proceso se corresponde con el de la figura 4.

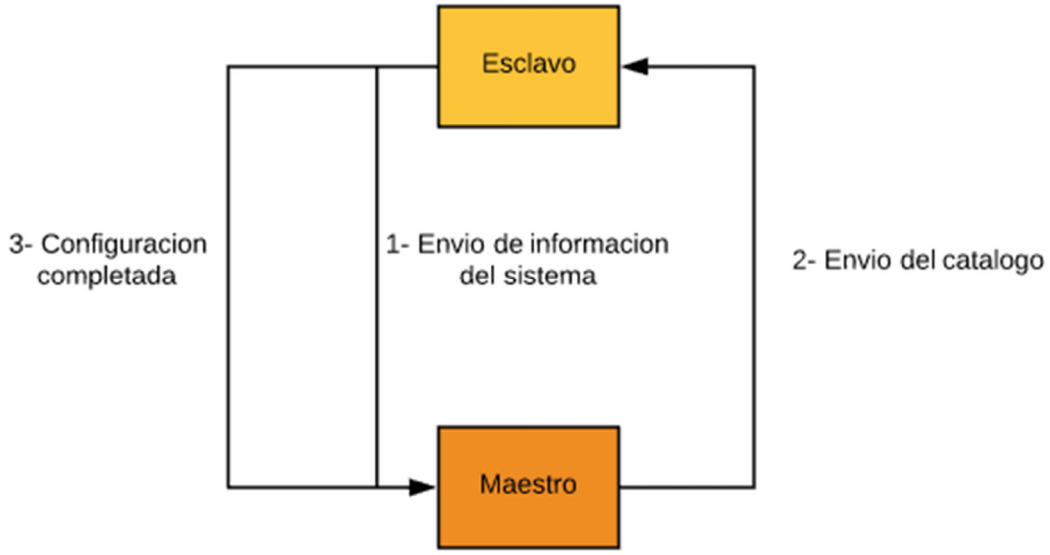


Figura 4 - Intercambio de mensajes en Puppet

Puppet utiliza un lenguaje declarativo propio para describir la configuración del sistema, que puede aplicarse directamente, o compilarse en un catálogo, y distribuirse al sistema de destino a través del cliente-servidor. El agente, para ello, utiliza proveedores específicos del sistema, haciendo cumplir los requerimientos del recurso especificado en los manifiestos. La capa de recursos permite a los administradores describir la configuración en términos de alto nivel, como usuarios, servicios y paquetes, sin necesidad de especificar comandos específicos del sistema operativo (como rpm, yum, apt).

En concreto, Packstack es una utilidad que hace uso de módulos Puppet para implementar varias partes de OpenStack en múltiples servidores preinstalados, sobre SSH

automáticamente. Actualmente solo se admite su instalación sobre sistemas operativos anfitriones CentOS, Red Hat Enterprise Linux (RHEL) y derivados compatibles de ambos.

2.3 Apache y PHP

Dentro de los objetivos de este trabajo se incluye el facilitar al usuario una interfaz web para el despliegue de los laboratorios virtuales. Para ello es necesario hacer uso de un servidor Web, como es Apache, y de las facilidades de programación asociadas, como es el caso de PHP.

El proyecto Apache HTTP Server, llamado comúnmente Apache, es un servidor web multiplataforma de código abierto, lo que significa que puede ser usado por sistemas operativos como UNIX, Windows y Macintosh. Apache es desarrollado y mantenido por la comunidad bajo la supervisión de la Apache Software Foundation.

Apache es actualmente un servidor Web seguro y eficiente, que provee servicios HTTP sobre los estándares más actuales. Sus características pueden ser extendidas mediante la instalación de módulos, los cuales pueden ser muy diversos, aunque destaca el uso de lenguajes de programación en del lado del servidor, como es el caso de PHP, y las implementaciones de SSL y TLS como mecanismos fundamentales de securización..
Por su parte, PHP (Personal Home Page) es un lenguaje de programación del lado del servidor diseñado para el desarrollo web, aunque también se puede utilizar como un lenguaje de programación de propósito general. Su principal característica reside en que el código PHP es procesado por un intérprete implementado directamente en el servidor web. El código PHP puede ser incrustado en el código HTML o puede ejecutarse sobre la línea de comandos y puede utilizarse para implementar aplicaciones independientes.

Haciendo uso de ambos elementos en este trabajo se propone una simple interfaz de usuario que permite simplificar el proceso de creación y arranque del laboratorio virtual a realizar por los alumnos,

3. Instalación y configuración de OpenStack

De acuerdo con el Trabajo de Fin de Máster de Alejandro Abascal Crespo, OpenStack puede ser desplegado de tres maneras distintas:

- XenServer y OpenStack usando DevStack,
- XenServer y OpenStack usando Mirantis OpenStack y
- CentOS y OpenStack usando PackStack.

Esta última forma de instalación es en la que se obtuvieron mejores resultados, por lo que se ha tomado como referencia en este trabajo.

PackStack, como se dijo anteriormente, está formado por un conjunto de scripts, que mediante el uso de Puppet facilita la instalación de OpenStack. PackStack puede generar un fichero de configuración en el cual es posible modificar todas las opciones de OpenStack como por ejemplo, instalar o no ciertos módulos, la configuración de la red, cambiar contraseñas de administrador, e incluso realizar el despliegue en un único equipo físico, como es el caso de este trabajo. Es fundamental tener en cuenta que PackStack solo está disponible para las distribuciones de Red Hat y CentOS, por lo que es este último el que se instala como sistema operativo anfitrión, proceso que el anexo 1 detalla, tanto la instalación como configuración previa de CentOS 7.

3.1 Despliegue de OpenStack

Como se ha comentado anteriormente para el despliegue de OpenStack se va a utilizar PackStack, lo que permite una instalación más sencilla y rápida comparado con un despliegue de OpenStack convencional y de su predecesor DevStack.

Para empezar desde una terminal y teniendo privilegios de administrador se recomienda ejecutar las siguientes instrucciones.

```
# systemctl disable firewalld
# systemctl stop firewalld
# systemctl disable NetworkManager
# systemctl stop NetworkManager
# systemctl enable network
# systemctl start network
```

Se debe desactivar el Network Manager y firewall que tiene incluido CentOS por defecto, ya que durante el despliegue de OpenStack pueden provocar alguna excepción que provoque que algún paso falle. Además, una vez instalado OpenStack, las funciones de ambos sistemas van a recaer directamente sobre Neutron.

A continuación, es necesario actualizar el sistema e instalar los paquetes de OpenStack para CentOS y PackStack.

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

El proceso de instalación de OpenStack, si bien no es lo más recomendable, puede realizarse de una forma sencilla con todos nodos en un único equipo utilizando el comando:

```
# packstack –allione
```

El principal problema que conlleva el uso de este comando tal cual, es que va a instalar más módulos de los estrictamente necesarios para el trabajo requerido y, debido a las limitaciones en hardware del host, no es precisamente lo más adecuado. Además, debido a que el host es una máquina de VirtualBox con dos interfaces de red, todos los nodos serían instalado sobre una sola, teniendo que ir posteriormente a los ficheros de configuración de los nodos para modificarlos a la interfaz correcta, para que OpenStack funcione sin problemas.

Así todo, la forma más correcta para tener más controlado que módulos se van a instalar, cuales son, la configuración de la red, etc., es usar la opción de la creación de un fichero de configuración, generado automáticamente por PackStack. Para obtener dicho fichero se ejecuta el siguiente comando.

```
# packstack --gen-answer-file=“nombre del fichero de configuracion”.txt
```

Tras generar el fichero de configuración de PackStack, es necesario modificar ciertos parámetros. Debido a las limitaciones de hardware, no instalaremos componentes superficiales, para tratar de conseguir la mayor fluidez posible. Como recomendación,

también es necesario modificar la contraseña por defecto del usuario administrador del Keystone y, opcionalmente, no crear un usuario y proyecto “demo”.

Para la configuración de la red es necesario confirmar las direcciones IP de los diferentes nodos (Compute, Control, Storage, etc.). En este caso, todos deben ser instalados sobre la dirección de la interfaz “solo anfitrión”, que en Centos suele corresponderse con la interfaz “enp0s8” con dirección 192.168.100.100. Además, es necesario configurar la red externa, que se asigna a la NAT (la interfaz “enp0s3”). De esta manera se permite dar acceso a internet a las máquinas virtuales de los proyectos creados en OpenStack.

Para confirmar los nombres reales de nuestras interfaces y sus correspondientes direcciones se hace uso de:

```
# ip addr
```

Para editar el fichero de configuración, con un editor de textos como puede ser “vi” o “nano”, modificamos los siguientes apartados (marcados en negrita).

```
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). Note this
44  # will also automatically install gnocchi service and configures it as
45  # the metrics backend. ['y', 'n']
46  CONFIG_CEILOMETER_INSTALL=n
47
48  # Specify 'y' to install OpenStack Telemetry Alarming (Aodh). Note
49  # Aodh requires Ceilometer to be installed as well. ['y', 'n']
50  CONFIG_AODH_INSTALL=n
                      . . .

92  # Server on which to install OpenStack services specific to the
93  # controller role (for example, API servers or dashboard).
94  CONFIG_CONTROLLER_HOST=192.168.100.100
95
96  # List the servers on which to install the Compute service.
97  CONFIG_COMPUTE_HOSTS=192.168.100.100
98
99  # List of servers on which to install the network service such as
```

```
100  # Compute networking (nova network) or OpenStack Networking (neutron).
101  CONFIG_NETWORK_HOSTS=192.168.100.100
                        . . .

322  # User name for the Identity service 'admin' user.  Defaults to
323  # 'admin'.
324  CONFIG_KEYSTONE_ADMIN_USERNAME=admin
325
326  # Password to use for the Identity service 'admin' user.
327  CONFIG_KEYSTONE_ADMIN_PW=telematica
                        . . .

870  # Comma-separated list of colon-separated Open vSwitch
871  # <bridge>:<interface> pairs. The interface will be added to the
872  # associated bridge. If you desire the bridge to be persistent a value
873  # must be added to this directive, also
874  # CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS must be set in order to create
875  # the proper port. This can be achieved from the command line by
876  # issuing the following command: packstack --allinone --os-neutron-
877  # ovs-bridge-mappings=ext-net:br-ex --os-neutron-ovs-bridge-interfaces
878  # =br-ex:eth0
879  CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:enp0s3
                        . . .

1190  # Specify 'y' to provision for demo usage and testing. ['y', 'n']
1191  CONFIG_PROVISION_DEMO=n
```

Acabada la modificación del fichero de configuración, se inicia la instalación de OpenStack usando PackStack.

```
# packstack --answer-file=“nombre del fichero de configuracion”.txt
Proceso de instalación

Welcome to the Packstack setup utility

The installation log file is available at: /var/tmp/packstack/20180822-
204321-KLTXj0/openstacksetup.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 192.168.100.100_controller.pp
192.168.100.100_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.
* File /root/keystonerc_admin has been created on OpenStack client host
192.168.100.100. To use the command line tools you need to source the file.
* To access the OpenStack Dashboard browse to
http://192.168.100.100/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/20180822-
204321-KLTXj0/openstack-setup.log
* The generated manifests are available at: /var/tmp/packstack/20180822-
204321-KLTXj0/manifests
```

Este es el mensaje que aparece cuando OpenStack ha acabado la instalación de manera satisfactoria. La instalación habrá generado un archivo llamado “keystonerc_admin” el cual contiene las credenciales del administrador.

```
unset OS_SERVICE_TOKEN
    export OS_USERNAME=admin
    export OS_PASSWORD=’telematica’
    export OS_AUTH_URL=http://192.168.100.100:5000/v3
    export PS1='[\u@\h \W(keystone_admin)]\$ '

export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_IDENTITY_API_VERSION=3
```

En OpenStack, aparte del clásico usuario y contraseña, se necesita que el usuario sea un miembro de algún proyecto para poder realizar cualquier acción. El fichero anterior facilita el acceso através de las consolas de los diferentes servidores que componen PackStack.

3.2 Configuración OpenStack

Una vez se ha desplegado OpenStack correctamente, es necesario configurar ciertos parámetros del mismo. En este trabajo se va a crear y configurar la red externa y añadir las imágenes de los diferentes sistemas operativos a utilizar.

Para ejecutar los comandos de OpenStack en necesario cargar las credenciales de OpenStack, para ello basta escribir el siguiente comando.

```
# . keystonerc_admin
```

Para configurar la red externa, se crea una red con los siguientes parámetros:

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

# openstack subnet create --network external_network --allocation-pool
start=10.0.2.100,end=10.0.2.199 --dns-nameserver 8.8.8.8 --dns-nameserver
8.8.4.4 --gateway 10.0.2.2 --subnet-range 10.0.2.0/24 external_network

Resultado

+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 10.0.2.100-10.0.2.199                |
| cidr              | 10.0.2.0/24                          |
| created_at        | 2018-08-22T23:31:55Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.4.4, 8.8.8.8                     |
| enable_dhcp       | True                                 |
| gateway_ip        | 10.0.2.2                             |
| host_routes       |                                      |
| id                | 7f66687f-dc9d-4516-9ece-d82de418f55c |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | external_network                     |
| network_id        | 69c399c8-62d0-4cf8-b450-dflab07d6f3d |
| project_id        | 6312db7b1f1541f3a2a6b8e294bbcab6     |
| revision_number   | 2                                    |
| segment_id        | None                                 |
| service_types     |                                      |
| subnetpool_id     | None                                 |
| updated_at        | 2018-08-22T23:31:55Z                 |
+-------------------+--------------------------------------+
```

- La opción *"--share*" permite que todos los proyectos puedan usar dicha red.
- La opción *"--external*" indica que la red es externa. Para los apartados siguientes vamos a usar la opción *"--internal*" que indica que la red es interna.
- Las opciones *"--provider-physical-network extnet*" y *"--provider-network-type flat*" le indicamos que conectamos la red nativa con la red virtual de OpenStack.

Se puede comprobar la correcta creación de la red accediendo desde un navegador al OpenStack Dashboard. Para ello es necesario utilizar la dirección del nodo Controller que, al estar montado en el mismo Host, seria "*http://10.0.2.50/dashboard*".

Una vez identificados, se accede a "Proyecto > Red > Topología de red", que debería mostrar algo similar a la figura 5 .

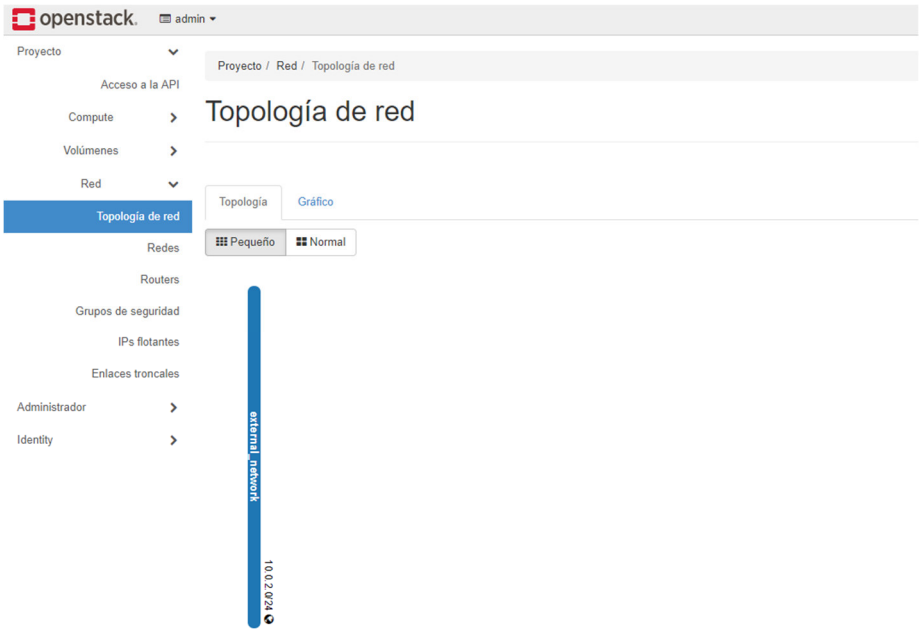


Figura 5 - Configuración red externa OpenStack

Según se muestra, las máquinas virtuales de los proyectos podrán acceder a internet a través de la red externa generada anteriormente.

Falta añadir las imágenes de los diferentes sistemas que son necesarias en los siguientes apartados, que son CirrOS, Kali Linux y Metasploitable.

- **CirrOS** es un sistema operativo ligero que suele ser utilizada para realizar pruebas y comprobar que todo funciona correctamente, ya que su despliegue es prácticamente inmediato.
- **Kali Linux** es una distribución de Linux orientada a realizar pruebas de penetración avanzada y auditorías de seguridad. Kali contiene cientos de herramientas para realizar diversas tareas de seguridad de la información, tales como pruebas de penetración, investigación de seguridad e ingeniería inversa.
- **Metasploitable** es una versión intencionalmente no segura de Ubuntu Linux que suele usarse para probar las herramientas incluidas en Kali con el fin de mostrar vulnerabilidades en el sistema.

Para añadirlas es necesario estar identificado como administrador. Se recomienda realizar todo el proceso a través de los clientes de la ventana de comandos, aunque sería posible hacer lo mismo desde el Dashboard.

```
# curl http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
| openstack image create "cirros" --disk-format qcow2 --container-format
bare –public


Resultado
+------------------+------------------------------------------------+
| Field            | Value                                          |
+------------------+------------------------------------------------+
| checksum         | 443b7623e27ecf03dc9e01ee93f67afe               |
| container_format | bare                                           |
| created_at       | 2018-08-22T23:34:14Z                           |
| disk_format      | qcow2                                          |
| file             | /v2/images/f546fbaf-ebd5-449b-a714-4628…/file |
| id               | f546fbaf-ebd5-449b-a714-46280a7fedd2           |
| min_disk         | 0                                              |
| min_ram          | 0                                              |
| name             | cirros                                         |
| owner            | 63a2db7b1f1541f3a2a6b8e294bbcab6               |
| protected        | False                                          |
| schema           | /v2/schemas/image                              |
| size             | 12716032                                       |
| status           | active                                         |
| tags             |                                                |
| updated_at       | 2018-08-22T23:34:14Z                           |
| virtual_size     | None                                           |
| visibility       | public                                         |
+------------------+------------------------------------------------+
```

Los enlaces para las descargas de Kali Linux y Metasploitable son:

```
Kali Linux
https://www.kali.org/downloads/
Metasploitable
https://sourceforge.net/projects/metasploitable/
```

Para añadirlas desde el dashboard se accede a “Administrador > Compute > Imágenes > Crear imagen”. Aparecerá una ventana como la figura 6 que es necesario completar.

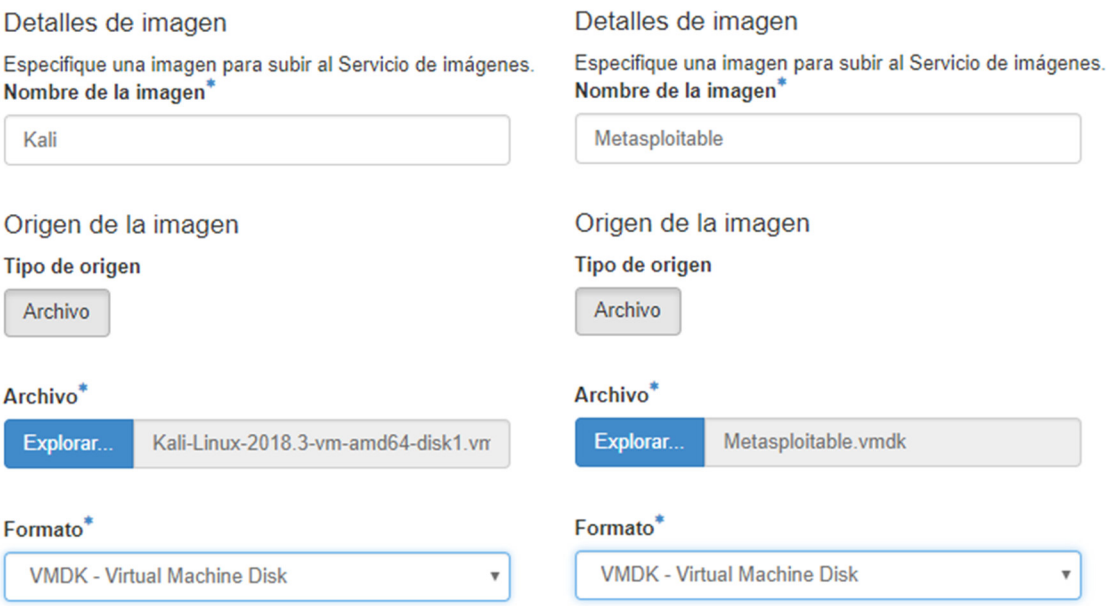


Figura 6 - Insertar imágenes en OpenStack

Para comprobar que las imágenes se han cargado correctamente desde el dashboard, se accede a "Administrador > Compute > Imágenes", y mostrará algo similar a la Figura 7.

Propietario	Nombre	Tipo	Estado	Visibilidad	Protegido	Formato de disco	Tamaño
admin	cirros	Imagen	Activo	Público	No	QCOW2	12.13 MB
admin	Kali	Imagen	Activo	Público	No	VMDK	3.82 GB
admin	Metasploitable	Imagen	Activo	Público	No	VMDK	1.81 GB

Figura 7 - Imágenes en OpenStack

3.3 Creación del laboratorio virtual

El principal motivo para la creación de estos laboratorios virtuales es su aplicación en el ámbito educativo. Para ello en este trabajo se propone automatizar la creación del entorno de trabajo en el que los alumnos deben desarrollar una práctica concreta, en este caso, relacionadas con las temáticas que abordan las diferentes asignaturas de la mención de telemática, como es el caso de la seguridad en redes, tomada como ejemplo de aplicación.

En este aparatado se explicará paso por paso la creación de un laboratorio virtual tipo para la realización de pruebas de detección de vulnerabilidades, de forma completamente aislada a la red Internet. Este laboratorio virtual se compone de dos redes independientes, sobre las que se corren sendas máquinas virtuales. Estas redes están conectadas con sendos routers, que a su vez se conectan con la red externa dando como resultado un esquema de red como el de la figura 8 (Importante: la conexión con la red externa permite poner en comunicación ambos routers, y da acceso de salida hacia Internet si fuera necesario).

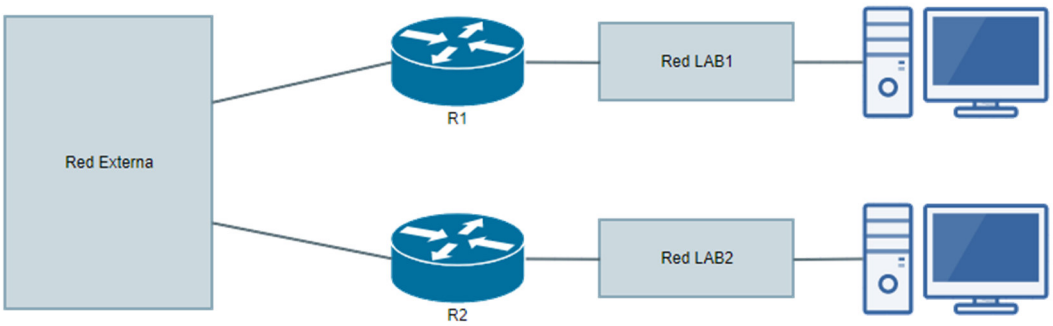


Figura 8 - Topología de un laboratorio

Con las credenciales de administrador de OpenStack que se encuentran en el archivo "keystonerc_admin" se crean los routers.

```
# openstack router create R1
# openstack router create R2
```

Tras la ejecución de estos comandos en la pestaña de "red > topología de red" debería aparece la figura 9.

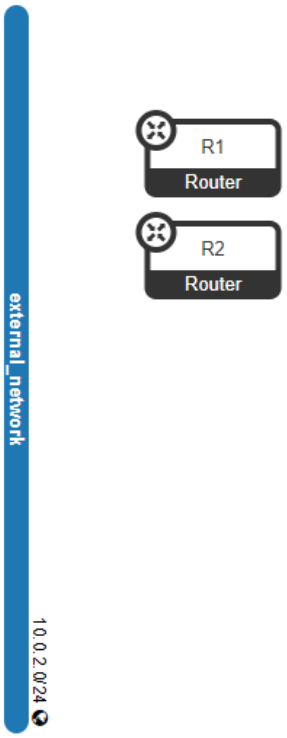


Figura 9 - Creación de routers en OpenStack

Lo siguiente es crear las dos redes de los laboratorios.

```
# openstack network create --internal LAB1

# openstack subnet create --network LAB1 --allocation-pool
start=10.10.10.5,end=10.10.10.254 --dns-nameserver 8.8.8.8 --dns-nameserver
8.8.4.4 --gateway 10.10.10.1 --subnet-range 10.10.10.0/24 subnet_LAB1

# openstack network create --internal LAB2

# openstack subnet create --network LAB2 --allocation-pool
start=20.20.20.5,end=20.20.20.254 --dns-nameserver 8.8.8.8 --dns-nameserver
8.8.4.4 --gateway 20.20.20.1 --subnet-range 20.20.20.0/24 subnet_LAB2
```

Otra vez en la pestaña “red > topología de red” aparecerá la figura 10.

Figura 10 - Creación de redes en OpenStack

Es necesario conectar los routers con las redes.

```
Con la red externa
# openstack router set --external-gateway external_network R1
# openstack router set --external-gateway external_network R2
Con los laboratorios
# openstack router add subnet R1 subnet_LAB1
# openstack router add subnet R2 subnet_LAB2
```

La figura 11 muestra el resultado de las conexiones. Los valores de las direcciones asignadas pueden variar, ya que OpenStack las asigna aleatoriamente dentro del pool de direcciones asignadas a dicha red.

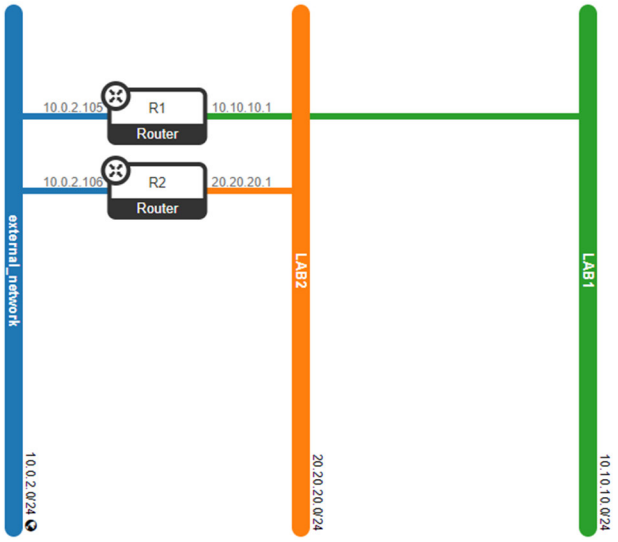


Figura 11 - Conectar redes en OpenStack

Para crear las máquinas virtuales.

```
# openstack server create --flavor m1.tiny --image cirros --nic net-
id=LAB1,v4-fixed-ip=10.10.10.10 --security-group default cirros1

# openstack server create --flavor m1.tiny --image cirros --nic net-
id=LAB2,v4-fixed-ip=20.20.20.20 --security-group default cirros2
```

Con esto ya estaría funcionando un primer laboratorio virtual, que da como resultado la topología de red que se muestra en la figura 12.

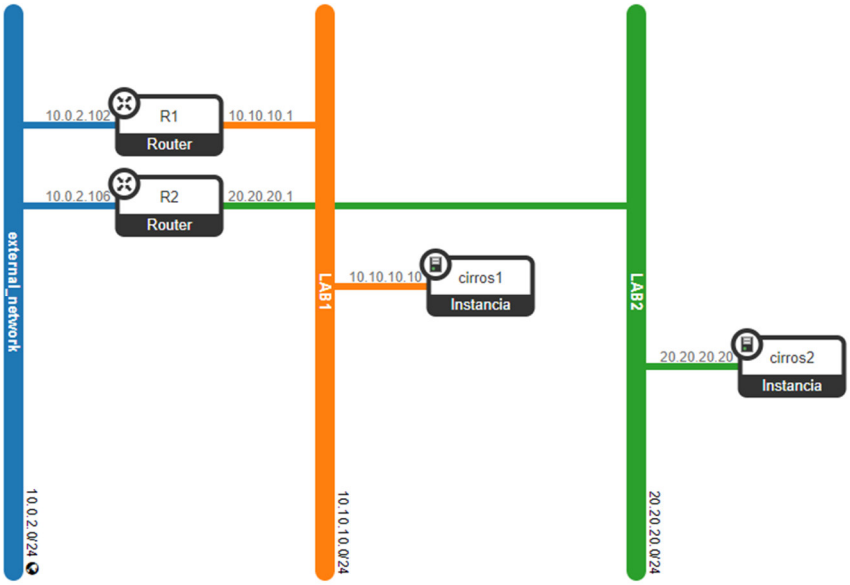


Figura 12 - Creación de instancias en OpenStack

En este momento las maquina virtuales solo pueden hacer ping para probar la conexión hacia redes externas. Para hacerlo bidireccional, es necesario añadir una nueva regla al grupo de seguridad para permitir el uso del protocolo ICMP desde cualquier equipo.

```
# openstack security group rule create --remote-ip 0.0.0.0/0 --protocol
icmp --ingress default
```

Es posible hacer una prueba de conexión entre ambas máquinas. En la figura 13 se muestra en ping desde la maquina 1 a la maquina 2 y en la figura 14 se realiza el ping desde la maquina 2 a la maquina 1. En ambas no se obtiene una respuesta.

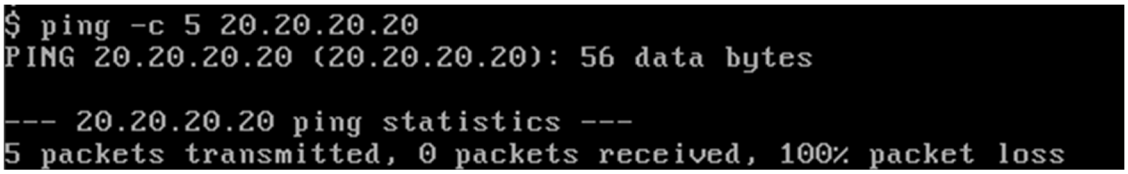


Figura 13 - Ping desde 1 hasta 2, sin conexión

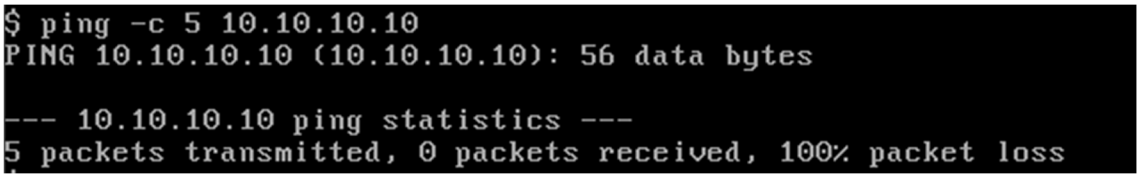


Figura 14 - Ping desde 2 hasta 1, sin conexión

Ninguna de las maquinas se puede conectar con la otra, esto es debido a que no saben por dónde ir a la otra red por lo que es necesario añadir las rutas.

```
# openstack router set --route destination=20.20.20.0/24,gateway=10.0.2.106
R1
# openstack router set --route destination=10.10.10.0/24,gateway=10.0.2.102
R2
```

Con las rutas establecidas se puede comprobar la conectividad entre los diferentes equipos. En la figura 15 se muestra el ping desde la maquina 1 a la maquina 2 y en figura 16 el ping se realiza desde la maquina 2 a la 1. Esta vez sí se obtiene una respuesta para ambos.

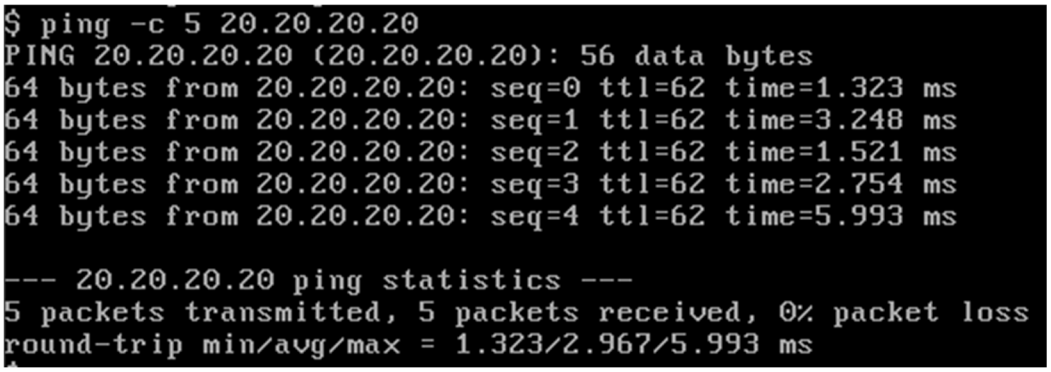


Figura 15 - Ping de 1 hasta 2, con conexión

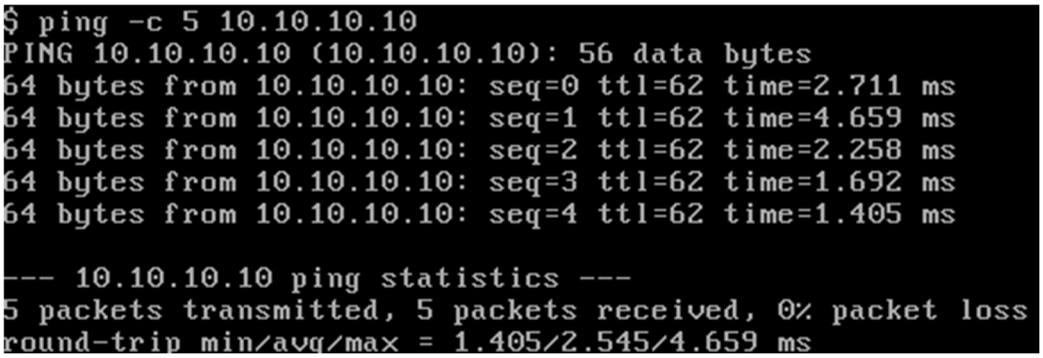


Figura 16 - Ping de 2 hasta 1, con conexión

Con esto el laboratorio virtual quedaría completado.

3.5 Automatización

Todo el proceso anterior, que puede ser considerado como una buena práctica en sí para el alumno, seguramente debiera repetirse en diferentes sesiones o prácticas, por lo que puede ser necesario automatizar dicho proceso para poder retomar sesiones anteriores, o simplemente, organizar diferentes laboratorios basados en esta configuración básica. El proceso de automatización más sencillo consiste en la creación de un script. En este ejemplo práctico, dicho script recibe como parámetros de entrada un usuario y una contraseña, crea la cuenta de usuario, crea el proyecto para el usuario y realiza el despliegue del laboratorio para ser usado por ese usuario.

Además de los comandos vistos en apartados anteriores, existen algunos comandos importantes que tienen que ser utilizados ahora y que van a ser explicados brevemente:

```
# openstack user create –password contraseña usuario
```

Crea un nuevo usuario de nombre “usuario” y con contraseña “contraseña”.

```
# openstack project create "proyecto"
```

Crea un nuevo proyecto de nombre “proyecto”.

```
# openstack role add --project "proyecto" --user usuario _member_
```

Al usuario “user” se le da el rol de “_member_” en el proyecto “proyecto”.

Para lanzar el script, el cual se ha llamado “lab.sh” se escribe en consola.

```
# . lab.sh usuario contraseña
```

El script recibe como entrada el nombre de usuario y la contraseña, estas variables se almacenan en $1 y $2 respectivamente.

A continuación, se procede a explicar paso por paso el funcionamiento del script y diferentes opciones que se pueden tomar.

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

En la primera línea se indica al sistema operativo qué Shell se va a usar para interpretar el script y la ubicación de dicho shell.

```
4    export OS_AUTH_URL=http://192.168.100.100:5000/v3
5    export OS_PROJECT_ID=1650279560964044aeb19029de67d568
6    export OS_PROJECT_NAME="admin"
7    export OS_USER_DOMAIN_NAME="Default"
8    if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset OS_USER_DOMAIN_NAME; fi
9    export OS_PROJECT_DOMAIN_ID="default"
10   if [ -z "$OS_PROJECT_DOMAIN_ID" ]; then unset OS_PROJECT_DOMAIN_ID; fi
11   unset OS_TENANT_ID
12   unset OS_TENANT_NAME
13   export OS_USERNAME="admin"
14   export OS_PASSWORD="telematica"
15   export OS_REGION_NAME="RegionOne"
16   if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
17   export OS_INTERFACE=public
18   export OS_IDENTITY_API_VERSION=3
```

Se cargan las credenciales de administrador. Estas credenciales se pueden obtener usando instrucciones OpenStack o bien desde dashboard. Una vez identificados se selecciona en la esquina superior derecha "admin > Fichero RC de OpenStack v3", como se muestra en la figura 17.

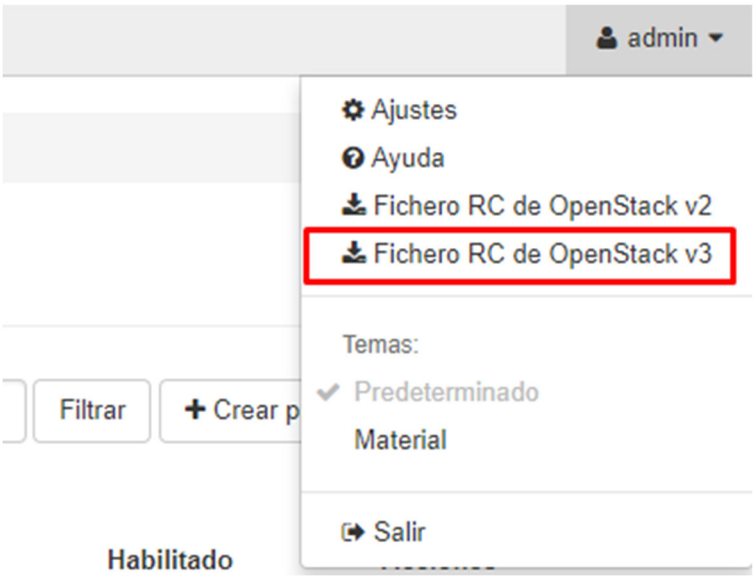


Figura 17 - Descargar credenciales de usuario OpenStack

De esta manera se descargará un fichero llamado "admin-openrc.sh". En el script se han conseguido las credenciales de esta manera ya que van a ser siempre las mismas y ahorra tiempo de ejecución del script al no usar los comandos OpenStack. La única diferencia con las credenciales del script es que, en dicho fichero, por motivos de seguridad, se pide que la contraseña del usuario sea introducida por teclado.

```
29  echo "Please enter your OpenStack Password for project $OS_PROJECT_NAME
as user $OS_USERNAME: "
30  read -sr OS_PASSWORD_INPUT
31  export OS_PASSWORD=$OS_PASSWORD_INPUT
```

Como se trata de automatizar el proceso esta parte se modifica para que al ejecutar el script se realice todo seguido.

```
21   test=$(openstack user create --password $2 $1)
22   if [ -z "$test" ]
23   then
24   echo "Nombre de usuario en uso"
25   exit;
```

```
26   fi
```

En esta parte se intenta crear el nuevo usuario $1 con contraseña $2, por lo que se usa la opción "--password", y se almacena el resultado en una variable llamada "test". A continuación, se realiza un if donde con la opción "-z" se comprueba si la longitud de "test" es cero, es decir, la variable está vacía. Si "test" no está vacía, continúa con la ejecución del script, pero si está vacía se cancela su ejecución. El que la variable "test" esté vacía significa que la instrucción "openstack user create --password $2 $1" no ha devuelto ningún resultado, y esto es debido a que el usuario ya existe y no puede realizar la instrucción. En este caso se mostraría por pantalla "Nombre de usuario en uso".

```
29   openstack project create "proyecto_$1"
```

Se crea el nuevo proyecto el cual se llama "proyecto_usuario".

En este punto ya estarían creados el nuevo usuario y el proyecto. Para que el usuario pueda acceder al dashboard, o realizar operaciones de creación de redes, routers, instancias, etc., se necesita que el usuario este asociado a algún proyecto. Para ello se realiza la siguiente instrucción.

```
32   openstack role add --project "proyecto_$1" --user $1 _member_
33   openstack role add --project "proyecto_$1" --user admin admin
```

Con esto el usuario ya puede acceder al proyecto. La opción "--project" indica el proyecto, la opción "--user" indica el usuario y por último se indica el rol, en este caso "_member_". Se le da el rol de "_member_" y no el de "admin", ya que "_member_" puede realizar todas las operaciones necesarias dentro del proyecto, como crear routers, redes, instancias, etc. Si se le diera el rol "admin" no solo sería administrador dentro de ese proyecto, sino que sería administrador de OpenStack, pudiendo borrar usuarios o provocar el falo de todos los servicios.

El script continúa con la carga de las credenciales del nuevo usuario.

```
35   #Credenciales del nuevo usuario
36   unset OS_USERNAME
37   unset OS_PASSWORD
38   unset OS_PROJECT_NAME
```

```
39   unset OS_PROJECT_ID
40   export OS_USERNAME="$1"
41   export OS_PASSWORD="$2"
42   export OS_PROJECT_NAME="proyecto_$1"
43   export OS_PROJECT_ID=$(openstack project list -f value -c ID)
```

Realmente no es necesario cambiar las credenciales como "OS_USERNAME" o "OS_PASSWORD", ya que se podría seguir trabajando con el usuario administrador, pero se ha optado por usar las del usuario ya que requieren menos instrucciones para obtener la ID y así ahorrar tiempo en la ejecución: Además, así se permite demostrar que con el rol de "_member_" se pueden realizar el resto de las operaciones del script.

Las únicas opciones que se deben cambiar son "OS_PROJECT_NAME", por el del nuevo proyecto, y "OS_PROJECT_ID", por la id del nuevo proyecto. Con ambos cambios las operaciones de creación de routers, redes, etc., se realizan sobre el nuevo proyecto. Para obtener estos parámetros no queda más remedio que usar las instrucciones OpenStack. La instrucción "openstack project list -f value -c ID" devolverá una lista de los proyectos a los que está asociado el usuario y la opción "-c ID" devuelve la columna ID. Como el usuario solo está asociado a un solo proyecto el resultado del comando es una única línea, por lo que no es necesario realizar ninguna otra opción para identificar la ID del nuevo proyecto.

Si se optase por seguir con el usuario "admin", el comando anterior cambiaria algo, ya que la lista de proyectos tendría más de una entrada. Para extraer la ID desde el usuario "admin" el comando correspondiente seria.

```
export OS_PROJECT_ID=$(openstack project list -f value | grep -w
"proyecto_$1" | cut -d " " -f 1)
```

Como el resultado de realizar "openstack project list -f value" devuelve una lista con más de una entrada, no se puede usar la opción "-c ID", ya que devolvería más de un valor y no sería capaz de identificar la ID del nuevo proyecto. Por ello, el resultado del comando OpenStack debe ser pasado al comando "grep" con la opción "-w "proyecto_$1". Así, se busca sobre la lista que devuelve OpenStack el nombre del proyecto, devolviendo como resultado una línea con el formato "ID nombre_proyecto. A continuación, se pasa el resultado al comando "cut", que con la opción "-d " ", indica el delimitador, que en este

caso es un espacio. Con la opción "-f 1" se le indicaría que se recoge todo desde el principio, hasta que se encuentre con el primer delimitador, es decir, que extraiga el ID.

Con esto queda acabada la parte de la creación de usuario y ya es posible proceder a la creación de la red del laboratorio virtual.

Primero los router R1 y R2:

```
46   openstack router create R1
47   openstack router create R2
```

A continuación, aparece la siguiente línea:

```
49   externa=$(openstack network list --external --provider-network-type
flat -f value -c Name)
```

La instrucción "openstack network list" devuelve una lista con todas las redes en OpenStack, y con la opción "--external --provider-network-type flat" devolverá las redes externas. Por último, la opción "-c Name" devuelve solo la columna Name.

Otra forma de obtener el mismo resultado habría sido escribir directamente el nombre de la red externa entre comillas, o en los comandos siguientes sustituir donde pone "$externa" por el nombre de la red externa. Sin embargo, la opción anterior es mucho más limpia y útil.

A continuación, se crean las redes del laboratorio y se conectan estas con la red externa.

```
52   openstack network create --internal LAB1
53   openstack subnet create --network LAB1 --allocation-pool
start=10.10.10.5,end=10.10.10.254 --dns-nameserver 8.8.8.8 --dns-nameserver
8.8.4.4 --gateway 10.10.10.1 --subnet-range 10.10.10.0/24 subnet_LAB1
```

El primer comando crea una red interna a la que llama LAB1. La opción "--internal" no es necesaria, ya que es el valor que toma por defecto el comando. A continuación, se crea la subred. Con la opción "--network" se indica el nombre de la red virtual, con "--allocation-pool" el pool de direcciones disponibles, con "--dns-nameserver" las direcciones a los servidores DNS, con "--gateway" la dirección del gateway, con "--subnet-range" el rango de la subred y por último se le pone nombre a la subred.

Todas las redes van a tener las mismas direcciones en todos los proyectos, pero como son redes virtuales en diferentes proyectos, son totalmente independientes entre sí. Esto permite que, si durante la realización de una práctica, el alumno comete algún error en algún comando, le sea más fácil tanto al alumno como al profesor detectar el fallo y corregirlo.

Falta conectar la red recién creada con el router, y el router con la red externa.

```
54   openstack router add subnet R1 subnet_LAB1
55   openstack router set --external-gateway $externa R1
```

Hay dos formas de conectar el router a cada una de las redes, dependiendo de si la red a la que la quieres conectar es una red virtual o es la red externa. En el caso de querer conectarlo con una red virtual se utiliza "openstack router add subnet R1 subnet_LAB1". En el caso de querer conectarlo con la red externa se utiliza "openstack router set --external-gateway $externa R1".

Se repite el proceso para el router R2.

```
58   openstack network create --internal LAB2
59   openstack subnet create --network LAB2 --allocation-pool
start=20.20.20.5,end=20.20.20.254 --dns-nameserver 8.8.8.8 --dns-nameserver
8.8.4.4 --gateway 20.20.20.1 --subnet-range 20.20.20.0/24 subnet_LAB2
60   openstack router add subnet R2 subnet_LAB2
61   openstack router set --external-gateway $externa R2
```

El siguiente paso sería indicar cómo llegar de R1 a R2 y viceversa, para ello se ejecutan los siguientes comandos.

```
64   ruta=$(openstack router show R1 -f value -c external_gateway_info |
cut -d "\"" -f 16)
65   openstack router set --route destination=10.10.10.0/24,gateway=$ruta
R2
```

Lo primero es conseguir la IP de la interfaz de red del router R1 conectada a la red externa, mediante "openstack router show R1", que devuelve las características del router R1. La opción "-f value" es para dar formato al resultado y la opción "-c external_gateway_info" para que solo devuelva la columna. Esta columna contiene más información que la IP, como es el caso del IDs. Para extraer únicamente la IP se vuelve a usar el comando "cut",

pero esta vez usando como delimitador "/"". Una vez extraída la IP, se añade el siguiente salto al R2. Con esto se conecta R2 con R1. Para acabar la configuración de red hay que añadir a R1 la ruta a R2.

```
66   ruta=$(openstack router show R2 -f value -c external_gateway_info |
cut -d "\"" -f 16)
67   openstack router set --route destination=20.20.20.0/24,gateway=$ruta
R1
```

Por defecto no se permite tráfico ICMP entrante, por lo que no es posible realizar un ping para comprobar que realmente las redes están conectadas. Para ello es necesario crear una nueva regla que lo permita.

```
70   id=$(openstack security group list -f value -c ID)
80   openstack security group rule create --remote-ip 0.0.0.0/0 --protocol
icmp --ingress $id
```

El primer paso es obtener la ID del grupo de seguridad del proyecto. Para ello se usa "openstack security group list", que devuelve la lista de los grupos de seguridad en el proyecto, y que en este caso solo tendrá una entrada. Por lo tanto, usando la opción "-c ID", se extrae la ID. A continuación, se crea la nueva regla en la que, con la opción "--ingress" se le indica que es para el tráfico entrante, "--protocol icmp" indica el protocolo y por último, la opción "--remote-ip 0.0.0.0/0" indica que la dirección de origen puede ser cualquiera.

Para acabar el script crean las instancias.

```
83   openstack server create --flavor m1.tiny --image cirros --nic net-
id=LAB1,v4-fixed-ip=10.10.10.10 --security-group default cirros1
84   openstack server create --flavor m1.tiny --image cirros --nic net-
id=LAB2,v4-fixed-ip=20.20.20.20 --security-group default cirros2
```

Para crear una instancia es necesario determinar ciertos parámetros. Con la opción "--flavor" se selecciona el sabor. Es posible obtener la lista de sabores disponibles con sus características mediante el comando.

```
# openstack flavor list
```

Otro parámetro que hay que especificar es la imagen para crear la instancia. Al igual que con el sabor, es posible utilizar un comando para que OpenStack nos devuelva la lista de imágenes que se encuentran cargadas cargadas.

```
# openstack image list
```

Con la opción "--security-group" se indica el grupo de seguridad al que pertenece. Como no se ha creado ninguno, solo existe el grupo default, el cual se crea por defecto al crear el proyecto. Este es el grupo sobre el cual se ha creado la nueva regla para el tráfico entrante ICMP. Por su parte, la opción "--nic net-id=" permite indicar la red a la cual se conecta la instancia.

Con los parámetros especificados sería suficiente para que OpenStack cree las instancias, pero además se ha inlcuido otro parámetro "v4-fixed-ip" con el fin de especificar la IP de la máquina que se quiera. Este parámetro se ha indicado porque en caso contrario, OpenStack da una IP del pool de direcciones disponibles de manera a aleatoria. De esta manera se acota el espacio de direcciones ante posibles errores.

Otra opción que puede ser interesante para ciertas circunstancias es la opción "--wait". Este parámetro introduce una espera hasta que la instancia este construida y operativa. Por defecto el comando OpenStack la manda construir y pasa a la siguiente instrucción, y si ésta necesitara que la instancia estuviese ya conectada a una red y con una dirección determinada, provocaría un fallo. Finalmente, esta opción no ha sido utilizada en el script ya que, con la implementación posterior sobre PHP, generaba errores de ejecución.

4. Servicio de Automatización

Otro objetivo del trabajo es que el alumno pueda crear su laboratorio virtual sin la necesidad de ejecutar el script directamente sobre el servidor OpenStack. Una de las razones es que, precisamente en el script se encuentran las credenciales de administrador y por lo tanto, sería fácil acceder como administrador, borrar proyectos, usuarios o quitar ciertas funcionalidades a OpenStack. Para evitar este problema en un primer momento se hizo una búsqueda de un plugin para OpenStack que permitiese la ejecución del script sin la necesidad de tener una cuenta de usuario en OpenStack. Inicialmente no se encontró dicho plugin, por lo que se optó por definir una solución alternativa, que consiste en crear

un servicio web en el cual el alumno introduzca su usuario y contraseña, y el propio servicio web se encarga de ejecutar el script y devuelve al alumno el acceso al dashboard de OpenStack. De esta forma se evita que el alumno haga uso de información privilegiada.

En este apartado se explica la instalación de los diferentes componentes que son necesarios para crear el nuevo servicio de automatización. En concreto, es necesario utilizar Apache y PHP, así como crear los ficheros de configuración correspondientes y solucionar un error común a la hora de ejecutar scripts en bash sobre PHP.

4.1 Instalación de Apache

Como primer paso para la creación del servicio web se procede a la instalación de un servidor Apache y su configuración para que se inicie con el arranque del sistema.

```
# yum -y install httpd
# systemctl start httpd
# systemctl enable httpd
```

El fichero de configuración de Apache se encuentra en "/etc/httpd/conf/httpd.conf", mientras que los contenidos del servidor se almacenan en "/var/www/html".

Para comprobar que la instalación ha sido correcta basta con acceder desde un navegador a la dirección IP del servidor. Debería aparecer la página de test de Apache como en la Figura 18.

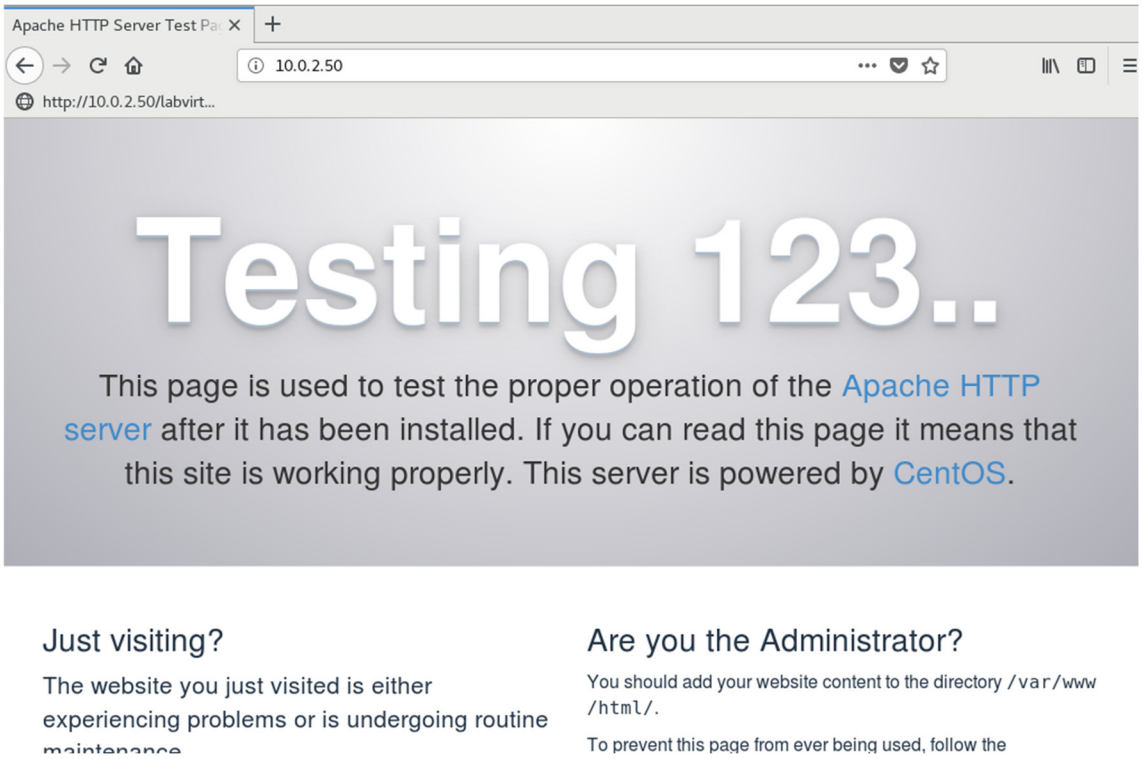


Figura 18 - Pagina test de Apache

4.2 Instalación de PHP

A continuación, es necesario instalar la versión 7.2 de PHP, para lo cual se recomienda activar los repositorios Remi y EPEL con los siguientes comandos.

```
# yum -y install epel-release
# rpm -Uvh http://rpms.famillecollet.com/enterprise/remi-release-7.rpm
```

Una vez con los repositorios instalados se procede a instalar PHP con el comando:

```
# yum --enablerepo=remi-php72 -y install php
```

Para la comprobación de la instalación de PHP basta con ejecutar:

```
# php -v

Resultado
PHP 7.2.9 (cli) (built: Aug 15 2018 09:19:33 ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
```

Asimismo, es necesario la instalación de ciertos módulos adicionales para el correcto funcionamiento del nuevo servicio web.

```
# yum --enablerepo=remi-php72 -y install php-xml php-soap php-xmlrpc
php-mbstring php-json php-gd php-mcrypt
```

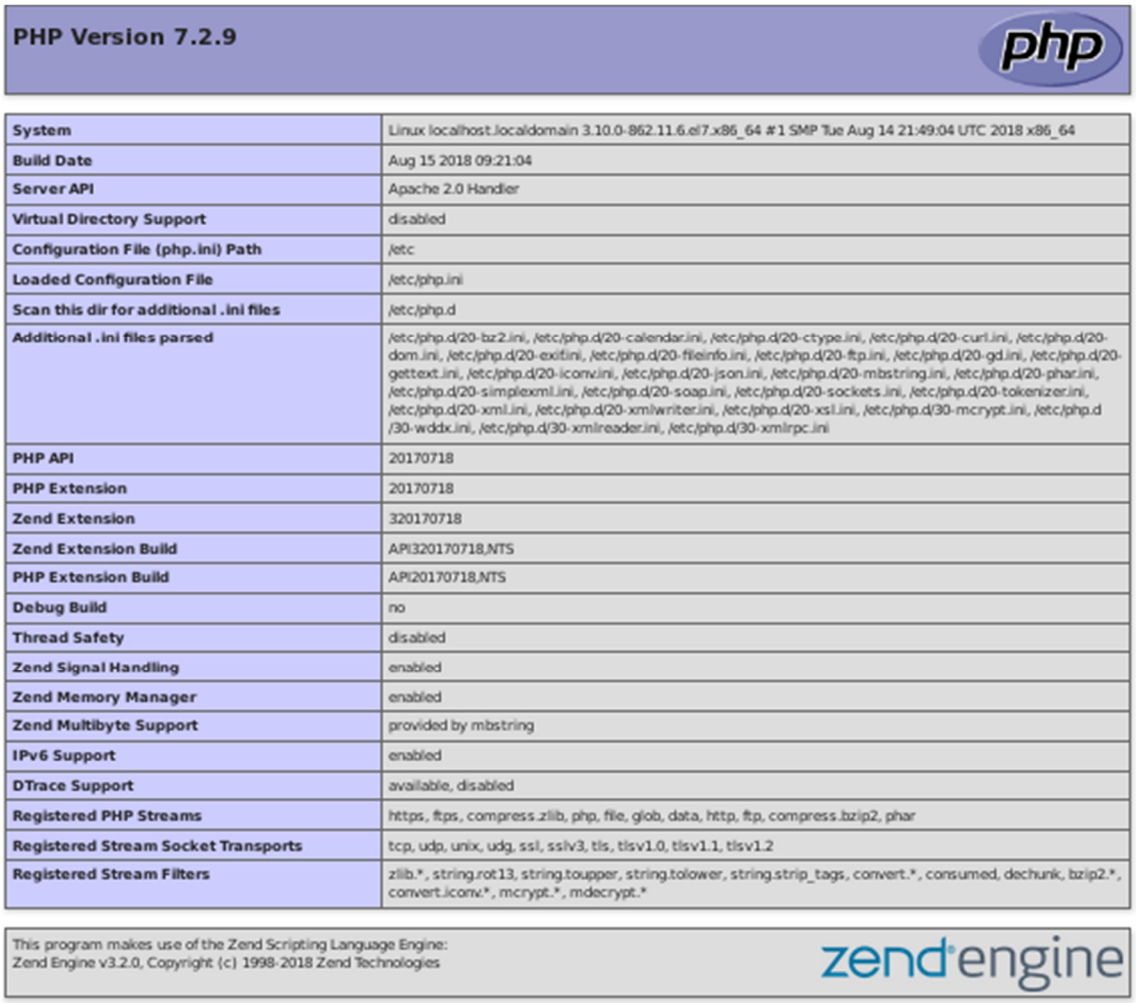
Tras lo cual será necesario reiniciar el servidor Apache para que los cambios surtan efecto.

```
# systemctl restart httpd
```

Para realizar la comprobación del correcto funcionamiento del conjunto, en "/var/www/html/" se crea un fichero PHP con nombre "info.php" y el siguiente código.

```
<?php
phpinfo();
?>
```

La función phpinfo() devuelve todos los parámetros de configuración de PHP, incluidos todos los módulos que han sido activados anteriormente. Un ejemplo de dicha configuración se muestra en la figura 19.



PHP Version 7.2.9

System	Linux localhost.localdomain 3.10.0-862.11.6.el7.x86_64 #1 SMP Tue Aug 14 21:49:04 UTC 2018 x86_64
Build Date	Aug 15 2018 09:21:04
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-json.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-soap.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mcrypt.ini, /etc/php.d/30-wddx.ini, /etc/php.d/30-xmlreader.ini, /etc/php.d/30-xmlrpc.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*, mcrypt.*, mdecrypt.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies

zend engine

Figura 19 - Resultado de phpinfo()

4.3 Servicio Web

A continuación, se va a mostrar cómo crear un servicio web en PHP que permita ejecutar cualquier script que realice la creación y configuración de laboratorios virtuales. Para ello es necesario crear una clase con todos los métodos que van a ser soportados, tanto en el cliente como en el servidor.

A modo de ejemplo, un primer servicio web va a ser creado en el directorio web de Apache, que por defecto es "/var/www/html". Para ello se crea una carpeta con el nombre del servicio a crear, como por ejemplo "labvirtual".

En este directorio se va a crear un archivo denominado "metodo.php", en donde se indicarán los métodos que va a soportar el servicio web.

```
1  <?php
2    class laboratorio
3    {
4      public function crear($user, $pass)
5      {
6       return exec("/usr/bin/env bash
        /var/www/html/labvirtual/lab.sh $user $pass");
7      }
8    }
9  ?>
```

En el archivo PHP de ejemplo, se hace referencia a una clase denominada “laboratorio”, la cual es la que implementa en sí los servicios ofrecidos. Dichos servicios aparecen en forma de funciones, como por ejemplo la función “crear”, la cual recibe dos parámetros, usuario (“user”) y contraseña (“pass”), y se encarga de llamar al script correspondiente. Para hacer la llamada, se hace uso de la función “exec”, a la cual basta con indicar la ubicación de la Shell que va a ser usada, así como la ubicación del script. Con el “return” previo, la llamada devuelve el resultado de la ejecución del script. En un futuro está previsto poblar el servicio Web con nuevas funciones que permitan realizar la gestión completa del entorno de laboratorios virtuales así creado.

El siguiente archivo a definir es el correspondiente al servidor, por lo que se llamará “server.php”.

```
<?php
 require_once('metodo.php');

 #Creamos el servidor
 $server = new SoapServer(null,array('uri' => 'urn:webservices'));
 $server->setClass('laboratorio');

 #Atender las llamadas
 $server->handle();
?>
```

Este es el archivo donde se va a implementar un punto de servicio específico, que será el que atienda las peticiones que se generen. Tras indicar el archivo de definición de funciones, se crea un servidor específico mediante función SoapServer de PHP, al qiue

se le asigna la clase implementadora, en el ejemplo, la clase “laboratorio”. La gestión de las peticiones se realiza mediante la asignación del correspondiente “handle()”, el cual será el responsable de llamar a las funciones necesarias y devolver la respuesta.

Por si solos estos dos archivos no realizan ninguna función desde el punto de vista del usuario. Para dar una interfaz gráfica al mismo, es necesario crear un último archivo, que será precisamente el que actúe como cliente del servicio así implementado, por lo que recibe el nombre “cliente.php”.

```
1   <!DOCTYPE HTML>
2   <html lang="es-ES">
3   <head>
4     <title>Crear laboratorio</title>
5     <meta charset="utf-8">
6   </head>
7
8   <body>
9     <form action="cliente.php" method="POST">
10      <table>
11        <tr>
12          <td>Usuario <input type="text" name="usuario"></td>
13        </tr>
14        <tr>
15          <td>Contraseña <input type="password" name="contraseña"></td>
16        </tr>
17        <tr>
18          <td> <input type="submit" value="crear"> </td>
19        </tr>
20      </table>
21    </form>
22  </body>
23  </html>
24
25  <?php
26  if($_POST)
27  {
28    $user = $_POST
29    ['usuario'];
```

```
30    $pass = $_POST
31    ['contraseña'];
33    $client = new SoapClient(null,Array('location' =>
      'http://10.0.2.50/labvirtual/servicio.php','uri' =>
      'urn:webservices',));
34    $valor=$client->crear($user,$pass);
35    if("Nombre de usuario en uso" === $valor)
37       echo "Nombre de usuario en uso";
38    else
39     echo "<a
       href=http://192.168.100.100/dashboard>
       http://192.168.100.100/dashboard </a>";
40  }
41  ?>
```

La primera parte del documento se corresponde con el formato típico de cualquier documento escrito en lenguaje HTML. Cabe destacar el código incluido en el apartado <body>, en el que se ha optado por dar un formato de tabla, en cuyas celdas se han puesto unas entradas para que sean completadas con el nombre de usuario y la contraseña del alumno que desee crear el laboratorio virtual. El bloque se finaliza con un botón que al ser pulsado realiza el envío de ambos valores.

La parte fundamental está a continuación, en el código PHP. Esta parte se encarga de usar los datos que se envían en el método POST anterior, y mediante el uso de "SoapClient" realiza una petición al servidor correspondiente, y así poder crear el laboratorio virtual con los datos de usuario y contraseña obtenidos. Por último, se encarga de mostrar los resultados obtenidos de la llamada al script.

Es posible hacer una rápida comprobación de los tres documentos anteriores sin más que usar el siguiente código en consola.

```
# php metodo.php
# php servicio.php
# php cliente.php
<!DOCTYPE HTML>
<html lang="es-ES">
<head>
  <title>Crear laboratorio</title>
```

```
  <meta charset="utf-8">
</head>

<body>
<form action="cliente.php" method="POST">
<table>
<tr>
<td>Usuario <input type="text" name="usuario"></td>
</tr>
<tr>
<td>Contraseña <input type="password" name="contraseña"></td>
</tr>
<tr>
<td> <input type="submit" value="crear"> </td>
</tr>
</table>
</form>
</body>
</html>
```

Como se puede observar, los archivos “método.php” y “server.php” no devuelven nada, lo cual es normal si no existen errores en ellos. En el caso de “cliente.php”, si todo está correcto este directamente devolverá el código HTML.

Tras la comprobación anterior, es el momento de abrir un navegador y utilizar la URL “http://10.0.2.50/labvirtual/cliente.php”. Se deberá abrir una página como la que aparece en la figura 20.

Figura 20 - Web

Si todo está correcto, al introducir un usuario y contraseña válidos, el laboratorio virtual debería haberse creado. Sin embargo, pueden surgir efectos no esperados, como por ejemplo errores al devolver la salida del script, fallos a la hora de crear el laboratorio

virtual, etc. Es posible depurar todos los posibles fallos añadiendo a los archivos "cliente.php" y "metodo.php" el siguiente código.

```
error_reporting(E_ALL);
ini_set('display_errors',1);
```

Con este código, en caso de error se mostrarán detalles sobre el mismo, lo que facilita su resolución. Hay que tener especial cuidado en incluir estos códigos solo en la parte PHP del archivo "cliente.php".

Un error habitual que tuvo que ser corregido durante la preparación de este trabajo, es el que devuelve el siguiente identificador: "Uncaught SoapFault exception: [HTTP] Error Fetching http headers". Este error suele venir debido a unos temporizadores que tiene PHP para la ejecución de scripts, según los cuales, si no se logra ejecutar el script en el tiempo máximo establecido, se reporta este error. Una solución inmediata pasa por modificar el fichero de configuración de PHP denominado "php.ini", y que se encuentra en "/etc/". Para que el servicio funcione correctamente es necesario modificar al menos dos parámetros, marcados a continuación en negrita:

```
380   ; Maximum execution time of each script, in seconds
381   ; http://php.net/max-execution-time
382   ; Note: This directive is hardcoded to 0 for the CLI SAPI
383   max_execution_time = 200
                      . . .
851   ; Default timeout for socket based streams (seconds)
852   ; http://php.net/default-socket-timeout
853   default_socket_timeout = 200
```

Para calcular el tiempo necesario para ejecutar el script de forma exitosa se recomienda ejecuta el script desde la ventana de comandos, medir el tiempo que tarda en finalizar y añadir un pequeño margen. El resultado debe ser incorporado en forma de segundos en los parámetros "max_execution_time" y "default_socket_timeout" y a continuación se reinicia el servidor para que los cambios surtan efecto.

```
# systemctl restart httpd
```

Desde el punto de vista de presentación, puede ser interesante darle a la página anterior un aspecto visualmente más atractivo, para lo cual se puede usar Bootstrap. Bootstrap es un framework CSS que permite dar formato a sitios web de una forma rápida utilizando los elementos de sus librerías. A continuación, se muestra un ejemplo del código generado por Bootstrap.

```
1   <!DOCTYPE HTML>
2   <html lang="es-ES">
3   <head>
4     <title>Crear laboratorio</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7    <link rel="stylesheet"
     href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css
     /bootstrap.min.css">
8    <script src="https://ajax.googleapis.com/ajax/
     libs/jquery/3.3.1/jquery.min.js"></script>
9    <script src="https://cdnjs.cloudflare.com/ajax/
     libs/popper.js/1.14.3/umd/popper.min.js"></script>
10   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/
     bootstrap.min.js"></script>
11  </head>
12
13  <body>
14    <form action="cliente.php" method="POST">
15      <div class="jumbotron text-center">
16        <h1>Trabajo Fin de Grado</h1>
17      </div>
18
19              <div class="container">
20              <div class="row">
21              <div class="col-sm-3">
22              </div>
23              <div class="col-sm-6">
24          <h4 style="text-align:center;">Crearlaboratorio</h4>
25
26              <div class="row">
27                 <div class="col-sm-4" style="text-align:right;">
```

```
28                   <label for="usuario">Usuario </label>
29                  </div>
32                  <div class="col-sm-6">
31                  <input type="text" style="width:100%;"name="usuario"/>
32              </div>
33              <div class="col-sm-2">
34              </div>
35          </div>
36
37          <div class="row">
38              <div class="col-sm-4" style="text-align:right;">
39                 <label for="contraseña" >Contraseña </label>
40              </div>
41              <div class="col-sm-6">
42      <input type="password" style="width:100%;" name="contraseña"/>
43              </div>
44              <div class="col-sm-2">
45                 </div>
46                 </div>
47
48       <br/>
49
50   <?php
51   if($_POST){
52    $user = $_POST
53    ['usuario'];
54    $pass = $_POST
55    ['contraseña'];
56    $client = new SoapClient(null,Array('location' =>
      'http://10.0.2.50/labvirtual/servicio.php','uri' =>
      'urn:webservices',));
57    $valor=$client->crear($user,$pass);
58    if("Nombre de usuario en uso" === $valor){
59      echo "<div class=\"row\">";
60      echo "<div class=\"col-sm-12\" style=\"text-align:center;\">";
61      echo "<p class=\"btn bg-danger text-white\"
        style=\"width:100%;\">Nombre de usuario en uso</p>";
62      echo "</div></div>";
63   ?>
```

```
64
65      <div class="row">
66      <div class="col-sm-12">
67     <input type="submit" style="width: 100%" class="btn btn-success"
         value="crear de nuevo">
68      </div>
69    </div>
70
71   <?php
72   } else {
73   echo "<a href=\"http://192.168.100.100/dashboard\"
     class=\"btn btn-info\" style=\"width: 100%\">Acceso a Dashboard</a>";
74   }
75   } else {
76   ?>
77
78     <div class="row">
79     <div class="col-sm-12">
80     <input type="submit" style="width: 100%"
        class="btn btn-success" value="crear">
81     </div>
82    </div>
83
84    <?php
85      }
86    ?>
87
88              </div>
89              <div class="col-sm-3">
90              </div>
91          </div>
92      </div>
93  </form>
94  </body>
95  </html>
```

A continuación, la figura 21 muestra el resultado final:

Figura 21 - Web con Bootstrap

5. Conclusiones y líneas futuras

A lo largo de este trabajo se muestra un ejemplo de automatización del proceso de creación de laboratorios virtuales en infraestructuras cloud OpenStack mediante la creación de un script y el uso de un servicio web para su ejecución.

La implementación se realiza sobre una plataforma de demostración del sistema OpenStack para la provisión de servicios de nube. La configuración de dicha plataforma suele ser denominada "All-in-one", ya que concentra todos los servicios necesarios en una única máquina servidora. Como condición previa de la plataforma, esta tenía que ser portable y reproducible en equipos portátiles para no tener que depender de un sistema en producción. Es por ello que el desarrollo se ha llevado a cabo sobre una máquina virtual con sistema operativo CentOS instalada en un portátil de gama media mediante el sistema de virtualización VirtualBox. Con el uso del paquete PackStack el despliegue de OpenStack sobre el sistema preparado resulta hasta cierto punto sencillo y permite poner en marcha la plataforma esperada en tiempos relativamente cortos.

Sin embargo, el objetivo principal del proyecto no es la plataforma de demostración en sí, sino el estudio de mecanismos específicos para la automatización de los procesos de gestión de laboratorios virtuales basados en servicios generados mediante OpenStack (creación y configuración de las infraestructuras virtuales necesarias). Este trabajo parte de la experiencia de otros trabajos anteriores, relacionados con este tema, en los que, por ejemplo, se planteaba la posibilidad de utilizar cierto tipo de scripts como línea futura de desarrollo. Precisamente, allí ya se apuntaban problemas tales como la imposibilidad de crear estructuras desde cero, la obligación de utilizar las ventanas de comandos o necesitar credenciales específicas de administrador.

Gracias a esas experiencias anteriores, y especialmente a su documentación, en este trabajo se han abordado directamente dichos problemas, y se han buscado las correspondientes soluciones, entre las que cabe destacar la definición del marco de script necesario, y su uso mediante la creación de servicios web específicos. Gracias a esta solución, se consigue ocultar las credenciales de administración, necesarias para la creación de determinados objetos, se evita la activación de consolas de administración y se simplifica el uso de escenarios complejos por parte de los usuarios finales, es decir, los alumnos.

Sin embargo, la solución aquí propuesta no está exenta de problemas y sus correspondientes riesgos. El servicio web, tal y como se presenta, posibilita el acceso desde cualquier usuario que acceda a la web para la creación sin límite de escenarios sin más que generarlos, y consumir sin control los recursos del sistema, e incluso una posible denegación de servicio. En las sucesivas versiones de esta solución es necesario integrar el sistema de identificación de OpenStack, denominado Keystone, para hacer el primer control de acceso, y a continuación, modular el sistema de gestión de políticas de consumo de recursos aplicadas a cada usuario también incluido en OpenStack.

En cuanto a su aplicación en la creación de laboratorios virtuales para la docencia, el servicio Web aquí creado debe ser poblado con todo un conjunto de funcionalidades que permita definir entornos de trabajo completos, tanto desde el punto de vista del alumno, como del profesor, permitiendo, por ejemplo, de una manera fácil e intuitiva definir parámetros a medida para la creación de proyectos a medida, facilitando así el despliegue de diferentes topologías de red según las necesidades de la práctica que se vaya a realizar.

Bibliografía

1. Alejandro Abascal Crespo «Aplicación de LDAP como método de autenticación en entornos de Cloud Privada» [En línea]. Avaible: https://repositorio.unican.es/xmlui/handle/10902/12184. [Último acceso: 22 10 2018]

2. Martín Pereira Diéguez «Implementación de un entorno cloud en las infraestructuras del laboratorio de aplicaciones telemáticas» [En línea]. Avaible: https://repositorio.unican.es/xmlui/handle/10902/11402. [Último acceso: 22 10 2018]

3. National Institute of Standards and Technology, «The NIST Definition of Cloud Computing» [En línea]. Avaible: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. [Último acceso: 9 9 2018]

4. NetworkWorld, «What is a hypervisor?» [En línea]. Avaible: https://www.networkworld.com/article/3243262/virtualization/what-is-a-hypervisor.html. [Último acceso: 9 9 2018]

5. Alberto E. García Gutiérrez, «Apuntes de la asignatura Servicios Inteligentes en Redes» [En línea]. Avaible: https://www.tlmat.unican.es/index.php?l=es&p=teaching&s=subjects&ss=g_sir& . [Último acceso: 10 9 2018]

6. OpenStack, «OpenStack Services» [En línea]. Avaible: https://www.openstack.org/software/project-navigator/openstack-components#openstack-services. [Último acceso: 10 9 2018]

7. Puppet, «Puppet» [En línea]. Avaible: https://puppet.com. [Último acceso: 29 9 2018]

8. CentOS, «What is Swap Space?» [En línea]. Avaible: https://www.centos.org/docs/5/html/5.2/Deployment_Guide/s1-swap-what-is.html. [Último acceso: 25 6 2018]

9. RDO, «Packstack: Create a proof of concept cloud» [En línea]. Avaible: https://www.rdoproject.org/install/packstack/. [Último acceso: 2 7 2018]

10. OpenStack, «Command List» [En línea]. Avaible: https://docs.openstack.org/python-openstackclient/queens/cli/command-list.html. [Último acceso: 15 8 2018]

11. Anthony, «Condition in bash scripting (if statements)» [En línea]. Avaible: https://linuxacademy.com/blog/linux/conditions-in-bash-scripting-if-statements/. [Último acceso: 10 07 2018]

12. Rahul K., «How to Install PHP 7.2, 7.1 on CentOS/RHEL 7.5 & 6.9» [En línea]. Avaible: https://tecadmin.net/install-php-7-on-centos/. [Último acceso: 24 10 2018]

13. Stackoverflow, «Uncaught SoapFault exception: [HTTP] Error Fetching http headers» [En línea]. Avaible: https://stackoverflow.com/questions/920646/uncaught-soapfault-exception-http-error-fetching-http-headers. [Último acceso: 16 10 2018]

14. Stackoverflow, «Run Bash Command form PHP» [En línea]. Avaible: https://stackoverflow.com/questions/11052162/run-bash-command-from-php. [Último acceso: 24 9 2018]

15. umidjons, «Simple Web service – SOAP Server/Client in PHP» [En línea]. Avaible: https://gist.github.com/umidjons/f3de2533c51495a9c557. [Último acceso: 11 9 2018]

Anexo 1: Instalación de CentOS

CentOS (Community ENTerprise Operating System) es un sistema operativo Linux que proviene de la distribución Red Hat Enterprise Linux (RHEL). Red Hat se compone de software libre y código abierto, y publican parte del código fuente bajo la licencia GNU General Public License (GPL). A partir de dicho código fuente, CentOS compila y distribuye el sistema operativo como software libre también bajo la licencia GPL. En 2014, CentOS ingreso en Red Hat siendo independiente de RHEL. La versión de CentOS que se va a instalar es la 7 minimal sobre VirtualBox.

El primer paso es crear la red de anfitrión, para ello vamos a "Archivo > Administrador de red anfitrión…". Se nos abrirá una nueva pantalla y le damos clic en crear. Se nos creara la nueva interfaz de red y la configuramos como se muestra en la Figura 22.

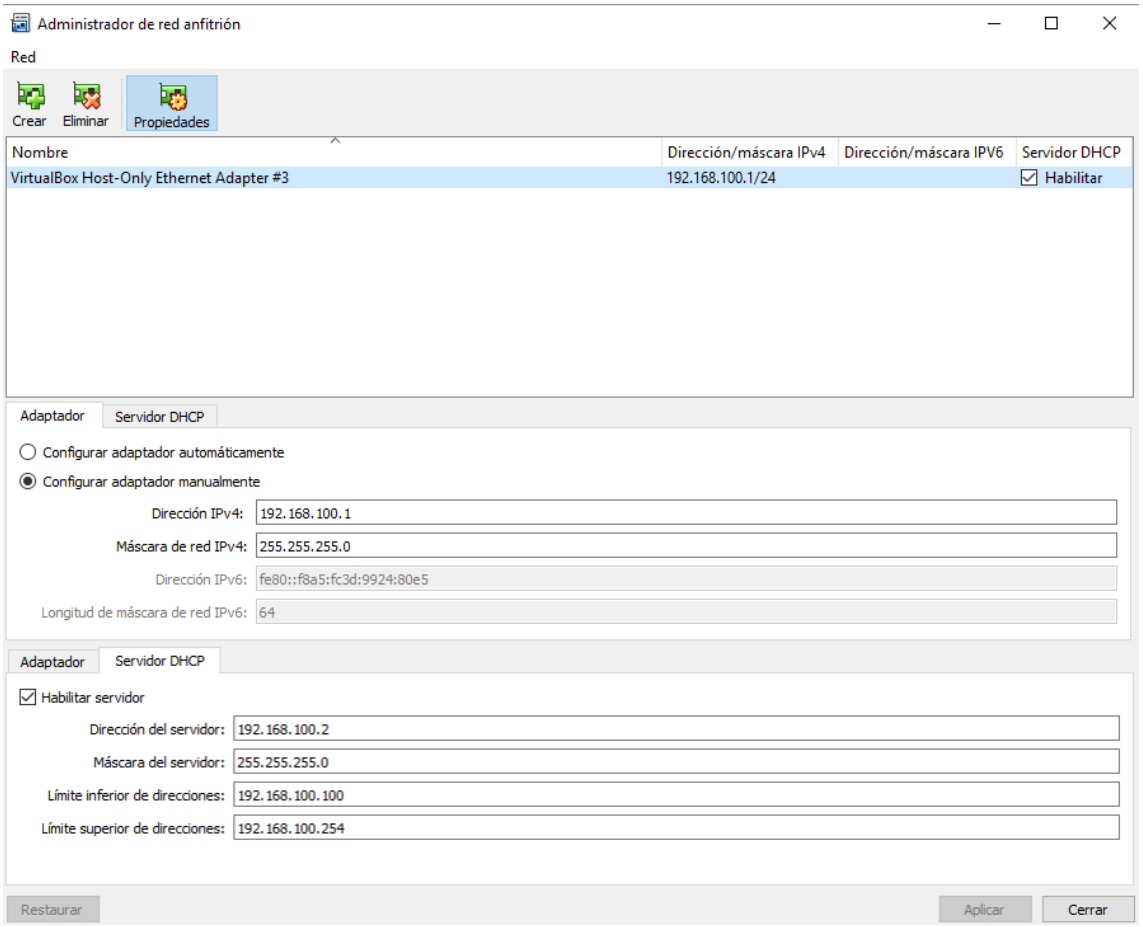


Figura 22 - Administrador de red anfitrion

A continuación, procedemos a crear la máquina virtual gestionada por VirtualBox. Para ellos vamos a "Máquina > Nueva" y le asignamos los siguientes valores:

- Nombre: centos7-openstack
- Tipo: Linux
- Versión: Red Hat 64b
- Memoria: 5120 MB
- Disco duro: 50 GB

Una vez creada la máquina virtual la seleccionamos vamos a "Maquina > Configuración", para modificar los siguientes apartados:

- En "Sistema > Placa Base" seleccionamos la opción "Habilitar I/O APIC"
- En "Sistema > Procesador" seleccionamos la opción "Habilitar PAE/NX"
- En "Almacenamiento" indicamos la ubicación de nuestra imagen para la instalación de CentOS.

- En “Red > Adaptador 1” lo habilitamos y conectado a “NAT”.
- En “Red > Adaptador 2” lo habilitamos, conectado a “Adaptador sólo-anfitrión” y seleccionamos la interfaz configurada anteriormente.

Con esto acabos la preparación de la máquina virtual. Al arranca la maquina se nos mostrara una pantalla como la de la Figura 6 . Se selecciona la opción “Install CentOS Linux 7 como se muestra en la figura 23.

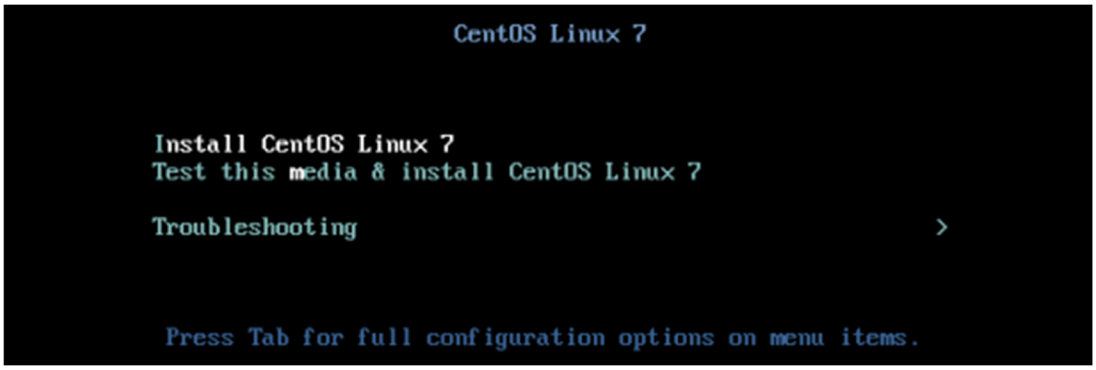


Figura 23 - Pantalla instalación de CentOS 7

Escogemos el idioma de la instalación. A continuación, modificamos los parámetros como la zona horaria y el idioma del teclado.

Nos dirigimos a “Destino de la instalación” y en particionado seleccionamos la opción “Voy a configurar las particiones”. Esto se debe hacer ya que si se dejase el particionado por defecto le asigna la mayor parte del disco a la partición “/home”. El problema es que PackStack se instala desde la partición de root y OpenStack solo podrá utilizar el espacio reservado en dicha partición. Para ello vamos a modificar el esquema de particionado, como se muestra en la Figura 24. El tamaño de Swap se puede calcular como:

```
RAM < 2 GB
    Swap = RAM * 2

RAM > 2 GB
    Swap = RAM + 2
```

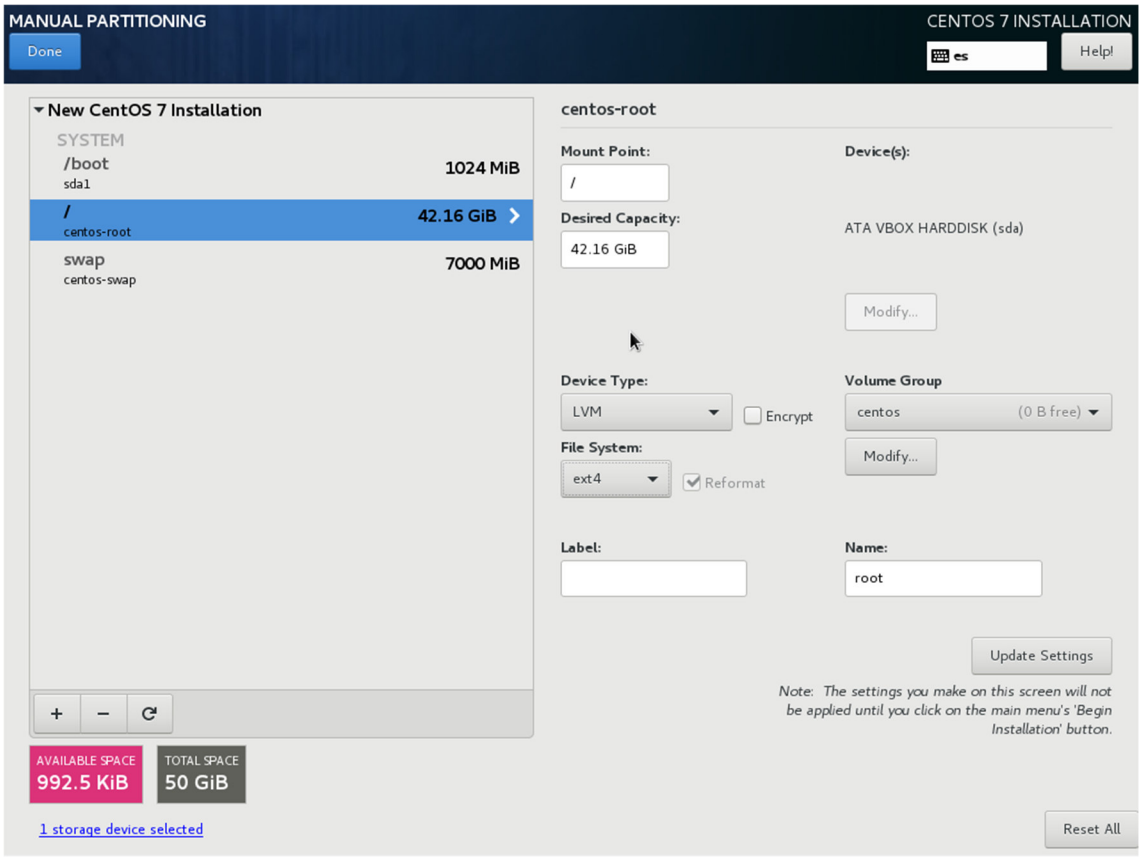


Figura 24 - Particionado del disco

Se asigna un nombre al host, en este caso se ha puesto “openstack” y hacer clic en “Aplicar” para que el cambio surta efecto. Además, se activan y configuran las interfaces de red como IPs estáticas como se ve en la figura 25.

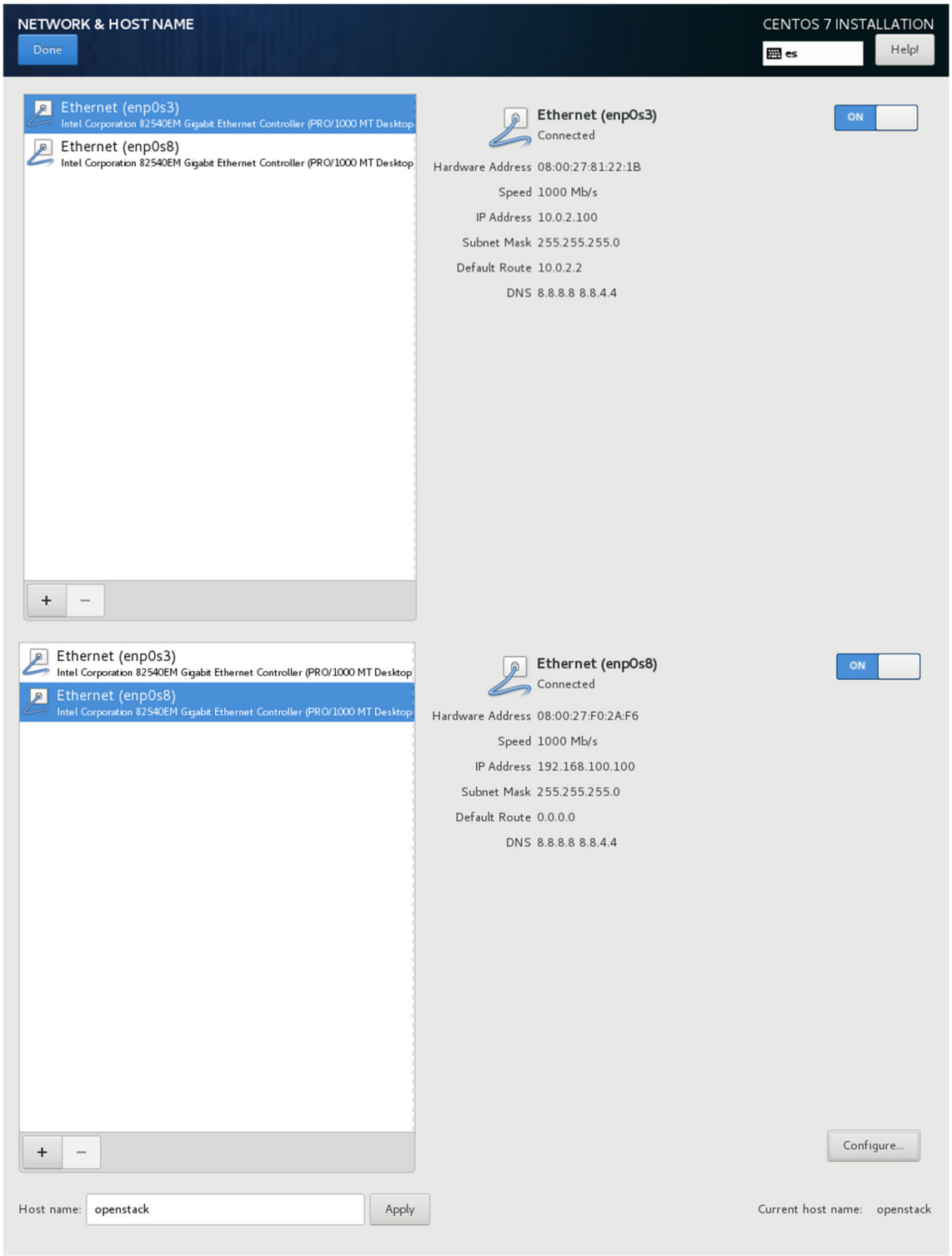


Figura 25 - Configuración de la red

Con todo configurado pulsar el botón “Iniciar instalación”. Durante el proceso de instalación se mostrará una pantalla como la de la figura 26 donde establecemos la contraseña root y podemos crear un nuevo usuario. Es recomendable por cuestiones de seguridad la creación de un nuevo usuario.

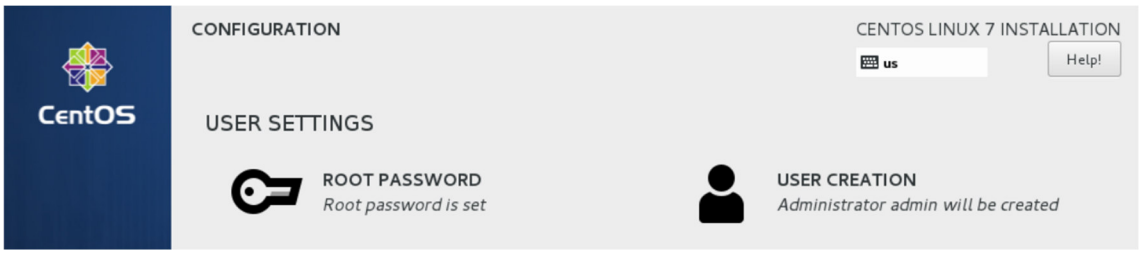


Figura 26 - Configuración de usuarios