Implantación de una nube privada con OpenStack en el I.E.S. Celia Viñas (Almería)

Departamento de Informática del I.E.S. Celia Viñas

Versión 1.0, 2022-05-05

Tabla de contenidos

Agradecimientos

Gracias a los profesores del I.E.S. Gonzalo Nazareno de Sevilla, **Alberto Molina Coballes** y **José Domingo Muñoz Rodríguez**, por compartir con nosotros su conocimiento y experiencia con OpenStack, y ofrecernos su ayuda siempre que los hemos necesitado.

Gracias a los profesores de la Universidad de Almería, **José Antonio Martínez García** y **Manuel Torres Gil**, por habernos guiado durante el proceso de instalación y habernos ayudado a resolver todos los problemas que han ido apareciendo a lo largo de esta aventura.

Gracias a los alumnos de 2º ASIR del I.E.S. Celia Viñas de Almería, **David Escoriza Martínez** y **Eduardo Saracho Cruz**, por colaborar y ayudar en la puesta en producción de este proyecto.

Muchas gracias a todos por vuestra ayuda.

1. Introducción

1.1. Breve descripción del proyecto

Este documento hace una breve descripción de los pasos que se han llevado a cabo para realizar la implantación de una nube privada con **OpenStack** en el **I.E.S. Celia Viñas (Almería)** durante el **curso 2021/2022**.

El objetivo de este documento es **compartir todo lo que hemos aprendido durante este proceso** y pueda servir de guía en futuras instalaciones o actualizaciones que se realicen en el centro.

La instalación de OpenStack se ha realizado en dos entornos físicos, un **entorno de desarrollo** y un **entorno de producción**. A lo largo de este documento se describen todos los pasos que se han llevado a cabo en cada uno de ellos.

El método que se ha utilizado para realizar la instalación y configuración del entorno OpenStack ha sido **OpenStack Ansible**. El sistema operativo utilizado en los nodos ha sido **Ubuntu Server 20.04 LTS (Focal Fossa)** y la versión de OpenStack elegida ha sido **Xena** (`24.0.0`), liberada el 10 de diciembre de 2021.

1.2. ¿Qué es OpenStack?

OpenStack es un proyecto open source que permite gestionar recursos virtuales de computación, redes, almacenamiento e imágenes, para diseñar y gestionar nubes privadas y públicas. También se puede definir como una plataforma de cloud computing que proporciona infraestructura como servicio o IaaS (*Infraestructure as a Service*).

1.3. Componentes de OpenStack

Los componentes básicos de OpenStack con los que vamos a trabaja en este proyecto son los siguientes:

- **Nova:** Gestiona la virtualización de los recursos de cómputo.
- **Neutron:** Gestiona los virtualización de los recursos de red.
- **Glance:** Se encarga de la gestión de las imágenes.
- **Cinder:** Gestiona el almacenamiento a nivel de bloques.
- **Horizon:** Nos permite interactuar con OpeStack a través de una interfaz web.
- **Keystone:** Se encarga de la gestión de identidades y acceso.
- **Swift:** Gestiona el almacenamiento a nivel de objetos.

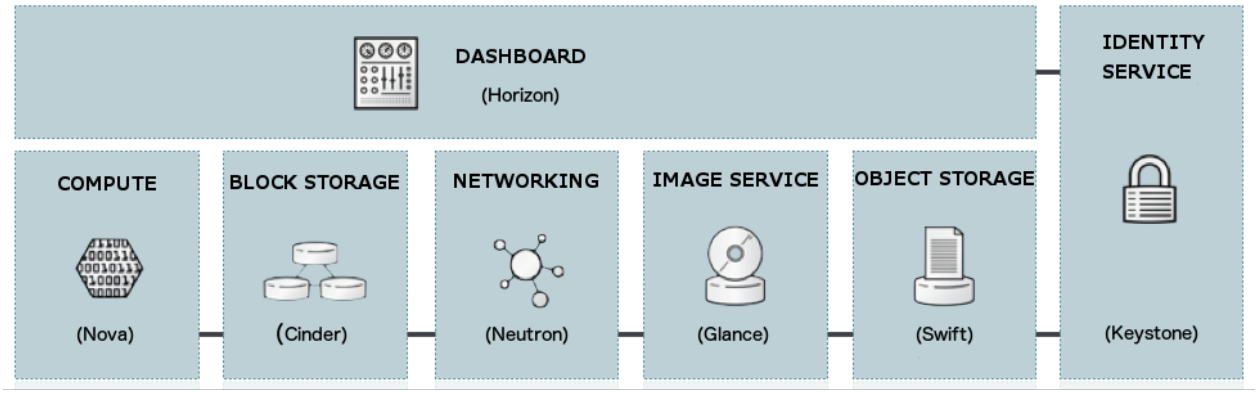

Figura 1. Componentes básicos de OpenStack. Imagen obtenida de la web oficial de OpenStack.

Referencias:

- Listado completo de todos los servicios de OpenStack
- Introduction to OpenStack

1.4. Arquitectura lógica de OpenStack

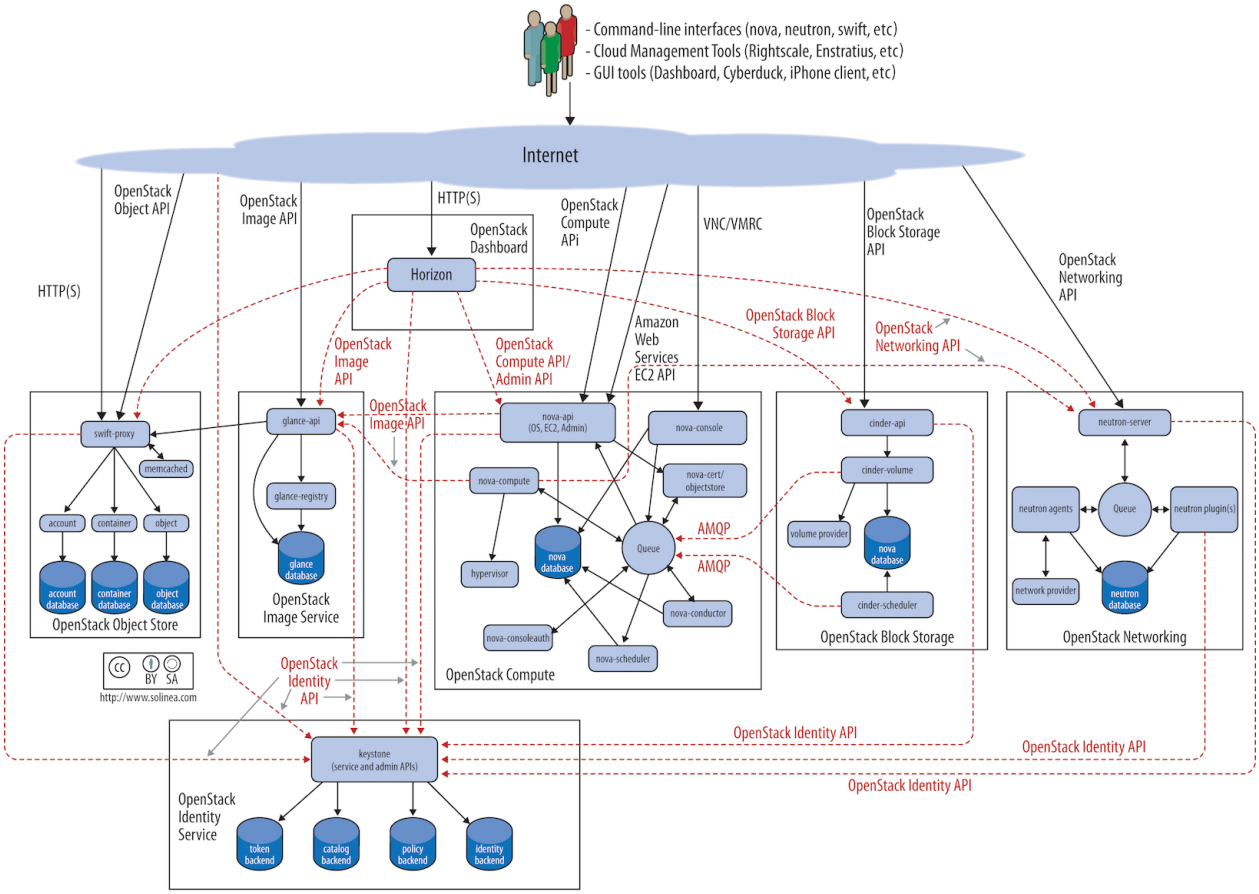

Figura 2. Arquitectura lógica de OpenStack. Imagen obtenida de la web oficial de OpenStack.

Referencias:

- Design an OpenStack Cloud

2. Infraestructura del IES Celia Viñas

En este apartado se describen las características del entorno de desarrollo y producción donde se ha realizado la instalación de OpenStack.

2.1. Entorno de desarrollo (Departamento)

Las máquinas que hemos utilizado en el entorno de desarrollo son antiguos equipos que se utilizaban en las aulas del centro. Son equipos con modestas prestaciones que nos permiten realizar una prueba de concepto de OpenStack.

2.1.1. Hardware

2.1.1.1. Nodo bastión

- Intel® Pentium® CPU G3220 @ 3.00GHz, 2 núcleos (1 Thread por core) = 2 CPUs
- Memoria: 4 GB de RAM
- Disco SSD 128 GB

2.1.1.2. Nodo de control

- Intel® Core™ i7 CPU 920 @ 2.67GHz, 4 núcleos (2 Threads por core) = 8 CPUs
- Memoria: 8 GB de RAM
- Disco SSD 128 GB

El nodo de control debe tener al menos **100 GB de disco**.

2.1.1.3. Nodo de cómputo

- Intel® Core™ i7 CPU 920 @ 2.67GHz, 4 núcleos (2 Threads por core) = 8 CPUs
- Memoria: 8 GB de RAM
- Disco SSD 128 GB

2.1.1.4. Nodo de almacenamiento

- Intel® Pentium® CPU G3220 @ 3.00GHz, 2 núcleos (1 Thread por core) = 2 CPUs
- Memoria: 4 GB de RAM
- 2 x Disco SSD 128 GB

2.1.1.5. Switch

- Modelo: Cisco Catalyst 2960-48TT-L
- Switch gestionable de Capa 2.
- 48 Puertos 10/100 BASE-T Ethernet.
- Fecha de lanzamiento: 18/09/2005
- Fecha de fin de soporte: 31/10/2019

Referencia:

- Switch Cisco Catalyst 2960-48TT-L

2.1.2. Arquitectura de red

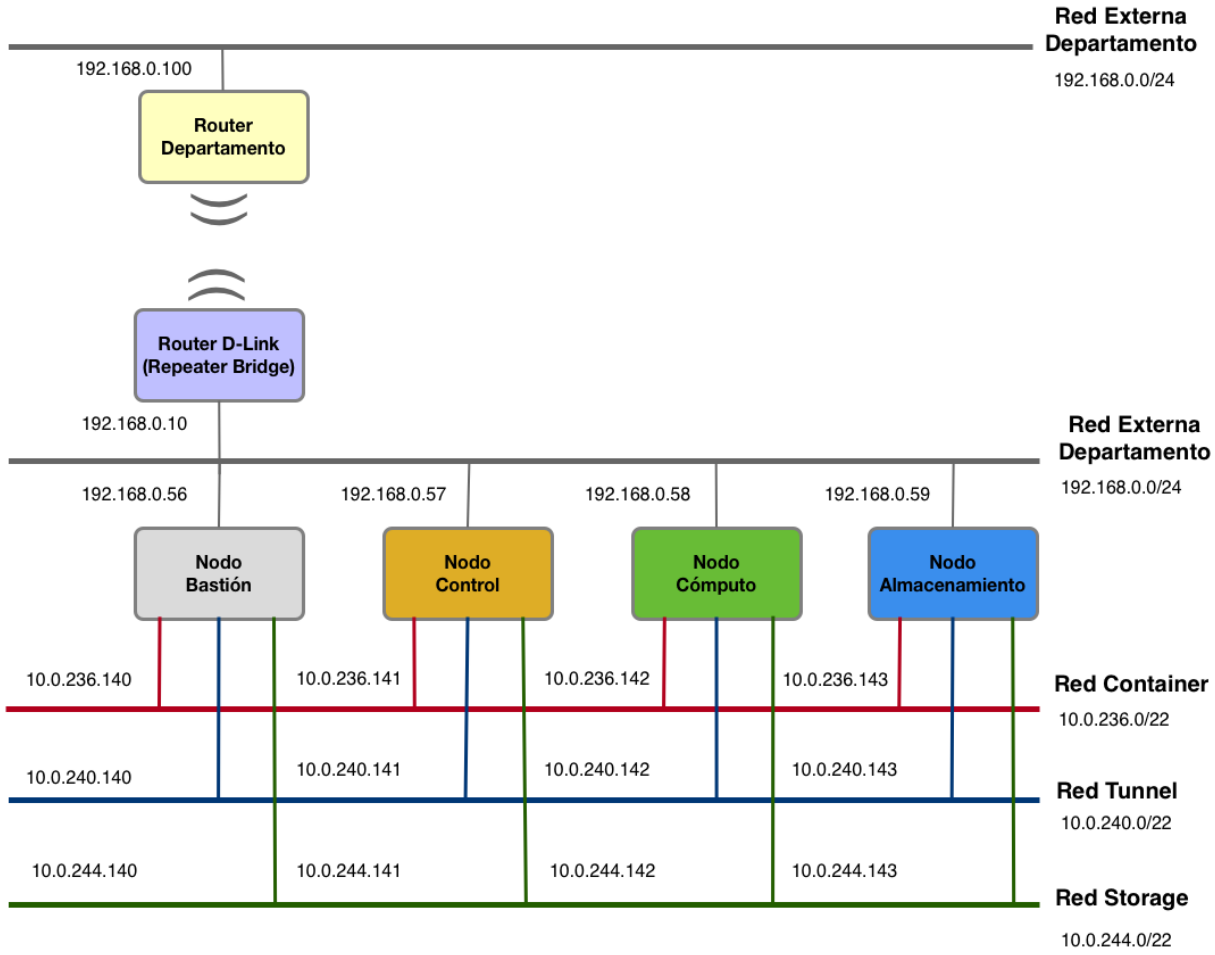

Figura 3. Arquitectura de red del entorno de desarrollo.

Observaciones

La red externa de OpenStack del entorno de desarrollo es la misma red que utilizan los equipos del departamento, es una **red de clase B** (`192.168.0.0/24`) que nos permite

direccionar **254 hosts**. Hay que tener en cuenta que en este entorno sólo podremos disponer de un pool reducido de direciones IPs flotantes para las máquinas virtuales creadas en OpenStack.

2.1.3. Tabla resumen con las direcciones IP de la red externa

Tabla 1. Direcciones IP utilizadas por los nodos y el switch en la red externa

Nodo	IP red externa
Bastión	192.168.0.56
Control	192.168.0.57
Cómputo	192.168.0.58
Almacenamiento	192.168.0.59
Switch Cisco Catalyst 2960	192.168.0.55

2.2. Entorno de producción (Aula Ateca)

2.2.1. Hardware

2.2.1.1. Nodo bastión

- Procesador: Intel® Pentium® CPU G3220 @ 3.00GHz, 2 núcleos (1 Thread por core) = 2 CPUs
- Memoria: 4 GB de RAM
- Disco SSD 128 GB

El nodo bastión es un antiguo equipo que se utilizaba en las aulas del centro. Este equipo solo se utiliza para realizar la instalación de OpenStack, así que no es necesario que sea un equipo con grandes prestaciones.

2.2.1.2. Nodo de control

- Dell PowerEdge R240
- Servidor Dell EMC PowerEdge de 14ª generación.
- Intel Xeon E-2224, 3,4 GHz, 4 núcleos (1 Thread por core) = 4 CPUs
- 32 GB de RAM (4 x 8 GB DDR4 UDIMM 3200MHz ECC)
- Sin controladora RAID
- Disco duro 1 TB

Sería ideal contar con **64 GB de RAM** en el nodo de control.

Nota sobre la memoria RAM soportada en el servidor Dell R240

La **capacidad máxima de RAM** que puede tener el servidor Dell PowerEdge R240 es de **64 GB**.

Hay que tener en cuenta que el tipo de memoria RAM que utiliza es **UDIMM** y que sólo dispone de 4 ranuras DIMM DDR4.

- Guía de configuración de la memoria soportada para los servidores PowerEdge

Referencia:

- Servidor para rack Dell PowerEdge R240

2.2.1.3. Nodo de cómputo

- Dell PowerEdge R440
- Servidor Dell EMC PowerEdge de 14ª generación.
- 2 x Intel Xeon Silver 4214, 2,2 GHz, 12 núcleos (2 Threads por core) = 48 CPUs
- 256 GB de RAM (4 x RDIMM de 64 GB, 3200 MT/s)
- Controladora RAID PERC H730P+
- Disco duro conectable en caliente de 2,5" 512n, 1,2 TB a 10K rpm, SAS a 12 Gbps

Referencia:

- Servidor para rack Dell PowerEdge R440

2.2.1.4. Nodo de almacenamiento

- Dell PowerEdge R640
- Servidor Dell EMC PowerEdge de 14ª generación.
- Intel Xeon Silver 4210R, 2,4 GHz 10 núcleos (2 Threads por core) = 20 CPUs
- 16 GB de RAM
- Controladora RAID PERC H730P
- Disco duro 480GB
- 4 x Discos duros 401-ABHQ 2.4TB 10K rpm SAS 12Gbps 512e

Referencia:

- Servidor para rack Dell PowerEdge R640

Convención de nomenclatura de los servidores Dell PowerEdge

Los nombres de los servidores Dell PowerEdge utilizan una letra seguida de tres o cuatro dígitos. Vamos a analizar lo que significa cada valor para los nombres **R240**, **R440** y **R640**.

La letra que aparece al inicio del nombre indica el factor de forma del servidor, es decir, si se trata de un servidor en torre, montable en rack, blade, etc.

- **R**: Servidores montables en rack.

Los tres números siguientes indican el número de CPUs, la generación y el fabricante de la CPU.

- El primer número indica el número de CPUs que tiene servidor.
  - **1 a 3**: Servidores con 1 CPU.
  - **4 a 7**: Servidores con 2 CPUs.
  - **8**: Servidores con 2 o 4 CPUs.
  - **9**: Servidores con 4 CPUs.
- El segundo número indica la generación del servidor.
  - **0**: Servidores de 10ª generación.
  - **1**: Servidores de 11ª generación.
  - **2**: Servidores de 12ª generación.
  - **3**: Servidores de 13ª generación.
  - **4**: Servidores de 14ª generación y así sucesivamente.
- El tercer número indica el fabricante de la CPU.
  - **0**: Intel.
  - **5**: AMD.

Por lo tanto, si analizamos el nombre de nuestros servidores obtenemos lo siguiente:

- El modelo **R240**:
  - R: es un servidor montable en rack.
  - 2: de 1 CPU.
  - 4: de la 14ª generación de servidores.
  - 0: con procesadores Intel.
- El modelo **R440**:

    - R: es un servidor montable en rack.
    - 4: de 2 CPUs.
    - 4: de la 14ª generación de servidores.
    - 0: con procesadores Intel.
- El modelo **R640**:
    - R: es un servidor montable en rack.
    - 6: de 2 CPUs.
    - 4: de la 14ª generación de servidores.
    - 0: con procesadores Intel.

Referencia:

- Cómo identificar la generación de servidores Dell EMC PowerEdge

2.2.1.5. Switch

- Modelo: D-Link DGS 1210-26.
- Switch gestionable de Capa 2.
- 24 Puertos 10/100/1000 BASE-T Gigabit Ethernet.

Referencia:

- Smart+ Managed Gigabit Switches DGS-1210 Series

2.2.2. Arquitectura de red

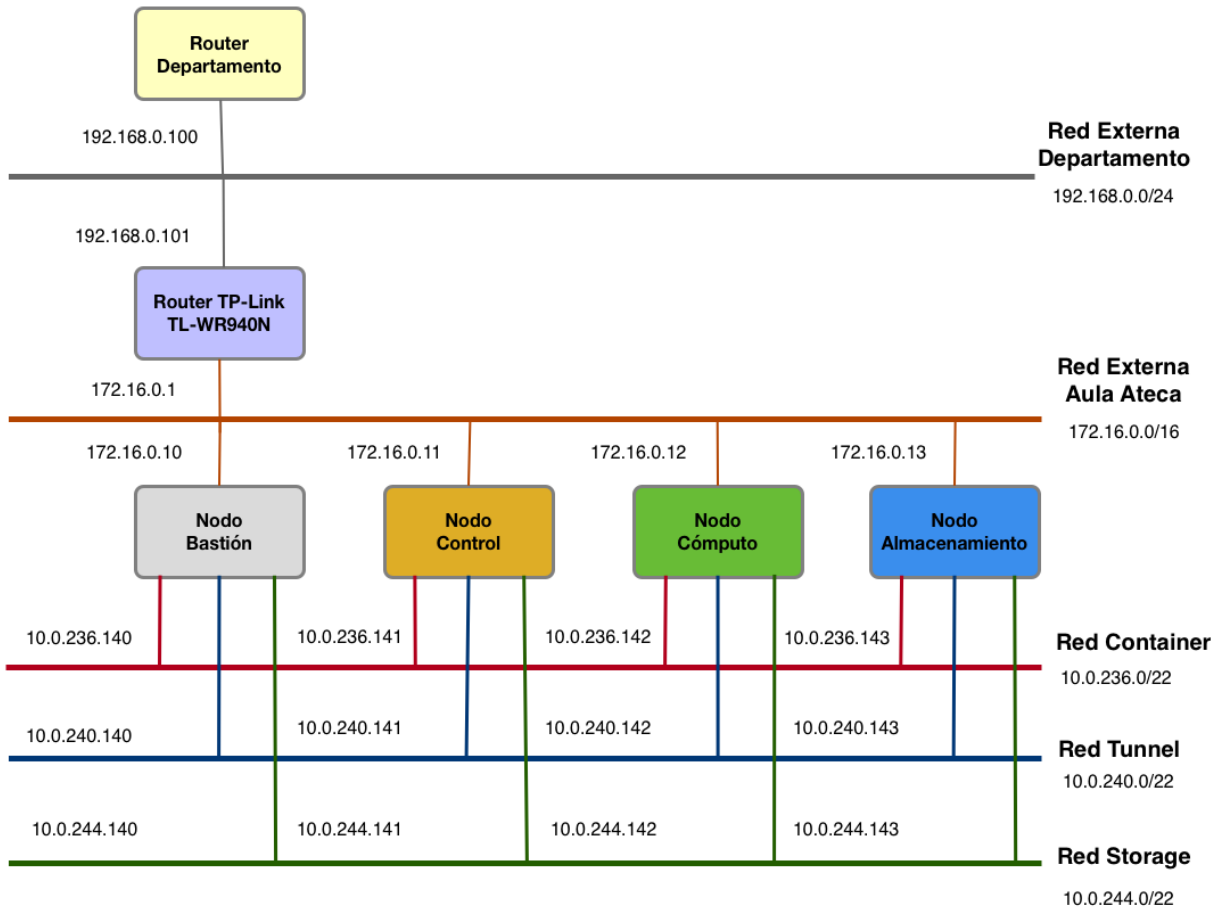

Figura 4. Arquitectura de red del entorno de producción.

Observaciones

- En la red externa de OpenStack hemos utilizado una **red de clase B** (`172.16.0.0/16`) ya que nos permite direccionar **65534 hosts**, en lugar de utilizar una red de clase C que sólo nos permitiría 254 hosts.
- También hemos tenido que **modificar la tabla de rutas del router del departamento** para que los equipos de las aulas puedan acceder a la red de OpenStack.
- En el router del aula Ateca (TP-Link TL-WR940N) hemos tenido que **deshabilitar el filtrado de paquetes** en el apartado de seguridad para permitir el tráfico entrante de las otras redes del centro.

2.2.3. Tabla resumen con las direcciones IP de la red externa

Tabla 2. Direcciones IP utilizadas por los nodos y el switch en la red externa

Nodo	IP red externa	IP iDRAC
Bastión	172.16.0.10	
Control	172.16.0.11	https://172.16.0.2
Cómputo	172.16.0.12	https://172.16.0.3
Almacenamiento	172.16.0.13	https://172.16.0.4
Switch D-Link DGS 1210-26	172.16.0.5	

3. Pasos previos a la instalación del Sistema Operativo en los servidores Dell del entorno de producción

Los pasos que se describen en este apartado solo se realizan en los nodos de **Control**, **Cómputo** y **Almacenamiento** del entorno de producción.

Antes de realizar la instalación del Sistema Operativo en los servidores Dell hemos realizado algunas operaciones desde la utilidad **Lifecycle Controller**. Para acceder a la utilidad Lifecycle Controller hay que pulsar la tecla **F10** durante el proceso POST (*Power On Self Test*) al iniciar el servidor.

Las operaciones que hemos realizado desde la utiliad **Lifecycle Controller** son las siguientes:

- Configuración del idioma y tipo de teclado.
- Configuración de la red.
- Actualización del firmware.
- Configuración de red para el acceso por iDRAC.
- Configuración del usuario y contraseña para el acceso por iDRAC.
- Configuración del RAID de discos a nivel hardware.

¿Qué es Lifecycle Controller?

Lifecycle Controller es una utilidad de incorporan los servidores Dell que permite realizar tareas de aministración y configuración sobre éstos. Esta utilidad dispone de una interfaz de usuario, se encuentra integrada en una tarjeta de memoria flash del servidor y se puede iniciar durante la secuencia de inicio del servidor, antes de iniciar el sistema operativo.

Referencia:

- Guía del usuario de Lifecycle Controller

3.1. Configuración del idioma y tipo de teclado

Una vez que hemos iniciado la utilidad Lifecycle Controller la primera operación que vamos a realizar es la configuración del idioma y el tipo de teclado.

Esta operación se realiza seleccionando la opción **Settings** y **Language and Keyboard**.

3.2. Configuración de la red

El siguiente paso es la configuración de la red externa que tendrán los equipos. Esta configuración no tiene por qué ser la configuración definitiva que tendrán los servidores. En este paso podemos obtener una dirección IP por DHCP y modificarla más adelante cuando se realice la instalación del sistema operativo.

3.3. Actualización del firmware

Una vez que hemos configurado la red podemos realizar la actualización del firmware del servidor mediante el asistente. De todos los métodos que aparecen hemos seleccionado la actualización mediante el **Sitio web de Dell**.

3.4. Configuración de red para el acceso por iDRAC

El siguiente paso es la configuración de red del interfaz iDRAC. Desde iDRAC podemos conectarnos de forma remota al servidor para encenderlo/apagarlo, configurarlo, etc.

Las direcciones IP estáticas que les hemos asiganado a cada uno de los servidores para el acceso por iDRAC son las siguientes.

Tabla 3. Direcciones IP utilizadas por los nodos para el acceso por iDRAC

Nodo	IP iDRAC	Versión iDRAC
Control	https://172.16.0.2	iDRAC 9 Express
Cómputo	https://172.16.0.3	iDRAC 9 Express
Almacenamiento	https://172.16.0.4	iDRAC 9 Enterprise

¿Qué es iDRAC?

iDRAC son las siglas de *Integrated Dell Remote Access Controller*. Los servidores Dell utilizan esta tecnología para poder administrar, configurar y gestionar los servidores de forma remota a través de una interfaz web o por SSH.

Referencia:

- Integrated Dell Remote Access Controller

3.5. Configuración del RAID de discos a nivel hardware.

El nodo de **Cómputo** y **Almacenamiento** disponen de una **controladora RAID** que tenemos que configurar antes de realizar la instalación del sistema operativo.

3.5.1. Nodo de cómputo

El nodo de cómputo cuenta con la controladora:

- RAID PERC H730P+.

Y un solo disco:

- Disco duro conectable en caliente de 2,5" 512n, 1,2 TB a 10K rpm, SAS a 12 Gbps.

Como solo disponemos de un disco lo hemos configurado como **RAID 0** desde la utilidad LifeCycle Controller. Para crear el RAID 0, en primer lugar hay que crear un disco virtual y luego se le añade el disco físico. Las capturas de pantalla que se muestran a continuación se han realizado desde la interfaz web de iDRAC.

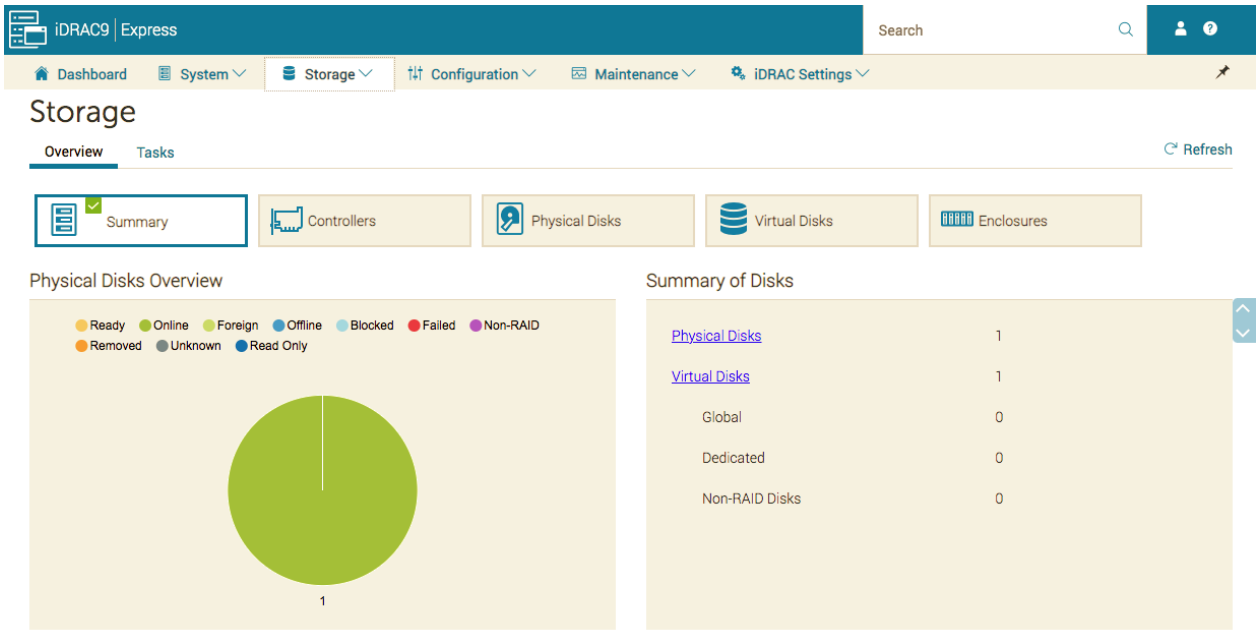

Figura 5. Discos físicos y discos virtuales del nodo de Cómputo.

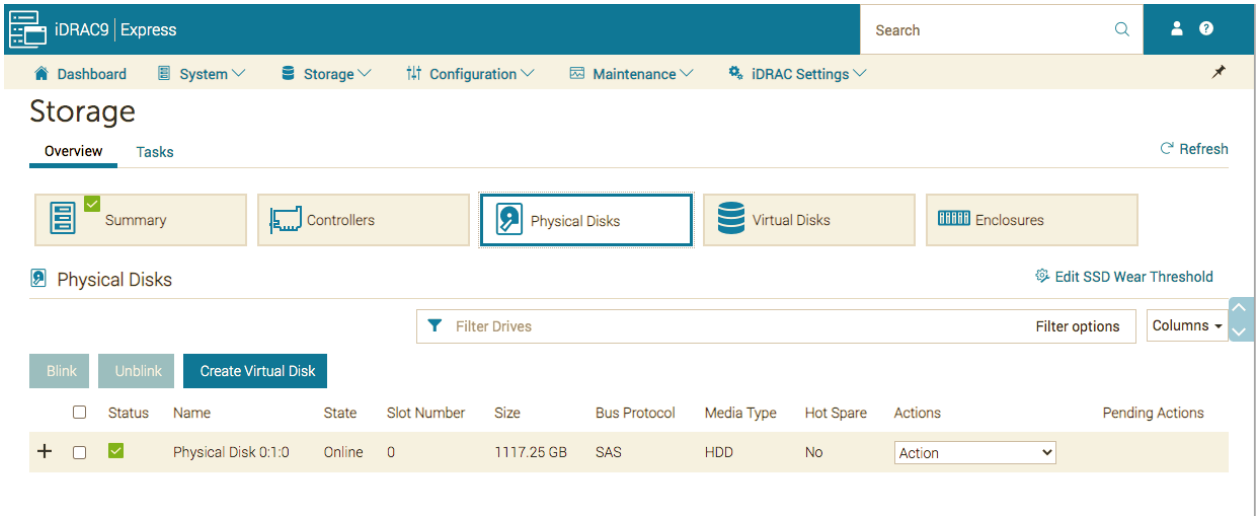

Figura 6. Disco físico del nodo de Cómputo.

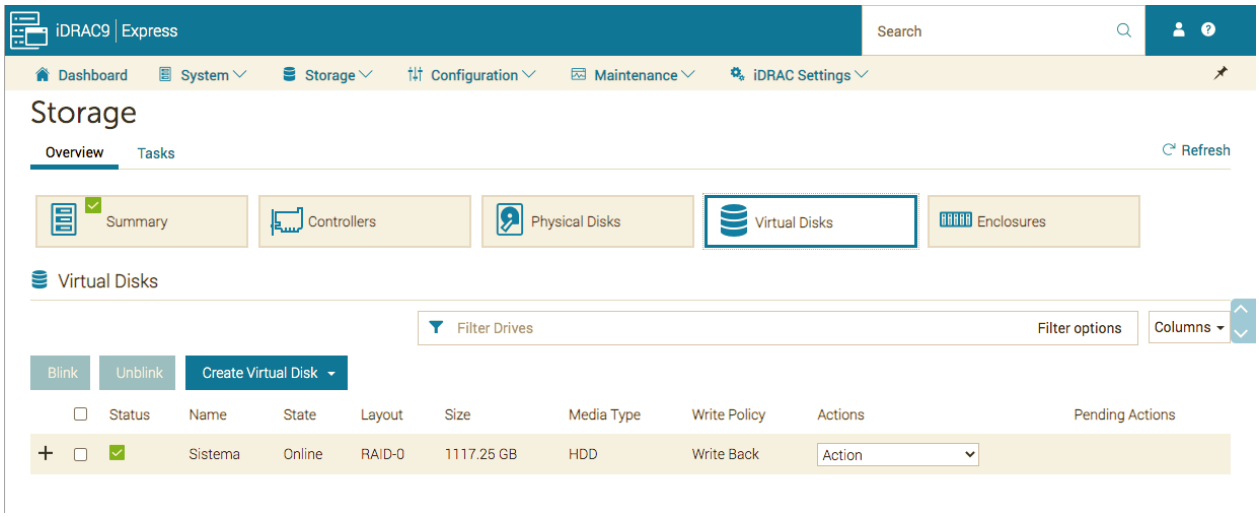

Figura 7. Disco virtual del nodo de Cómputo.

3.5.2. Nodo de almacenamiento

El nodo de almacenamiento cuenta con la controladora:

- RAID PERC H730P.

Y los siguientes discos:

- 1 Disco duro 480GB.
- 4 x Discos duros 401-ABHQ 2.4TB 10K rpm SAS 12Gbps 512e.

El disco duro de **480 GB** lo hemos configurado como **RAID 0** y lo hemos utilizado para instalar el sistema operativo. Los **4 discos duros de 2.4 TB** los hemos configurado como **RAID 5** y los estamos utilizando almacenar los volúmenes del servicio **Cinder** de OpenStack.

El orden en el que se crean los RAID de discos es importante ya que el nombre con el que

aparecerán en el sistema operativo dependerá del orden en el que se han creado. En primer lugar hemos creado el RAID 0 que aparecerá con el nombre `/dev/sda` y en segundo lugar el RAID 5 que aparecerá como `/dev/sdb`.

El procedimiento para crear el RAID de discos es igual que en el paso anterior, en primer lugar se crea el disco virtual y luego se añaden los discos físicos que estarán asociados a ese disco virtual.

La configuración del RAID se ha realizado desde la utilidad LifeCycle Controller aunque las capturas de pantalla que se muestran a continuación se han realizado desde la interfaz web de iDRAC.

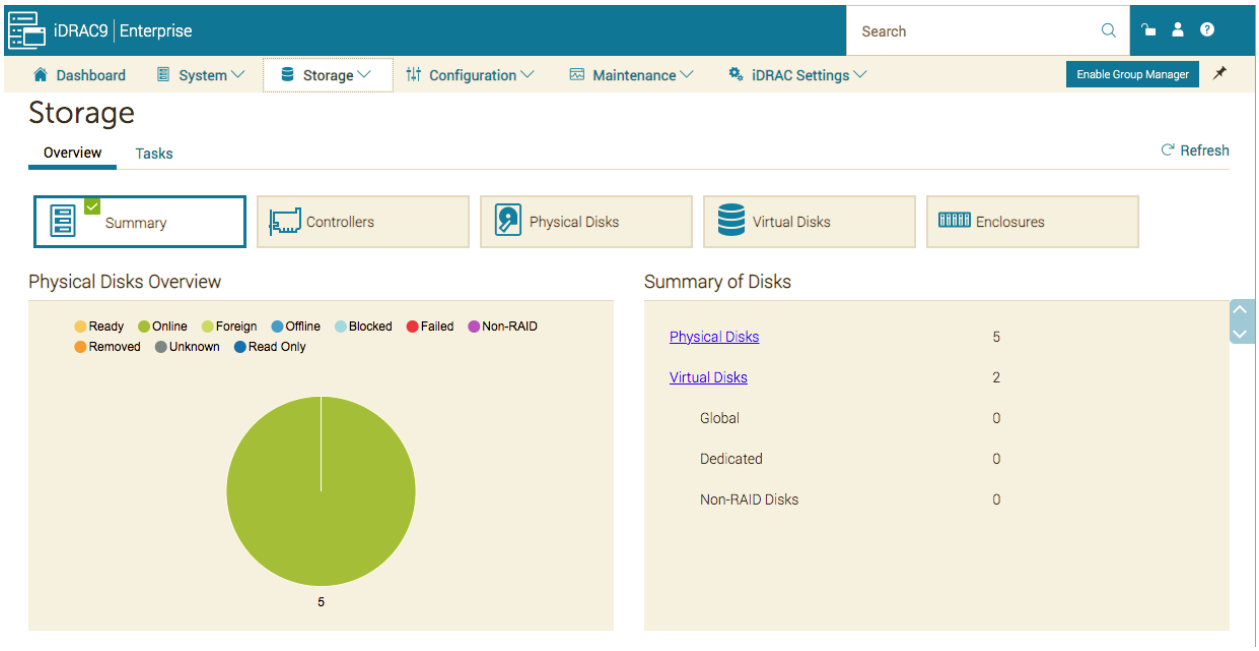

Figura 8. Discos físicos y discos virtuales del nodo de Almacenamiento.

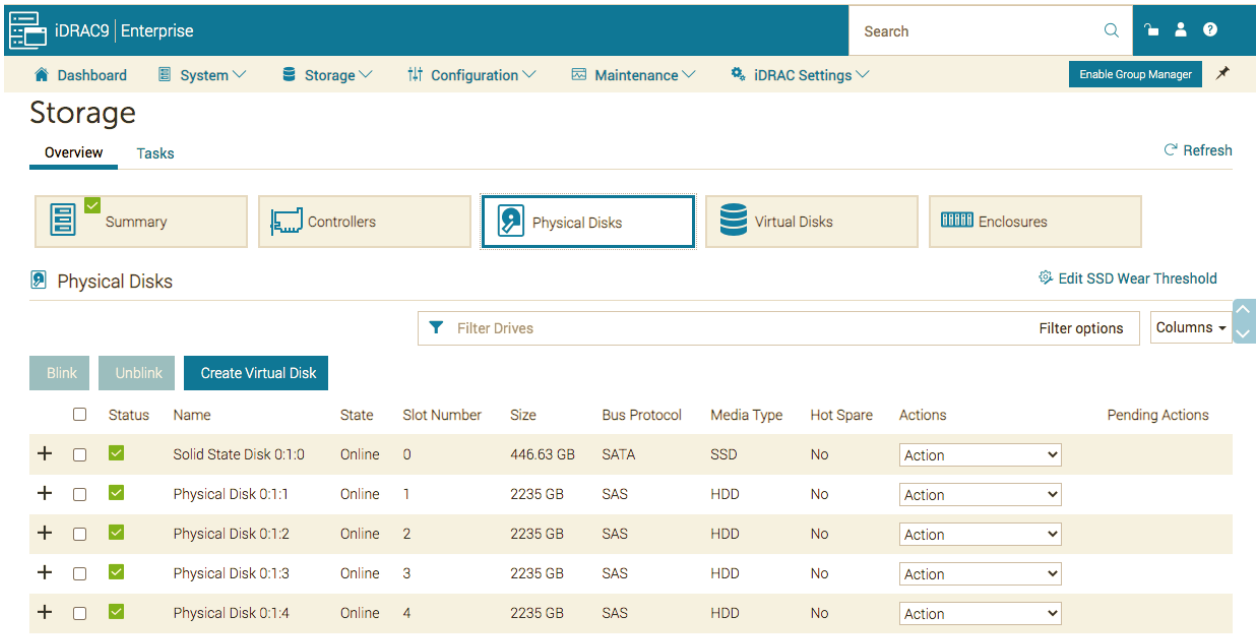

Figura 9. Discos físicos del nodo de Almacenamiento.

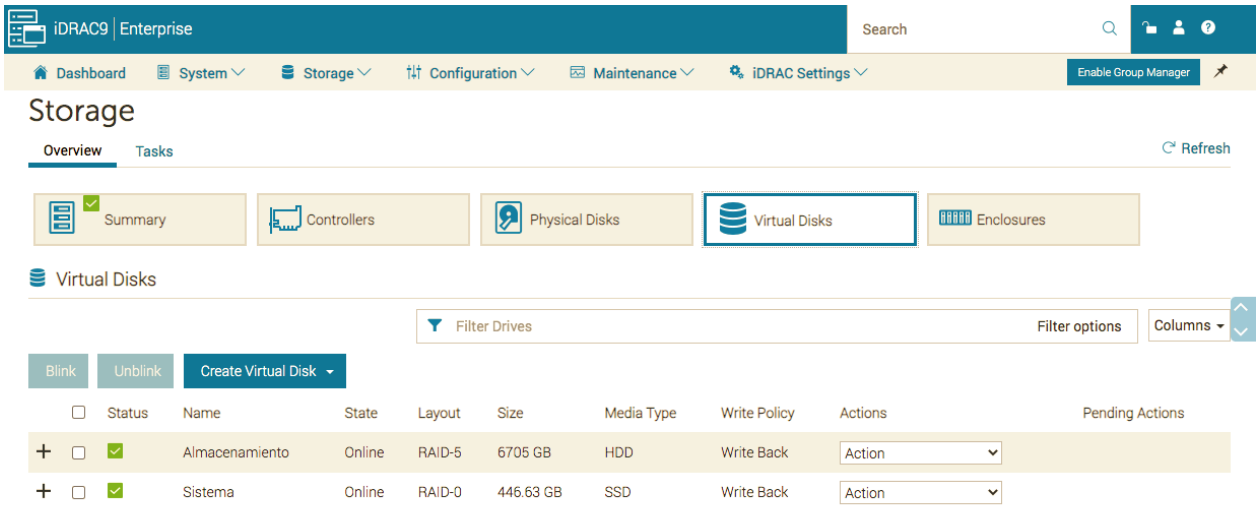

Figura 10. Discos virtuales del nodo de Almacenamiento.

4. Instalación y configuración del Sistema Operativo Ubuntu Server 20.04 LTS en los servidores Dell del entorno de producción

El sistema operativo que hemos instalado en los nodos ha sido **Ubuntu Server 20.04 LTS (Focal Fossa)**, ya que era uno de los sistemas operativos recomendados por la versión de OpenStack **Xena** (`24.0.0`), que es la que hemos utilizado en la implantación del cloud del centro.

4.1. Configuración de las tarjetas de red en modo *bonding*

Durante la instalación de Ubuntu Server 20.04 el asistente nos ofrece la posibilidad de configurar las tarjetas de red en modo *bonding*.

La configuración que vamos a utilizar para configurar las tarjetas en modo *bonding* es la siguiente:

- Nombre: `bond0`
- Bond mode: `802.3ad`
- XMIT hash policy: `layer3+4`
- LACTP rate: `fast`

El *bonding* de los nodos de **Control** y **Cómputo** estará formado por las interfaces de red `eno1` y `eno2`, mientras que en el nodo de **Almacenamiento** estará formado por las interfaces `eno3` y `eno4`.

Tabla 4. Interfaces de red utilizadas en cada uno de los nodos para crear el bonding

Nodos	Interfaces de red
Control	• `eno1` • `eno2`
Cómputo	• `eno1` • `eno2`

Nodos	Interfaces de red
Almacenamiento	• `eno3` • `eno4`

¿Qué es el bonding de red?

El *bonding de red* consiste en agrupar varias interfaces de red de un mismo equipo para que trabajen de forma conjunta como si fueran una única tarjeta de red. El uso de esta técnica permite repartir el tráfico de red entre las tarjetas que forman el *bonding* y aumentar así la capacidad del enlace, además de garantizar la tolerancia a fallos en el caso de que se produzca un error en alguna de las interfaces de red. A nivel de sistema operativo el *bonding* aparece como una única tarjeta de red.

4.2. Creación de volúmenes LVM

El asistente de instalación de Ubuntu Server 20.04 también nos permite configurar el sistema de archivos como un grupo de volúmenes LVM (*Logical Volume Manager*).

La configuración que hemos seleccionado en este paso ha sido la de utilizar el disco completo y crear un volumen lógico para el directorio raíz del sistema. Hay que tener en cuenta que habrá que modificar el tamaño del volumen lógico que aparece por defecto, ya que el asistente no utiliza todo el espacio disponible del disco.

Las particiones que hemos configurado en cada uno de los nodos son las siguientes:

Tabla 5. Particiones utilizadas en cada uno de los nodos

Partición	Descripción
`/`	Volumen lógico para el directorio raíz del sistema operativo.
`/boot`	Partición para los archivos de arranque.
`/boot/efi`	Partición para los cargadores de arranque EFI, aplicaciones y controladores utilizados por el firmware UEFI.

El nodo de **Almacenamiento** contará además con otro grupo de volúmenes LVM llamadao `cinder-volumes` que será donde se almacenen los volúmenes gestionados por el servicio Cinder.

4.3. Notas sobre la partición `Linux_OEMDRV` que aparece durante la instalación del sistema operativo

Al instalar el sistema operativo se crea automáticamente una partición de 300 MB (aprox.) con los controladores Dell para el sistema operativo seleccionado. Esta partición se elimina automáticamente al pasar un número determinado de horas o se puede eliminar iniciando el servidor en modo Lifecycle controller y reiniciando el servidor.

Referencia:

- Remove OEMDRV Drive from Dell Server

5. Configuración de red

5.1. Entorno de desarrollo (Departamento)

5.1.1. Configuración del switch Cisco Catalyst 2960

Para configurar el switch Cisco Catalyst 2960 seguimos los pasos de la guía oficial para resetear el switch y preparar su configuración.

En el apartado **Ejecución de Express Setup** de la guía oficial encontramos los primeros pasos que tenemos que realizar.

Referencia:

- Guía de inicio del Switch Catalyst 2960-S.

5.1.2. Datos de configuración del switch

- IP estática: `192.168.0.55`

5.1.3. Creación de las VLANs en el switch Cisco Catalyst 2960

En este paso vamos a crear las cuatro VLANs que necesitamos en OpenStack.

Tabla 6. Listado de VLANs de OpenStack

VLAN Id	Nombre
10	vlan10
20	vlan20
30	vlan30
40	vlan40

Es necesario **habilitar el servicio de telnet** para poder configurar el switch mediante la línea de comandos. Una vez que hemos habilitado el servicio de telnet podemos conectarnos con el siguiente comando.

```
# telnet 192.168.0.55
```

Lo primero que haremos será activar el modo de configuración.

```
Switch>enable
Switch#configure terminal
```

Creamos las VLAN que necesitamos. En primer lugar creamos la VLAN `10` y le asignamos el nombre `vlan10`.

```
Switch(config)#vlan 10
Switch(config-vlan)#name vlan10
Switch(config-vlan)#end
```

Creamos la VLAN `20` y le asignamos el nombre `vlan20`.

```
Switch#configure terminal
Switch(config)#vlan 20
Switch(config-vlan)#name vlan20
Switch(config-vlan)#end
```

Creamos la VLAN `30` y le asignamos el nombre `vlan30`.

```
Switch#configure terminal
Switch(config)#vlan 30
Switch(config-vlan)#name vlan30
Switch(config-vlan)#end
```

Creamos la VLAN `40` y le asignamos el nombre `vlan40`.

```
Switch#configure terminal
Switch(config)#vlan 40
Switch(config-vlan)#name vlan40
Switch(config-vlan)#end
```

5.1.4. Configuración de los interfaces en modo trunk

Configuramos el interfaz del switch `Fa0/37` en modo trunk para que pueda trabajar con las VLAN `10`, `20`, `30` y `40`. El interfaz `Fa0/37` se corresponde con la boca `37` del switch.

```
Switch#configure terminal
Switch(config)#interface Fa0/37
Switch(config-if)#switchport mode trunk
Switch(config-if)#switchport access vlan 10
Switch(config-if)#switchport access vlan 20
Switch(config-if)#switchport access vlan 30
Switch(config-if)#switchport access vlan 40
Switch(config-if)#end
```

Configuramos el interfaz del switch `Fa0/39` en modo trunk para que pueda trabajar con las VLAN `10`, `20`, `30` y `40`. El interfaz `Fa0/39` se corresponde con la boca `39` del switch.

```
Switch#configure terminal
Switch(config)#interface Fa0/39
Switch(config-if)#switchport mode trunk
Switch(config-if)#switchport access vlan 10
Switch(config-if)#switchport access vlan 20
Switch(config-if)#switchport access vlan 30
Switch(config-if)#switchport access vlan 40
Switch(config-if)#exit
```

Configuramos el interfaz del switch `Fa0/41` en modo trunk para que pueda trabajar con las VLAN `10`, `20`, `30` y `40`. El interfaz `Fa0/41` se corresponde con la boca `37` del switch.

```
Switch(config)#interface Fa0/41
Switch(config-if)#switchport mode trunk
Switch(config-if)#switchport access vlan 10
Switch(config-if)#switchport access vlan 20
Switch(config-if)#switchport access vlan 30
Switch(config-if)#switchport access vlan 40
Switch(config-if)#exit
```

Configuramos el interfaz del switch `Fa0/43` en modo trunk para que pueda trabajar con las VLAN `10`, `20`, `30` y `40`. El interfaz `Fa0/43` se corresponde con la boca `37` del switch.

```
Switch(config)#interface Fa0/43
Switch(config-if)#switchport mode trunk
Switch(config-if)#switchport access vlan 10
Switch(config-if)#switchport access vlan 20
Switch(config-if)#switchport access vlan 30
Switch(config-if)#switchport access vlan 40
Switch(config-if)#exit
```

Guardamos los cambios.

```
Switch#wr
```

Referencia:

- Cómo configurar VLANs en el switch Cisco Catalyst 2960

5.1.5. Configuración de red en los nodos (Netplan)

La configuración de red de los nodos del entorno de desarrollo va a tener algunas diferencias respecto a los nodos del entorno de producción, ya que los equipos del entorno de desarrollo sólo disponen de una tarjeta de red, mientras que los equipos del entorno de producción disponen de dos tarjetas de red que se configurarán en modo *bonding*.

5.1.5.1. Nodo bastión

```
network:
  version: 2
  renderer: networkd

  ethernets:
    enp2s0: ①
      dhcp4: false

  vlans: ②
     enp2s0.10:
        id: 10
        link: enp2s0

     enp2s0.20:
        id: 20
        link: enp2s0

     enp2s0.30:
        id: 30
        link: enp2s0

     enp2s0.40:
        id: 40
        link: enp2s0

  bridges: ③
    br-mgmt:
       interfaces: [enp2s0.10]
       addresses: [10.0.236.140/22]
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500

    br-storage:
       interfaces: [enp2s0.20]
       addresses: [10.0.244.140/22]
       mtu: 1500

    br-vxlan:
       interfaces: [enp2s0.30]
       addresses: [10.0.240.140/22]
       mtu: 1500

    br-vlan:
       interfaces: [enp2s0.40]
       mtu: 1500

    br-ex:
       interfaces: [enp2s0]
       addresses: [192.168.0.56/24]
       gateway4: 192.168.0.100
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500
       parameters:
          stp: true
          forward-delay: 4
```

① `enp2s0` es el nombre que tiene la interfaz de red en el nodo Bastión. Tenga en cuenta que

el nombre de la interfaz de red puede cambiar en otras máquinas.

② En esta sección definimos las VLANs que vamos a utilizar.

③ En esta sección definimos los *bridges* de red.

5.1.5.2. Nodo de control

```
network:
  version: 2
  renderer: networkd

  ethernets:
    enp6s0:
      dhcp4: false

  vlans:
     enp6s0.10:
        id: 10
        link: enp6s0

     enp6s0.20:
        id: 20
        link: enp6s0

     enp6s0.30:
        id: 30
        link: enp6s0

     enp6s0.40:
        id: 40
        link: enp6s0

  bridges:
    br-mgmt:
       interfaces: [enp6s0.10]
       addresses: [10.0.236.141/22]
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500

    br-storage:
       interfaces: [enp6s0.20]
       addresses: [10.0.244.141/22]
       mtu: 1500

    br-vxlan:
       interfaces: [enp6s0.30]
       addresses: [10.0.240.141/22]
       mtu: 1500

    br-vlan:
       interfaces: [enp6s0.40]
       mtu: 1500

    br-ex:
       interfaces: [enp6s0]
       addresses: [192.168.0.57/24]
       gateway4: 192.168.0.100
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500
       parameters:
          stp: true
          forward-delay: 4
```

5.1.5.3. Nodo de cómputo

```
network:
  version: 2
  renderer: networkd

  ethernets:
    enp6s0:
      dhcp4: false

  vlans:
     enp6s0.10:
        id: 10
        link: enp6s0

     enp6s0.20:
        id: 20
        link: enp6s0

     enp6s0.30:
        id: 30
        link: enp6s0

     enp6s0.40:
        id: 40
        link: enp6s0

  bridges:
    br-mgmt:
       interfaces: [enp6s0.10]
       addresses: [10.0.236.142/22]
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500

    br-storage:
       interfaces: [enp6s0.20]
       addresses: [10.0.244.142/22]
       mtu: 1500

    br-vxlan:
       interfaces: [enp6s0.30]
       addresses: [10.0.240.142/22]
       mtu: 1500

    br-vlan:
       interfaces: [enp6s0.40]
       mtu: 1500

    br-ex:
       interfaces: [enp6s0.40]
       addresses: [192.168.0.58/24]
       gateway4: 192.168.0.100
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500
       parameters:
          stp: true
          forward-delay: 4
```

5.1.5.4. Nodo de almacenamiento

```
network:
  version: 2
  renderer: networkd

  ethernets:
    enp2s0:
      dhcp4: false

  vlans:
     enp2s0.10:
        id: 10
        link: enp2s0

     enp2s0.20:
        id: 20
        link: enp2s0

     enp2s0.30:
        id: 30
        link: enp2s0

     enp2s0.40:
        id: 40
        link: enp2s0

  bridges:
    br-mgmt:
       interfaces: [enp2s0.10]
       addresses: [10.0.236.143/22]
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500

    br-storage:
       interfaces: [enp2s0.20]
       addresses: [10.0.244.143/22]
       mtu: 1500

    br-vxlan:
       interfaces: [enp2s0.30]
       addresses: [10.0.240.143/22]
       mtu: 1500

    br-vlan:
       interfaces: [enp2s0.40]
       mtu: 1500

    br-ex:
       interfaces: [enp2s0]
       addresses: [192.168.0.59/24]
       gateway4: 192.168.0.100
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500
       parameters:
          stp: true
          forward-delay: 4
```

5.2. Entorno de producción (Aula Ateca)

5.2.1. Creación de las VLAN en el switch D-Link DGS 1210-26

En este paso vamos a crear las cuatro VLANs que necesitamos en OpenStack.

Tabla 7. Listado de VLANs de OpenStack

VLAN Id	Nombre
10	br-mgmt
20	br-storage
30	br-vxlan
40	br-vlan

En este caso la creación de las VLANS la vamos a realizar desde la interfaz web del switch. Para acceder al panel de administración del switch abrimos la URL `https://172.16.0.5` desde un navegador web.

Una vez que hemos accedido al panel de administración seleccionamos la opción **VLAN → 802.1Q VLAN** en el panel de la izquierda para crear las VLANS.

A continuación se muestra una captura de pantalla de todo el proceso de creación de las VLANs desde la interfaz web del switch.

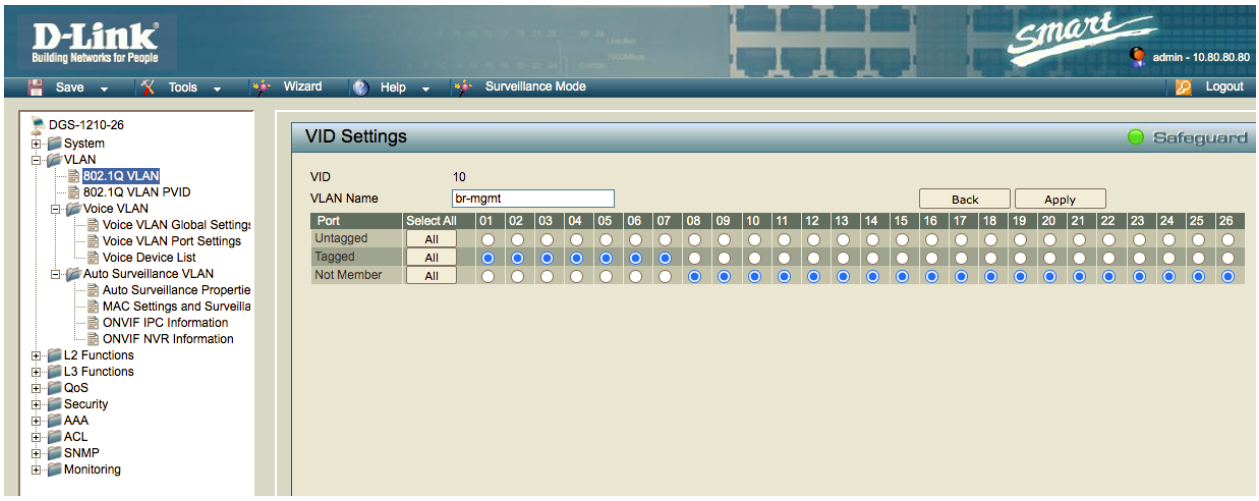

Figura 11. Creación de la VLAN 10 para los puetos del 1 al 7.

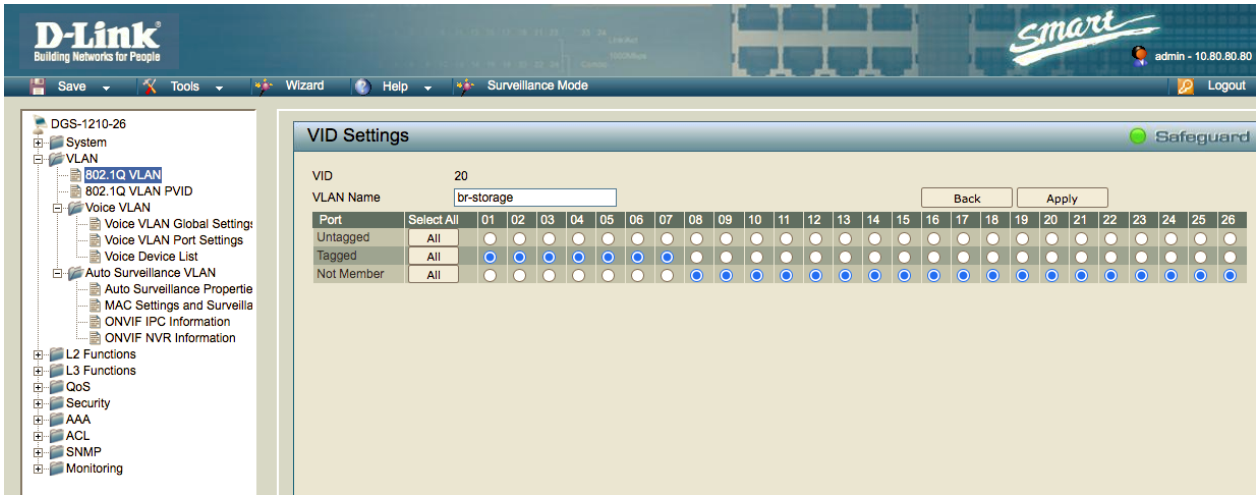

Figura 12. Creación de la VLAN 20 para los puetos del 1 al 7.

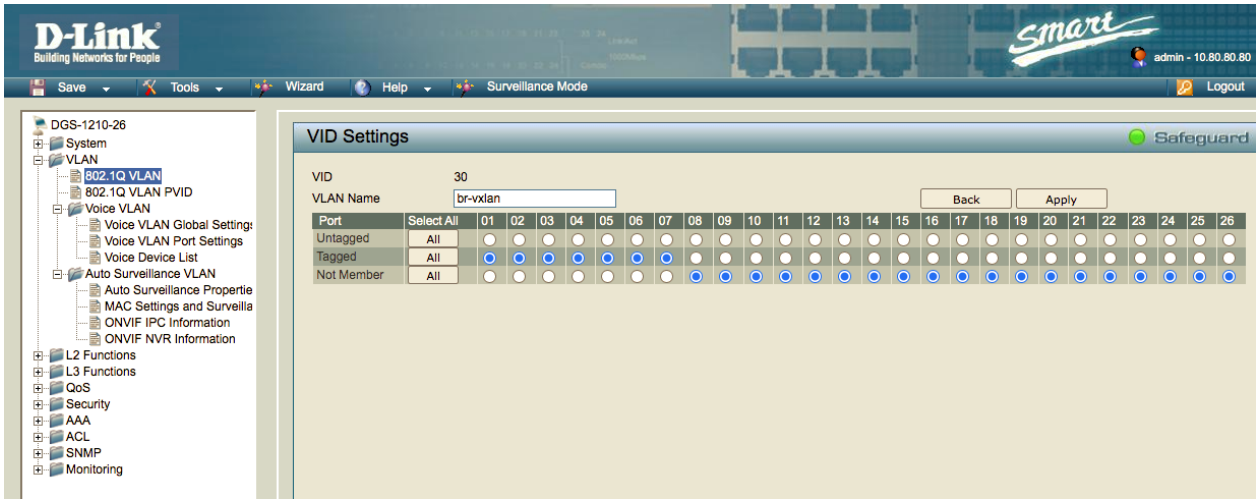

Figura 13. Creación de la VLAN 30 para los puetos del 1 al 7.

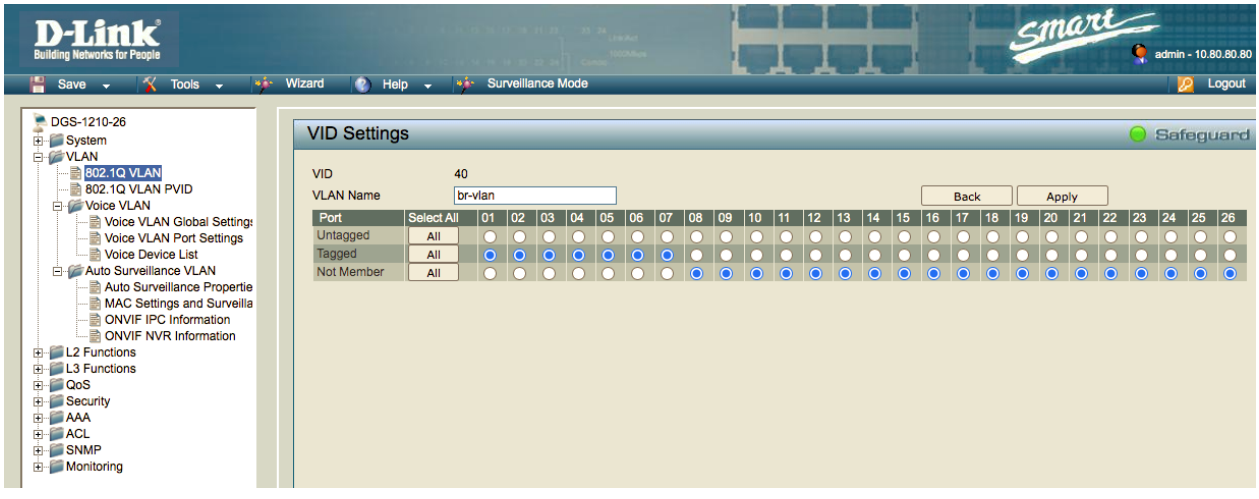

Figura 14. Creación de la VLAN 40 para los puetos del 1 al 7.

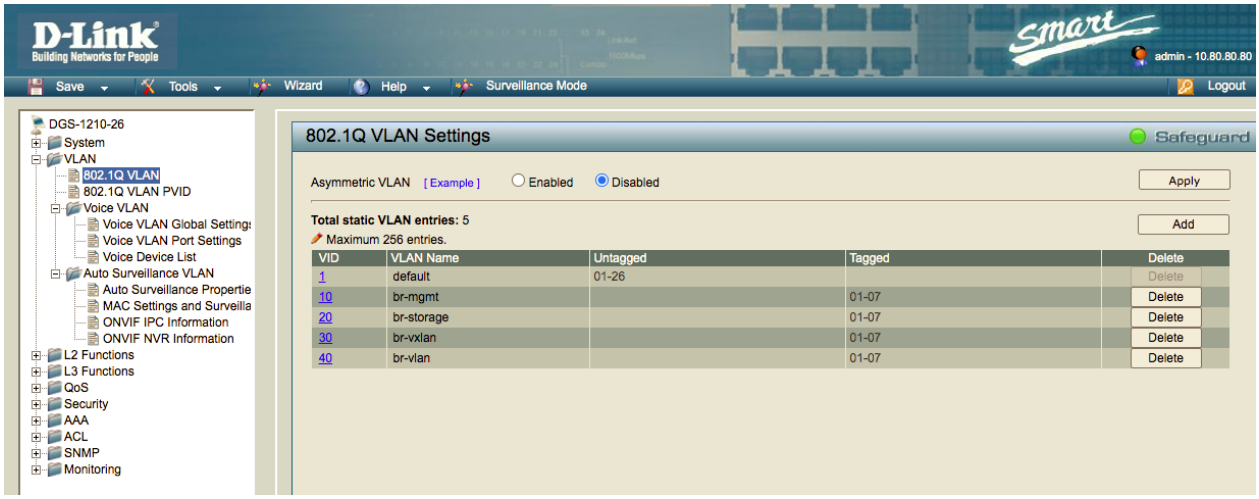

Figura 15. Resumen de todas las VLANs que hemos creado en el switch.

Después de configurar las VLANs tenemos que guardar la configuración para que los cambios realizados sean persistentes, en otro caso, las VLANs que hemos creado se perderían al reiniciar el switch.

Para guardar los cambios accedemos a la opción **Save Config** del panel de la izquierda. La ruta completa es la siguiente: **Save** → **Save Configuration** → **Save Config**.

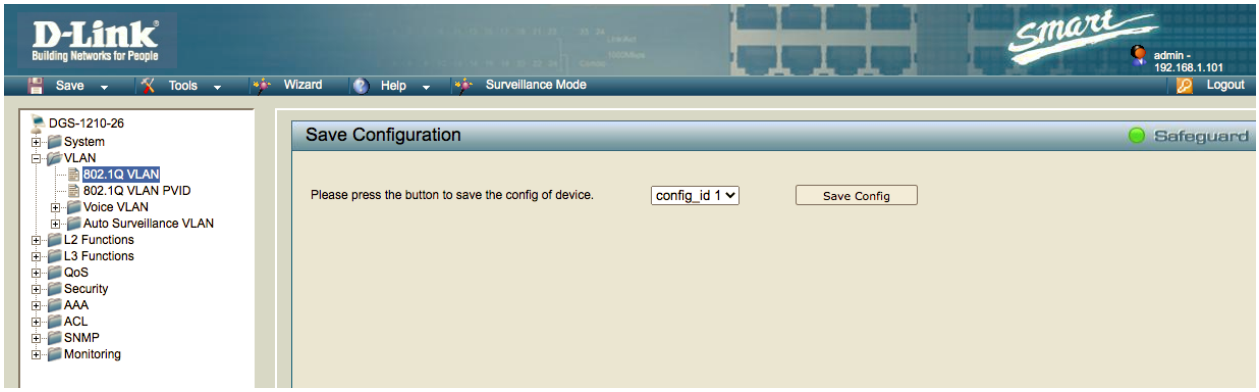

Figura 16. Guardamos la configuración del switch.

Referencia:

- Manual de usuario del Switch D-Link DGS 1210-26

5.2.2. Configuración de las tarjetas de red en los nodos (Netplan)

La configuración de red de los nodos del entorno de producción van a tener algunas diferencias respecto a los nodos del entorno de desarrollo, ya que los equipos del entorno de producción disponen de dos tarjetas de red que se configurarán en modo *bonding*, mientras que los equipos del entorno de desarrollo sólo disponen de una tarjeta de red.

La configuración que vamos a utilizar para configurar las tarjetas en modo *bonding* es la siguiente:

- Nombre: `bond0`
- Bond mode: `802.3ad`
- XMIT hash policy: `layer3+4`
- LACTP rate: `fast`

Nota importante sobre el valor del MTU (*Maximum Transmission Unit*)

- El valor del MTU en una instalación sobre máquinas físicas se deja por defecto a **1500** bytes.
- El valor del MTU en una instalación sobre máquinas virtuales se configura a **1450** bytes.
- El valor del MTU en las instancias creadas sobre una instalación virtual se configura a **1400** bytes.

Tenemos pediente la configuración del MTU a **9000** bytes (Jumbo Frames) para mejorar el rendimiento de la red y hacer que las transmisiones de datos sean más eficientes.

Referencias:

- Ejemplo de configuración Netplan para un nodo de OpenStack
- Netplan configuration examples
- Netplan Reference.

5.2.2.1. Nodo bastión

```yaml
network:
  version: 2
  renderer: networkd

  ethernets:
    enp2s0:
      dhcp4: false

  vlans:
     enp2s0.10:
        id: 10
        link: enp2s0

     enp2s0.20:
        id: 20
        link: enp2s0

     enp2s0.30:
        id: 30
        link: enp2s0

     enp2s0.40:
        id: 40
        link: enp2s0

  bridges:
    br-mgmt:
       interfaces: [enp2s0.10]
       addresses: [10.0.236.140/22]
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500

    br-storage:
       interfaces: [enp2s0.20]
       addresses: [10.0.244.140/22]
       mtu: 1500

    br-vxlan:
       interfaces: [enp2s0.30]
       addresses: [10.0.240.140/22]
       mtu: 1500

    br-ex:
       interfaces: [enp2s0]
       addresses: [172.16.0.10/16]
       gateway4: 172.16.0.1
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500
       parameters:
          stp: true
          forward-delay: 4

    br-vlan:
       interfaces: [enp2s0.40]
       mtu: 1500
```

5.2.2.2. Nodo de control

```
network:
  version: 2
  renderer: networkd

  ethernets:
    eno1: {}
    eno2: {}

  bonds:
    bond0:
      dhcp4: false
      interfaces:
      - eno1
      - eno2
      parameters:
        lacp-rate: fast
        mode: 802.3ad
        transmit-hash-policy: layer3+4

  vlans:
     bond0.10:
        id: 10
        link: bond0

     bond0.20:
        id: 20
        link: bond0

     bond0.30:
        id: 30
        link: bond0

     bond0.40:
        id: 40
        link: bond0

  bridges:
    br-mgmt:
       interfaces: [bond0.10]
       addresses: [10.0.236.141/22]
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500

    br-storage:
       interfaces: [bond0.20]
       addresses: [10.0.244.141/22]
       mtu: 1500

    br-vxlan:
       interfaces: [bond0.30]
       addresses: [10.0.240.141/22]
       mtu: 1500

    br-vlan:
       interfaces: [bond0.40]
       mtu: 1500

    br-ex:
       interfaces: [bond0]
```

```
      addresses: [172.16.0.11/16]
      gateway4: 172.16.0.1
      nameservers:
         addresses: [8.8.8.8]
      mtu: 1500
      parameters:
         stp: true
         forward-delay: 4
```

5.2.2.3. Nodo de cómputo

```
network:
  version: 2
  renderer: networkd

  ethernets:
    eno1: {}
    eno2: {}

  bonds:
    bond0:
      dhcp4: false
      interfaces:
      - eno1
      - eno2
      parameters:
        lacp-rate: fast
        mode: 802.3ad
        transmit-hash-policy: layer3+4

  vlans:
     bond0.10:
        id: 10
        link: bond0

     bond0.20:
        id: 20
        link: bond0

     bond0.30:
        id: 30
        link: bond0

     bond0.40:
        id: 40
        link: bond0

  bridges:
    br-mgmt:
       interfaces: [bond0.10]
       addresses: [10.0.236.142/22]
       nameservers:
          addresses: [8.8.8.8]
       mtu: 1500

    br-storage:
       interfaces: [bond0.20]
       addresses: [10.0.244.142/22]
       mtu: 1500

    br-vxlan:
```

```
      interfaces: [bond0.30]
      addresses: [10.0.240.142/22]
      mtu: 1500

  br-vlan:
      interfaces: [bond0.40]
      mtu: 1500

  br-ex:
      interfaces: [bond0]
      addresses: [172.16.0.12/16]
      gateway4: 172.16.0.1
      nameservers:
        addresses: [8.8.8.8]
      mtu: 1500
      parameters:
        stp: true
        forward-delay: 4
```

5.2.2.4. Nodo de almacenamiento

```
network:
  version: 2
  renderer: networkd

  ethernets:
    eno3: {}
    eno4: {}

  bonds:
    bond0:
      dhcp4: false
      interfaces:
      - eno3
      - eno4
      parameters:
        lacp-rate: fast
        mode: 802.3ad
        transmit-hash-policy: layer3+4

  vlans:
     bond0.10:
        id: 10
        link: bond0

     bond0.20:
        id: 20
        link: bond0

     bond0.30:
        id: 30
        link: bond0

     bond0.40:
        id: 40
        link: bond0

  bridges:
    br-mgmt:
      interfaces: [bond0.10]
      addresses: [10.0.236.143/22]
```

```
      nameservers:
        addresses: [8.8.8.8]
      mtu: 1500

  br-storage:
      interfaces: [bond0.20]
      addresses: [10.0.244.143/22]
      mtu: 1500

  br-vxlan:
      interfaces: [bond0.30]
      addresses: [10.0.240.143/22]
      mtu: 1500

  br-vlan:
      interfaces: [bond0.40]
      mtu: 1500

  br-ex:
      interfaces: [bond0]
      addresses: [172.16.0.13/16]
      gateway4: 172.16.0.1
      nameservers:
        addresses: [8.8.8.8]
      mtu: 1500
      parameters:
        stp: true
        forward-delay: 4
```

6. Instalación de OpenStack Ansible

6.1. Preparación del host Bastión

El host **Bastión** es el host desde el que vamos a realizar la ejecución de los playbooks de OpenStack Ansible. En este equipo tenemos que realizar las siguientes operaciones:

1. Instalar las dependencias.
2. Comprobar que el servicio NTP está activado.
3. Crear un par de claves SSH par el usuario `root` y copiar la clave pública en cada uno de los nodos.

6.1.1. Instalación de las dependencias

- Conectamos por SSH con la instancia.
- Ejecutamos los siguientes comandos para instalar los paquetes necesarios.

```
# apt update
# apt dist-upgrade -y
# apt install build-essential git chrony openssh-server python3-dev sudo -y
# reboot
```

6.1.2. Comprobación del servicio NTP

Comprobamos que el servicio NTP está activado. Podemos comprobarlo ejecutando el comando: `timedatectl`.

6.1.3. Creación de un par de claves SSH para el usuario `root`

En el host **Bastión** ejecutamos el comando `ssh-keygen` para crear una clave pública y privada para el usuario `root`.

Una vez que hemos creado la clave pública y privada del usuario `root`, consultamos el valor de la clave pública para copiarla en el resto de nodos. Desde el host **Bastión** ejectuamos los siguientes comandos:

```
sudo su
ssh-keygen
cat /root/.ssh/id_rsa.pub
(Copiamos la clave pública)
```

Conectamos por SSH con cada una de las máquinas (**Control**, **Cómputo** y **Almacenamiento**) y copiamos la clave pública del host **Bastión** en el archivo `/root/.ssh/authorized_keys`.

```
sudo su
nano /root/.ssh/authorized_keys
(Copiamos la clave pública del host Bastión)
```

Hay que copiar el contenido del archivo `id_rsa.pub` al directorio `/root/.ssh/authorized_keys`.

Para comprobar que podemos conectar con todos los nodos podemos ejecutar los siguientes comandos desde el host **Bastión**.

```
ssh root@IP-CONTROL
ssh root@IP-COMPUTO
ssh root@IP-ALMACENAMIENTO
```

6.2. Preparación de los equipos destino

En los nodos de **Control**, **Cómputo** y **Almacenamiento** realizamos los siguientes pasos.

1. Instalar las dependencias.
2. Comprobar que el servicio NTP está activado.

6.2.1. Instalación de las dependencias en los equipos destino

- Conectamos por SSH con la instancia.
- Ejecutamos los siguientes comandos para instalar los paquetes necesarios.

```
# apt update
# apt dist-upgrade -y
# apt install bridge-utils debootstrap openssh-server tcpdump vlan python3 -y
# apt install linux-modules-extra-$(uname -r) -y
# reboot
```

Recuerde que estos pasos se tienen que realizar en los nodos: **Control**, **Cómputo** y **Almacenamiento**.

6.2.2. Comprobación del servicio NTP

Comprobamos que el servicio NTP está activado. Podemos comprobarlo ejecutando el comando: `timedatectl`.

6.3. Creación de un servicio para crear el par veth en el nodo de control

En la documentación oficial de OpenStack Ansible encontramos la siguiente información sobre la creación del par de interfaces `veth` en el nodo de control.

Creación de un par de interfaces veth

(*Fuente: Texto extraído literalmente de la documentación oficial*)

Configure OpenStack Networking flat (untagged) network in the provider_networks subsection:

```
provider_networks:
  - network:
      group_binds:
        - neutron_linuxbridge_agent
      container_bridge: "br-ex"
      container_type: "veth"
      container_interface: "eth12"
      host_bind_override: "PHYSICAL_NETWORK_INTERFACE"
      type: "flat"
      net_name: "flat"
```

Replace `PHYSICAL_NETWORK_INTERFACE` with the network interface used for flat networking. Ensure this is a physical interface on the same L2 network being used with the `br-ex` devices. If no additional network interface is available, a veth pair plugged into the `br-ex` bridge can provide the necessary interface.

The following is an example of creating a veth-pair within an existing bridge:

```
# Create veth pair, do not abort if already exists
pre-up ip link add br-vlan-veth type veth peer name PHYSICAL_NETWORK_INTERFACE || true

# Set both ends UP
pre-up ip link set br-vlan-veth up
pre-up ip link set PHYSICAL_NETWORK_INTERFACE up

# Delete veth pair on DOWN
post-down ip link del br-vlan-veth || true
bridge_ports br-vlan-veth
```

Referencias:

- Configuring target host networking. ☆
- veth - Virtual Ethernet Device.
- Networkd for nspawn with OpenStack-Ansible.
- Network Configuration.

Netplan todavía no tiene soporte para `veth`, por lo tanto podemos crear un script que ejecute los comandos necesarios y gestionarlo como un servicio.

Creamos un archivo en la siguiente ruta.

```
# nano /usr/local/bin/br-ex-veth.sh
```

Añadimos los comandos de creación del para la pareja de interfaces virtuales veth.

```
#!/bin/bash
ip link add br-ex-veth type veth peer name eth12
ip link set br-ex-veth up
ip link set eth12 up
brctl addif br-ex br-ex-veth
```

Hemos utilizado el nombre de `eth12` para el interfaz que utiliza el agente de neutron, para no tener que modificar el parámetro `host_bind_override: "eth12"` del archivo `openstack_user_config.yml`.

Actualizamos los permisos del archivo

```
# chmod 744 /usr/local/bin/br-ex-veth.sh
```

Creamos un archivo para definir el servicio.

```
# nano /etc/systemd/system/br-ex-veth.service
```

Definimos el servicio en el archivo que acabamos de crear.

```
[Unit]
After=network.service

[Service]
ExecStart=/usr/local/bin/br-ex-veth.sh

[Install]
WantedBy=default.target
```

Actualizamos los permisos del archivo

```
# chmod 644 /etc/systemd/system/br-ex-veth.service
```

Configuramos el servicio para se inicie de forma automática al iniciar la máquina

```
# systemctl daemon-reload
# systemctl enable br-ex-veth.service
# systemctl start br-ex-veth.service
```

Podemos comprobar el estado de los bridges de red con el comando:

```
brctl show
```

6.4. Configuramos un volumen de almacenamiento para el nodo de almacenamiento

En este paso vamos a configurar un volumen LVM para almacenar los volúmenes del servicio Cinder.

Para consultar la lista de volúmenes que existen en el sistema podemos utilizar el comando:

```
lsblk
```

Entorno de desarrollo

En el entorno de desarrollo del **departamento** estamos utilizando un segundo disco SSD en **/dev/sdb**.

```
sudo pvcreate --metadatasize 2048 /dev/sdb
sudo vgcreate cinder-volumes /dev/sdb
```

Comprobamos que el volumen se ha creado de forma correcta.

```
vgdisplay
```

Entorno de producción

En el entorno de producción del **aula Ateca** estamos utilizando un **RAID 5** formado por cuatro discos físicos en **/dev/sdb**.

```
sudo pvcreate --metadatasize 2048 /dev/sdb
sudo vgcreate cinder-volumes /dev/sdb
```

Comprobamos que el volumen se ha creado de forma correcta.

```
vgdisplay
```

El resultado del comando nos muestra que existen dos volúmenes LVM uno para el sistema operativo (`ubuntu-vg`) y el que acabamos de crear para el servicio Cinder (`cinder-`

`volumes`).

```
root@almacenamiento:/home/usuario# vgdisplay
  --- Volume group ---
  VG Name               cinder-volumes
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               <6,55 TiB
  PE Size               4,00 MiB
  Total PE              1715967
  Alloc PE / Size       0 / 0
  Free  PE / Size       1715967 / <6,55 TiB
  VG UUID               QHIOA2-DXc1-NfSG-0Vu4-h22Q-o3tN-sZ1N6D

  --- Volume group ---
  VG Name               ubuntu-vg
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               444,07 GiB
  PE Size               4,00 MiB
  Total PE              113682
  Alloc PE / Size       113682 / 444,07 GiB
  Free  PE / Size       0 / 0
  VG UUID               3hMt0M-BdhG-J570-eLCE-c1oB-3yrt-Cvozvl
```

Comandos útiles para trabajar con LVM

```
pvcreate                    # Crea un volumen físico (physical volume). Destruye todos los datos.
vgcreate                    # Crea un grupo de volúmenes (volume group)
lvcreate                    # Crea un volumen lógico (logical volume)

pvs                         # Muestra los atributos de los volúmenes físicos
vgs                         # Muestra los atributos de los grupos de volúmenes

mkfs -t ext4 ...            # Formatea el volumen lógico que hemos creado

pvdisplay                   # Muestra información de los physical volumes
vgdisplay                   # Muestra información de todos los volúmenes creados
vgdisplay cinder-volumes    # Muestra información de un volumen en concreto

vgremove                    # Elimina un volumen
```

Referencias:

- Ejemplos de configuración de LVM.
- How To Use LVM To Manage Storage Devices on Ubuntu.
- Creación de Volúmenes Lógicos en Linux (LVM).

6.5. Creamos un archivo de swap en el nodo de Control

En el **entorno de desarrollo** es necesario crear un archivo de swap en el **nodo de Control**, para hacerlo tenemos que ejecutar los siguientes comandos como `root`:

Reservamos espacio de disco para crear un archivo de swap llamado `/swapfile`. En este caso vamos a crear un archivo de 16 GB, pero dependiendo de la instalación será necesario ampliar el tamaño del archivo a 32 o 64 GB.

```
# fallocate -l 16G /swapfile
```

Modificamos los permisos del archivo de swap para que solo el usuario `root` pueda escribir en el.

```
# chmod 600 /swapfile
```

Convertimos el archivo en un área de swap para Linux.

```
# mkswap /swapfile
```

Activamos el archivo de swap.

```
# swapon /swapfile
```

Editamos el archivo `/etc/fstab` y añadimos la siguiente línea para el archivo de swap sea permanente después de cada reinicio de la máquina.

```
/swapfile swap swap defaults 0 0
```

Comprobamos que la swap está activa.

```
# sudo swapon --show
```

Si la swap está activa debe aparecer un resultado similar a este:

```
NAME      TYPE SIZE USED PRIO
/swapfile file  16G 1.5G   -2
```

6.6. Configuramos el valor de `swappiness` que determina el uso de la memoria virtual

El **swappiness** es una propiedad del kernel de Linux que permite configurar cuando se utilizará la memoria virtual **swap** en función del espacio libre que haya de memoria física **RAM**.

La propiedad de swappiness se puede configurar con un valor entre 0 y 100. Un valor igual a 0 quiere decir que el sistema nunca hará uso de la memoria virtual, y un valor igual a 100, quiere decir que se estará haciendo uso de la memoria virtual todo momento. Esta configuración puede provocar una degradación del rendimiento del sistema.

El valor que suele tener una instalación Linux por defecto es **60**, que quiere decir que se empezará a utilizar la memoria virtual swap cuando quede un 40% libre de memoria RAM del sistema.

Para comprobar el valor del **swapiness** del sistema podemos ejecutar este comando.

```
# cat /proc/sys/vm/swappiness
```

Podemos modificar el valor del **swapiness** en tiempo de ejecucion, con el comando:

```
# sysctl vm.swappiness=10
```

El valor que se haya configurado con este comando se perderá después de reiniciar el sistema. Si queremos que los cambios sean permanentes podemos hacerlo configurando el parámetro `vm.swappiness` del archivo `/etc/sysctl.conf`.

```
vm.swappiness=10
```

6.7. Instalamos el código fuente y las dependencias en el host bastión

Seleccionamos el **branch** que queremos instalar. En este caso se está seleccionando la **stable/xena**, que se corresponde con la versión **24.0.0**, pero la última en desarrollo es **24.0.1**.

```
# git clone -b stable/xena https://opendev.org/openstack/openstack-ansible /opt/openstack-ansible
# cd /opt/openstack-ansible
# scripts/bootstrap-ansible.sh
```

Referencia:

- https://releases.openstack.org

6.8. Configuramos el deployment en el host bastion

```
# cp -R /opt/openstack-ansible/etc/openstack_deploy/ /etc/
# cd /etc/openstack_deploy
# cp openstack_user_config.yml.example openstack_user_config.yml
```

Referencia:

- https://docs.openstack.org/project-deploy-guide/openstack-ansible/latest/configure.html

6.9. Configuramos el archivo `openstack_user_config.yml`

Copiamos el archivo `openstack_user_config.yml` de nuestro repositorio al directorio `/etc/openstack_deploy`.

Referencia:

- https://docs.openstack.org/openstack-ansible/latest/user/prod/provnet_groups.html

6.9.1. openstack_user_config.yml

```
---
cidr_networks:
  container: 10.0.236.0/22
  tunnel: 10.0.240.0/22
  storage: 10.0.244.0/22
  exit: 172.16.0.0/16

used_ips:
  - "10.0.236.1,10.0.236.254"
  - "10.0.240.1,10.0.240.254"
  - "10.0.244.11,10.0.244.254"
  - "172.16.0.1,172.16.0.20"

global_overrides:
  internal_lb_vip_address: 10.0.236.141       # Nodo de Control. VIP Interna
  external_lb_vip_address: 172.16.0.11        # Nodo de Control. VIP Externa
  management_bridge: "br-mgmt"
  provider_networks:
    - network:
        container_bridge: "br-mgmt"
        container_type: "veth"
        container_interface: "eth1"
        ip_from_q: "container"
        type: "raw"
        group_binds:
          - all_containers
          - hosts
        is_container_address: true
    - network:
        container_bridge: "br-vxlan"
        container_type: "veth"
        container_interface: "eth10"
        ip_from_q: "tunnel"
        type: "vxlan"
        range: "1:1000"
        net_name: "vxlan"
        group_binds:
          - neutron_linuxbridge_agent
    - network:
        container_bridge: "br-ex"          # Valor original: br-vlan
        container_type: "veth"
        container_interface: "eth12"
        host_bind_override: "eth12"        # Valor original: eth12. Nombre de la interfaz del par veth
        ip_from_q: "exit"
        type: "flat"
        net_name: "exit"
        group_binds:
          - neutron_linuxbridge_agent
    - network:
        container_bridge: "br-vlan"
        container_type: "veth"
        container_interface: "eth11"
        type: "vlan"
        range: "101:200,301:400"
        net_name: "vlan"
        group_binds:
          - neutron_linuxbridge_agent
    - network:
        container_bridge: "br-storage"
        container_type: "veth"
```

```
        container_interface: "eth2"
        ip_from_q: "storage"
        type: "raw"
        group_binds:
          - glance_api
          - cinder_api
          - cinder_volume
          - nova_compute

###
### Infrastructure
###

# galera, memcache, rabbitmq, utility
shared-infra_hosts:
  infra1:
    ip: 10.0.236.141

# repository (apt cache, python packages, etc)
repo-infra_hosts:
  infra1:
    ip: 10.0.236.141

###
### OpenStack
###

# List of target hosts on which to deploy the glance API, nova API, heat API,
# and horizon. Recommend three minimum target hosts for these services.
# Typically contains the same target hosts as 'shared-infra_hosts' level.
os-infra_hosts:
  infra1:
    ip: 10.0.236.141

# Define three OpenStack identity hosts:
#
identity_hosts:
  infra1:
    ip: 10.0.236.141

# Define three OpenStack storage infrastructure hosts:
# cinder api services
storage-infra_hosts:
  infra1:
    ip: 10.0.236.141

# glance
image_hosts:
  infra1:
    ip: 10.0.236.141

# placement
placement-infra_hosts:
  infra1:
    ip: 10.0.236.141

# nova api, conductor, etc services
compute-infra_hosts:
  infra1:
    ip: 10.0.236.141

# heat
orchestration_hosts:
```

```
    infra1:
      ip: 10.0.236.141

# horizon
dashboard_hosts:
  infra1:
    ip: 10.0.236.141

# Define three OpenStack network hosts:
# neutron server, agents (L3, etc)
network_hosts:
  infra1:
    ip: 10.0.236.141

# Define an OpenStack compute host:
# nova hypervisors
compute_hosts:
  compute1:
    ip: 10.0.236.142

# Define an OpenStack storage host:
# cinder storage host
storage_hosts:
  storage1:
    ip: 10.0.236.143
    container_vars:
      cinder_backends:
        limit_container_types: cinder_volume
        lvm:
          volume_group: cinder-volumes
          volume_driver: cinder.volume.drivers.lvm.LVMVolumeDriver
          volume_backend_name: LVM_iSCSI
          iscsi_ip_address: "10.0.244.143"
```

6.10. Configuramos el archivo `user_variables.yml`

- Configuramos el método de instalación como `source`.

```
install_method: source
```

En la documentación oficial hay una tabla donde aparece la compatibilidad entre los sistemas operativos y los dos métodos de instalación `source` y `distro`.

En nuestro caso vamos a utilizar el método de instalación `source` que es compatible con la versión `Xena` para el sistema operativo `Ubuntu 20.04`.

Incrementamos el valor del parámetro: `lxc_cache_prep_timeout`.

```
lxc_cache_prep_timeout: 2700
```

En la documentación ofcial describen el parámetro `lxc_cache_prep_timeout` como:

```
The maximum amount of time (in seconds) to wait until failing the cache
preparation process. This is necessary to mitigate the issue that can
arise where the cache prep hangs and never fails.
The value is specified in seconds, with the default being 20 minutes.
```

6.10.1. user_variables.yml

```
## Debug and Verbose options.
debug: false

# The maximum amount of time (in seconds) to wait until failing the cache
# preparation process. This is necessary to mitigate the issue that can
# arise where the cache prep hangs and never fails.
# The value is specified in seconds, with the default being 20 minutes.
lxc_cache_prep_timeout: 2700

## Installation method for OpenStack services
install_method: source
```

6.11. Configuramos las credenciales

Configuramos las contraseñas del archivo `/etc/openstack_deploy/user_secrets.yml`.

```
# cd /opt/openstack-ansible
# ./scripts/pw-token-gen.py --file /etc/openstack_deploy/user_secrets.yml
```

6.12. Comprobamos la integridad de los archivos de instalación

```
# cd /opt/openstack-ansible/playbooks
# openstack-ansible setup-infrastructure.yml --syntax-check
```

6.13. Ejecutamos los playbooks

Ejecutamos el playbook encargado de configurar los nodos y crear los contenedores para cada uno de los servicios de OpenStack.

```
# cd /opt/openstack-ansible/playbooks
# openstack-ansible setup-hosts.yml
```

Error: `FAILED - RETRYING: Ensure that the LXC cache has been prepared`

Cuando realizamos la instalación en el entorno de desarrollo del departamento, nos apareció un mensaje de error al ejecutar el playbook `setup-hosts.yml`.

Error

```
FAILED - RETRYING: Ensure that the LXC cache has been prepared
```

Solución

Ese error se soluciona incrementando el valor del timeout de la preparación de la caché.

En el archivo `/etc/openstack_deploy/user_variables.yml` hay que añadir este valor:

```
lxc_cache_prep_timeout: 2700
```

Referencia:

- https://bugs.launchpad.net/openstack-ansible/+bug/1748864

Ejecutamos el playbook que se encarga de configurar la infraestructura de OpenStack.

```
# openstack-ansible setup-infrastructure.yml
```

Ahora podríamos comprobar que el cluster Galera de base de datos se ha creado de forma correcta. En nuestro caso el cluster sólo estará formado por un nodo.

```
# ansible galera_container -m shell \
  -a "mysql -h localhost -e 'show status like \"%wsrep_cluster_%\";'"
```

Para finalizar la instalación de OpenStack ejecutamos el playbook `setup-openstack.yml`.

```
# openstack-ansible setup-openstack.yml
```

Referencia:

- Ejecución de los playbooks.

6.14. Verificación de la instalación

Desde el **nodo de Control** con el usuario `root`, nos conectamos al contenedor de utilidades.

```
# lxc-attach -n `lxc-ls -1 | grep utility | head -n 1`
```

Una vez que estamos dentro dentro del contenedor, ejectuamos el script con las credenciales del usuario `admin`.

```
. ~/openrc
```

Una vez hecho esto ya podemos utilizar la utilidad `openstack` para la línea de comandos CLI.

Seguir los pasos que aparecen en la documentación oficial:

- https://docs.openstack.org/openstack-ansible/latest/admin/openstack-firstrun.html

6.15. Comprobamos la configuración de los agentes de red para verificar que el par veth está funcionando de forma correcta

En el nodo de control

En la etiqueta `exit` (que es la que hemos definido en OpenStack como red externa) del archivo en el archivo: `/etc/neutron/plugins/ml2/linuxbridge_agent.ini`, configuramos la interfaz `eth12`.

```
[linux_bridge]
physical_interface_mappings = vlan:br-vlan,exit:eth12
```

Reiniciamos los servicios de neutron:

```
systemctl restart neut*
```

Podemos comprobar el estado de los bridges de red con el comando:

```
brctl show
```

En el nodo de cómputo

En el nodo de cómputo no es necesario tener la etiqueta `exit` en el archivo: `/etc/neutron/plugins/ml2/linuxbridge_agent.ini`.

```
[linux_bridge]
physical_interface_mappings = vlan:br-vlan
```

Reiniciamos los servicios de neutron:

```
systemctl restart neut*
```

Referencia:

- https://cloudbase.it/openstack-on-arm64-lbaas/

7. Configuración de OpenStack

7.1. Obtener la contraseña del usuario admin

Para obtener la contraseña del usuario admin podemos ejecutar el siguiente comando desde el **nodo Bastión**:

```
# cat /etc/openstack_deploy/user_secrets.yml | grep keystone_auth_admin_password
```

7.2. Imágenes disponibles para OpenStack

En la documentación oficial de OpenStack podemos encontrar un listado muy completo de imágenes preparadas para crear instancias en un cloud como el de OpenStack. Podemos encontrar imágenes de Debian, Ubuntu, Fedora, OpenSuse, Windows Server 2012 R, entre otras.

La mayoría de estas imágenes incluyen el paquete cloud-init que permite realizar una configuración inicial sobre las instancias que se crean en el cloud. Una de las principales tareas de cloud-init es que permite inyectar una clave pública SSH a la instancia para que podamos conectarnos a ella con una clave privada, ya que la mayoría de las imágenes tiene deshabilitado el mecanismo de autenticación por contraseña.

Referencias:

- Listado de imágenes para OpenStack
- Documentación oficial de cloud-init
- Vídeo sobre cloud-init de Alberto Molina

7.3. Publicar imágenes en OpenStack

Podemos publicar imágenes en OpenStack desde:

- el **panel de control web de Horizon**,
- o desde la **línea de comandos con la utilidad OpenStackClient (OSC)**.

En nuestro caso hemos utilizado la utilidad de línea de comandos. A continuación se describen los pasos que hemos realizado.

En primer lugar, nos conectamos al contenedor de utilidades que está en el **nodo de Control**.

```
lxc-attach -n `lxc-ls -1 | grep utility | head -n 1`
```

Una vez que estamos dentro dentro del contenedor, ejectuamos el script con las credenciales del usuario `admin`.

```
. ~/openrc
```

Una vez hecho esto ya podemos utilizar la utilidad `openstack` para la línea de comandos CLI.

7.3.1. Cirros

```
# cd /tmp

# wget http://download.cirros-cloud.net/0.5.1/cirros-0.5.1-x86_64-disk.img

# openstack image create \
   --container-format bare \
   --disk-format qcow2 \
   --file cirros-0.5.1-x86_64-disk.img \
   --progress \
   cirros

# openstack image set \
   --public \
   --protected \
   cirros
```

Los datos de acceso predefinidos son los siguientes:

- Usuario: `cirros`
- Password: `gocubsgo`

Referencias:

- Cirros Cloud Images
- Documentación oficial `image create`
- Documentación oficial `image set`

7.3.2. Ubuntu 18.04

```
# cd /tmp

# wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img

# openstack image create \
   --container-format bare \
   --disk-format qcow2 \
   --file bionic-server-cloudimg-amd64.img \
   --progress \
   ubuntu-bionic

# openstack image set \
   --public \
   --protected \
   ubuntu-bionic
```

Al crear la instancia podemos definir cuál será el password del usuario `ubuntu`.

Launch Instance → Configuracion → Customisation Script

```
#cloud-config
password: mypasswd
chpasswd: { expire: False }
ssh_pwauth: True
```

Referencias:

- Ubuntu Cloud Images
- Documentación oficial `image create`
- Documentación oficial `image set`

7.3.3. Ubuntu 20.04.4 LTS

```
# cd /tmp

# wget https://cloud-images.ubuntu.com/focal/current/focal-server-cloudimg-amd64.img

# openstack image create \
   --container-format bare \
   --disk-format qcow2 \
   --file /tmp/focal-server-cloudimg-amd64.img \
   --progress \
   ubuntu-20.04

# openstack image set \
   --public \
   --protected \
   ubuntu-20.04
```

El usuario predefinido es `ubuntu`.

Referencias:

- Ubuntu Cloud Images
- Documentación oficial `image create`
- Documentación oficial `image set`

7.3.4. Debian 10

```
# cd /tmp

# wget http://cdimage.debian.org/cdimage/openstack/current-10/debian-10-openstack-amd64.qcow2

# openstack image create \
   --container-format bare \
   --disk-format qcow2 \
   --file debian-10-openstack-amd64.qcow2 \
   --progress \
   debian-10

# openstack image set \
   --public \
   --protected \
   debian-10
```

El usuario predefinido es `debian`.

Referencias:

- Debian Official Cloud Images for OpenStack
- Documentación oficial `image create`
- Documentación oficial `image set`

7.3.5. Fedora 35

```
# cd /tmp

# wget
https://download.fedoraproject.org/pub/fedora/linux/releases/35/Cloud/x86_64/images/Fedora-
Cloud-Base-35-1.2.x86_64.qcow2

# openstack image create \
   --container-format bare \
   --disk-format qcow2 \
   --file Fedora-Cloud-Base-35-1.2.x86_64.qcow2 \
   --progress \
   fedora-35

# openstack image set \
   --public \
   --protected \
   fedora-35
```

El usuario predefinido es `fedora`.

Referencias:

- Fedora Cloud Base Images
- Documentación oficial `image create`
- Documentación oficial `image set`

7.3.6. Free BSD 13.0

```
# cd /tmp

# wget https://object-
storage.public.mtl1.vexxhost.net/swift/v1/1dbafeefbd4f4c80864414a441e72dd2/bsd-cloud-
image.org/images/freebsd/13.0/freebsd-13.0-zfs.qcow2

# openstack image create \
   --container-format bare \
   --disk-format qcow2 \
   --file freebsd-13.0-zfs.qcow2 \
   --progress \
   freebsd-13

# openstack image set \
   --public \
   --protected \
   freebsd-13
```

El usuario predefinido es `freebsd`.

Referencias:

- BSD Cloud Images
- Documentación oficial `image create`
- Documentación oficial `image set`

7.3.7. Windows Server 2012 R2

Descargamos la imagen de Windows Server 2012 R de la web oficial de Windows Cloud Images.

Seleccionamos la imagen para KVM con formato QCOW2 y la descargamos en nuestro equipo.

Una vez que hemos descargado la imagen la copiamos por `scp` al nodo de **Control** y desde ahí la copiamos al contenedor de utilidades `infra1_utility_container`.

Desde el nodo de Control podemos acceder al sistema de archivos del contenedor de utilidades a través de la siguiente ruta:

```
/var/lib/lxc/infra1_utility_container-ID/rootfs/
```

Donde `ID` será el identificador del contenedor.

```
# cd /tmp

# openstack image create \
   --container-format bare \
   --disk-format qcow2 \
   --file windows_server_2012_r2_standard_eval_kvm_20170321.qcow2  \
   --progress \
   --property os_type=windows \
   windows-server-2012

# openstack image set \
   --public \
   --protected \
   windows-server-2012
```

Referencias:

- Windows Cloud Images
- OpenStack Windows Server 2012 R2 Evaluation Image
- Documentación oficial `image create`
- Documentación oficial `image set`

7.3.8. OpenSuse 15.2

```
# cd /tmp

# wget https://download.opensuse.org/repositories/Cloud:/Images:/Leap_15.2/images/openSUSE-
Leap-15.2-OpenStack.x86_64.qcow2

# openstack image create \
   --container-format bare \
   --disk-format qcow2 \
   --file openSUSE-Leap-15.2-OpenStack.x86_64.qcow2 \
   --progress \
   opensuse-15-2

# openstack image set \
   --public \
   --protected \
   opensuse-15-2
```

El usuario predefinido es `opensuse`.

Referencias:

- OpenSuse Cloud Images
- Documentación oficial `image create`
- Documentación oficial `image set`

7.4. Obtener un listado de las imágenes

```
openstack image list
```

Referencia:

- Documentación oficial `image list`

7.5. Eliminar una imagen

```
openstack image set --unprotected <ID>
openstack image delete <ID>
```

Donde <ID> puede ser el identificador o el nombre de la imagen.

Referencias:

- Documentación oficial `image set`
- Documentación oficial `image delete`

7.6. Crear imágenes para OpenStack

Referencias:

- OpenStack Virtual Machine Image Guide
- Example: Ubuntu image.
- Creación de una imagen OpenStack para desarrollo con XUbuntu.
- How to Quickly Create Your Own OpenStack Image.
- Create a custom Ubuntu image for OpenStack using VirtualBox.

7.7. Creación de los flavors

```
openstack flavor create m1.nano \
  --ram 128 \
  --disk 10 \
  --vcpus 1 \
  --id 1
```

```
openstack flavor create m1.micro \
  --ram 256 \
  --disk 10 \
  --vcpus 1 \
  --id 2
```

```
openstack flavor create m1.mini \
  --ram 512 \
  --disk 10 \
  --vcpus 1 \
  --id 3
```

```
openstack flavor create m1.normal \
  --ram 1024 \
  --disk 10 \
  --vcpus 2 \
  --id 4
```

```
openstack flavor create m1.medium \
  --ram 2048 \
  --disk 10 \
  --vcpus 2 \
  --id 5
```

```
openstack flavor create m1.large \
  --ram 4096 \
  --disk 20 \
  --vcpus 4 \
  --id 6
```

```
openstack flavor create m1.xlarge \
  --ram 8192 \
  --disk 20 \
  --vcpus 4 \
  --id 7
```

Tabla resumen con la lista de flavors disponibles.

```
+----+-----------+------+------+-----------+-------+-----------+
| ID | Name      |  RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+------+------+-----------+-------+-----------+
| 1  | m1.nano   |  128 |   10 |         0 |     1 | True      |
| 2  | m1.micro  |  256 |   10 |         0 |     1 | True      |
| 3  | m1.mini   |  512 |   10 |         0 |     1 | True      |
| 4  | m1.normal | 1024 |   10 |         0 |     2 | True      |
| 5  | m1.medium | 2048 |   10 |         0 |     2 | True      |
| 6  | m1.large  | 4096 |   20 |         0 |     4 | True      |
| 7  | m1.xlarge | 8192 |   20 |         0 |     4 | True      |
+----+-----------+------+------+-----------+-------+-----------+
```

Referencias:

- Documentación oficial `flavor`

7.8. Creación de una red externa

```
openstack network create \
  --external \
  --provider-physical-network exit \
  --provider-network-type flat \
  --enable \
  --description "Red Externa" \
  red-externa
```

Entorno de desarrollo

En el entorno de desarrollo vamos a utilizar una red de **clase C** y el *pool* de direcciones IPs

flotantes que vamos a asignar en el entorno de desarrollo está entre comprendido entre las direcciones `192.168.0.200` y `192.168.0.254`, por lo tanto, en este entorno solo dispondremos de `55` direcciones IPs flotantes.

```
openstack subnet create \
  --network red-externa \
  --allocation-pool start=192.168.0.200,end=192.168.0.254 \
  --dns-nameserver 192.168.0.100 \
  --gateway 192.168.0.100 \
  --subnet-range 192.168.0.0/24 \
  --description "Subred Externa" \
  subred-externa
```

Entorno de producción

En el entorno de producción vamos a utilizar una red de **clase B** y el *pool* de direcciones IPs flotantes que vamos a asignar está comprendido entre las direcciones `172.16.10.1` y `172.16.14.254`, por lo tanto, en este entorno dispondremos de `1270` direcciones IPs flotantes.

```
openstack subnet create \
  --network red-externa \
  --allocation-pool start=172.16.10.1,end=172.16.14.254 \
  --dns-nameserver 172.16.0.13 \
  --gateway 172.16.0.1 \
  --subnet-range 172.16.0.0/16 \
  --description "Subred Externa" \
  subred-externa
```

Referencias:

- https://github.com/iesgn/openstack-debian-ansible
- Documentación oficial `network`
- Documentación oficial `subnet`

7.9. Crear una red y un router

```
openstack network create \
  --enable \
  --description "Red Celia Cloud" \
  celia-net
```

```
openstack subnet create \
  --network celia-net \
  --subnet-range 10.0.0.0/24 \
  --dns-nameserver 172.16.0.13 \
  celia-subnet
```

En el entorno de desarrollo habrá que modificar el valor del parámetro `--dns-nameserver` para que apunte el servidor DNS `192.168.0.100`.

```
openstack subnet create \
  --network celia-net \
  --subnet-range 10.0.0.0/24 \
  --dns-nameserver 192.168.0.100 \
  celia-subnet
```

```
openstack router create \
    --project admin \
    celia-router
```

```
openstack router add subnet \
    celia-router \
    celia-subnet
```

```
openstack router set \
    --external-gateway red-externa \
    celia-router
```

7.10. Añadimos nuevas reglas al grupo de seguridad

Permitimos el tráfico SSH.

```
openstack security group rule create \
  --protocol tcp \
  --remote-ip 0.0.0.0/0 \
  --dst-port 22 \
  default
```

Permitimos el tráfico ICMP.

```
openstack security group rule create \
  --protocol icmp \
  default
```

Referencias:

- Documentación oficial `security group rule`

7.11. Creación de proyectos y usuarios

Referencias:

- Administración básica con CLI. IES Gonzalo Nazareno
- Administración de usuarios. UAL

8. Operaciones de mantenimiento

8.1. Reiniciar los servicios

Referencia:

- Restarting OpenStack services

8.2. Destruir y volver a crear todos los contenedores

Desde el nodo **Bastión**:

```
# cd /opt/openstack-ansible/playbooks
# openstack-ansible lxc-containers-destroy.yml
# openstack-ansible lxc-containers-create.yml
```

Referencia:

- Destroy and recreate containers.

8.3. Cómo comprobar el estado de los servicios

Para comprobar el estado de los servicios podemos ejecutar el comando:

```
systemctl
```

o

```
systemctl status
```

sobre los nodos de **Control**, **Cómputo**, **Almacenamiento** y sobre los **contenedores LXC** que se ejecutan en el nodo de **Control**.

8.4. Cómo reiniciar los servicios

8.4.1. Image service

Desde el contenedor `infra1_glance_container` que está en el **nodo de Control**.

```
lxc-attach -n `lxc-ls -1 | grep glance_container | head -n 1`
```

```
# systemctl restart glance-api
```

8.4.2. Compute service (Nodo de control)

Desde el contenedor `infra1_nova_api_container` que está en el **nodo de Control**.

```
lxc-attach -n `lxc-ls -1 | grep nova_api_container | head -n 1`
```

```
# systemctl restart nova-api-os-compute
# systemctl restart nova-scheduler
# systemctl restart nova-conductor
# systemctl restart nova-api-metadata
```

8.4.3. Compute service (Nodo de cómputo)

Desde el **nodo de Cómputo**.

```
# systemctl restart nova-compute
```

8.4.4. Networking service (Nodo de control)

Desde el **contenedor** `infra1_neutron_server_container` que está en el **nodo de Control**.

```
lxc-attach -n `lxc-ls -1 | grep infra1_neutron_server_container | head -n 1`
```

```
# systemctl restart neutron-server
```

Los siguientes servicios se están ejecutando directamente sobre el **nodo de Control**.

```
# systemctl restart neutron-dhcp-agent
# systemctl restart neutron-l3-agent
# systemctl restart neutron-metadata-agent
# systemctl restart neutron-linuxbridge-agent
```

8.4.5. Networking service (Nodo de cómputo)

Desde el **nodo de Control**.

```
# systemctl restart neutron-linuxbridge-agent
```

8.4.6. Block Storage service (Nodo de control)

Desde el contenedor `infra1_cinder_api_container` que está en el **nodo de Control**.

```
lxc-attach -n `lxc-ls -1 | grep infra1_cinder_api_container | head -n 1`
```

```
# systemctl restart cinder-api
# systemctl restart cinder-scheduler
```

9. Errores en los servidores Dell

9.1. Lifecycle Controller not available

El error *Lifecycle Controller not available* ocurre cuando existe otro proceso que está haciendo uso de iDRAC.

Solución

La documentación oficial de Dell dice que debemos esperar `30` minutos para que finalice el proceso que está haciendo uso de iDRAC. Una vez pasado este tiempo se puede volver a reiniciar el servidor e intentar acceder la utilidad de Lifecycle Controller.

Es posible conocer la lista de procesos que están en cola y consultar cuál es su estado desde la interfaz web de iDRAC.

Si pasado un tiempo el problema no se soluciona y no aparece ningún proceso en la cola, podemos conectarnos por SSH al interfaz iDRAC del servidor y resetear el iDRAC (*Integrated Dell Remote Access Controller*).

Ejemplo

Paso 1. Nos conectamos por SSH al interfaz iDRAC del servidor.

```
# ssh root@192.168.1.2
```

Paso 2. Ejecutamos el comando `racreset` para resetear el iDRAC.

```
racadm>> racreset soft
```

A continuación se muetra la ayuda del comando `racreset`.

```
racadm>>help racreset

racreset -- perform a RAC reset operation

Usage:

racadm racreset <type>
racadm racreset <type> -f

-------------------------------------------------------------------------------

Valid Options:

<type>  : What type of reset to perform (soft or hard).  Must be one of:
          soft : perform a soft reset
          hard : perform a hard reset
-f      : force racreset

-------------------------------------------------------------------------------

Usage Examples:

- Perform a soft reset of the idrac
  racadm racreset soft

- Perform a hard reset of the idrac
  racadm racreset hard

- Force a soft reset of the idrac
  racadm racreset soft -f

- Force a hard reset of the idrac
  racadm racreset hard -f

-------------------------------------------------------------------------------
```

10. Errores durante la instalación de OpenStack

A continuación se describen algunos de los errores que han aparecido durante la instalación de OpenStack.

10.1. Error al ejecutar: `openstack-ansible setup-openstack.yml`

Al ejecutar el playbook `openstack-ansible setup-openstack.yml` obtenemos un error. La instalación se queda detenida en este paso.

```
[18336.924339] Out of memory: Killed process 148504 (mariadbd) total-vm:10194672kB, anon-
rss:316400kB, file-rss:0kB, shmem-rss:0kB, UID:106 pgtables:1372kB oom_score_adj:0
[20643.129225] Out of memory: Killed process 153371 (beam.smp) total-vm:3342996kB, anon-
rss:192340kB, file-rss:0kB, shmem-rss:57388kB, UID:998 pgtables:884kB oom_score_adj:0
```

Solución: Aumentar la RAM del equipo o ampliar la **swap**.

10.2. Error al ejecutar: `openstack-ansible setup-hosts.yml`

Durante la ejecución del playbook `openstack-ansible setup-hosts.yml` se puede producir un error al reiniciar el servicio **nginx** del contenedor **infra1_repo_container** porque no es posible iniciarlo en la dirección IP y puerto que se ha configurado.

Cuando ocurre este error podemos conectarnos al contenedor **infra1_repo_container** y revisar si el archivo de configuración de **nginx** es correcto ejecutando el comando:

```
nginx -t
```

Si el comando se ejecuta sin mostrar ningún mensaje de error, entonces la configuración es correcta, sin embargo si hay algún error con la dirección IP y el puerto nos aparcerá un mensaje similar a éste:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: [emerg] bind() to 172.29.238.113:9191 failed (99: Cannot assign requested address)
nginx: configuration file /etc/nginx/nginx.conf test failed
```

Solución: Revisar la configuración del archivo `openstack_user_config.yml` porque puede

ser que la dirección IP que se haya configurado en el parámetro `external_lb_vip_address` sea incorrecta. Esta dirección tiene que tener la dirección IP pública del nodo de control

10.3. Error al subir una imagen a OpenStack desde el panel web de Horizon

La opción de subir imágenes a OpenStack desde el panel web de Horizon está desactivada por defecto, sin embargo, sí se pueden subir desde la utilidad de línea de comandos del contenedor de utilidades del **nodo de Control**.

En la documentación oficial explican que hay que configurar la variable `HORIZON_IMAGES_UPLOAD_MODE` con el valor `direct`, en el servicio de **Horizon** para permitir subir imágenes directamente al servicio Glance desde el panel web de Horizon.

Referencia:

- Horizon's settings. `HORIZON_IMAGES_UPLOAD_MODE`

10.4. Error al crear un volumen LVM

Si el servicio de cinder se ha configurado para trabajar con volúmenes LVM y iSCSI hay que conectarse al nodo de almacenamiento y comprobar que el volumen de cinder está creado.

```
vgdisplay
```

Si no aparece en el listado habrá que crearlo.

En nuestro caso nos aparecía el siguiente mensaje de error en el entorno de desarrollo.

```
Device /dev/sdb excluded by a filter.
```

Para solucionarlo, editamos el archivo de configuración de LVM.

```
# nano /etc/lvm/lvm.conf
```

Revisamos los filtros activos.

```
devices {
   ...
   filter = [ "a/.*/" ]
   ...
```

Reiniciamos el servicio de LVM.

```
# systemctl restart lvm
```

Referencia:

- LVM and multipathing – sample LVM filter strings

10.5. Cómo reiniciar el servicio `libvirtd` en el nodo de Cómputo

En primer lugar detenemos el servicio `libvirtd`:

```
systemctl stop libvirtd
```

Reiniciamos los siguientes servicios:

```
systemctl restart libvirtd-ro.socket
systemctl restart libvirtd-tls.socket
systemctl restart libvirtd.socket
systemctl restart libvirtd-admin.socket
```

Volvemos a iniciar el servicio `libvirtd`:

```
systemctl start libvirtd
```

Podemos comprobar el estado del servicio para verificar que todo está funcionando correctamente.

```
systemctl status libvirtd
```

11. Referencias

Guía oficial de instalación

- OpenStack-Ansible Deployment Guide

Anexos:

- Appendix A: Example test environment configuration.
- Appendix B: Example production environment configuration.

Ejemplos de instalación de Openstack con Ansible

- https://github.com/cyverse/openstack-ansible-host-prep
- https://github.com/iesgn/openstack-ansible_deploys

Libros

- Libro: Diseño de la arquitectura de OpenStack
- Libro: OpenStack Operations Guide
- Libro: OpenStack Security Guide
- Production Ready OpenStack - Recipes for Successful Environments.

Redes

- OpenStack Networking Guide.

Operaciones:

- OpenStack Operations Guide
- Maintenance tasks.

Administración:

- Administración de OpenStack. Universidad de Almería

OpenStackClient:

- OpenStackClient (OSC)

Recursos imprescindibles del IES Gonzalo Nazareno:

- Curso OpenStack de José Domingo Muñoz y Alberto Molina
- Introducción al Cloud Computing

- Fundamentos de Cloud Computing con OpenStack y OpenShift
- Vídeos sobre OpenStack de Alberto Molina (I)
- Vídeos sobre OpenStack de Alberto Molina (II)

Recursos de interés de la Universidad de Almería:

- https://ualmtorres.github.io/OpenStackDI/
- Guía básica de uso de OpenStack.
- Guía de operaciones de OpenStack.
- Administración de OpenStack.
- Instalación de OpenStack.
- Plantilla Heat para la creación de la infraestructura básica para la instalación de OpenStack.
- Uso de OpenStack como proveedor de infraestructura para Rancher.
- Curso Uso de OpenStack STIC.
- Scripts de post-instalación en instancias de OpenStack.
- Guía de uso de OpenStack DI.
- http://ualmtorres.github.io/howtos/CloudDIOcata/
- Eliminación de proyectos OpenStack-DI.
- Tutorial para crear una imagen OpenStack para desarrollo con Xubuntu.
- Tutorial para crear una imagen Windows 7 para OpenStack.

12. Licencia